KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Design Time Evaluation for Side-Channel Attack Resistant Cryptographic Implementations

Danilo Šijačić

Supervisors:
Prof. dr. ir. Ingrid Verbauwhede
Prof. dr. Josep Balasch

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

October 2020

Design Time Evaluation for Side-Channel Attack Resistant Cryptographic Implementations

Danilo ŠIJAČIĆ

Examination committee:
Prof. dr. ir. Jean Berlamont, chair
Prof. dr. ir. Ingrid Verbauwhede, supervisor
Prof. dr. Josep Balasch, supervisor
Prof. dr. ir. Bart Preneel
Prof. dr. ir. Wim Dehaene
Prof. dr. ir. Nele Mentens
Prof. dr. Francesco Regazzoni
(University of Amsterdam and ALaRI USI)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

October 2020

Preface

The story of COSIC and me began in the late 2013. I was a naive bachelor student, clasped by the Ann Arbor winter, looking for a master's degree program. In a very fortunate, and equally random, turn of events I met Dr. Duško Karaklajić who had encouraged me to apply for the PhD position at COSIC. For this I will always be grateful. Soon after, I did my master's thesis work in Leuven, under supervision of Dr. Begül Bilgin and Dr. Bohan Yang, in the spring of 2015. I graduated in Belgrade later that year. The next thing I remember was the summer school in Sardinia that kicked off my PhD.

This dissertation synthesizes some of my research into a, hopefully, compelling narrative. Aside from the exciting work, my years at COSIC were marked by many extraordinary people. I want to express my gratitude to all of you.

Dear Ingrid, thank you for your guidance throughout these years. I can not appreciate enough your patience and your effort to keep up with my last minute changes and submissions that were always mere minutes before the deadlines. But above all, thank you for giving me the freedom to make every mistake I possibly could—I would not have learned as much otherwise.

Dear Josep, thank you for always having the time and answers to my many questions, however reasonable they may have been. Your timely and insightful nudges always got me back on the track. Without you, few of those deadlines would have been met.

Dear Jury Members, thank you for attentively reading and evaluating this dissertation. I hope I managed to materialize your valuable feedback.

Dear Vladimir, I still remember the beers we had after my first visit to COSIC and my babbling how cool it would be to break the AES. Needless to say, I have not done much on this account. However, no matter what I did in the following years, you were there with a kind friendly word, a brilliant remark or the most rational of criticisms. Thank you for everything!

Dear Bohan, you were my first office mate and a guide to COSIC. Thank you for all the wisdom, knowledge and the affinity for scripting you instilled in me.

Dear COSICs, it was a pleasure and a privilege to work with all of you. Some of you were my office mates and co-authors. With some of you I shared lunch conversations and Friday Beers, or the joy of teaching EAGLE. Every one of you enriched my life in your own way, and for that you have my thanks. Special thanks must go the one special COSIC, namely Péla Noë. Thank you for untangling administrative messes, for organizing everything—especially the summer schools—and being there whenever I needed help.

Dear Barracks crew, also known as the Friday crew at "De Metafoor", you have my extra special thanks. Your friendship brightened many a cloudy day—one must be cognizant of the Belgian weather to fully understand this praise. Between Fridays we worked, travelled and had the best of fun together, creating memories for which I will always be grateful to all of you. Thank you for listening to my monologues on ketones, stem cells or any other fascination I somehow dug up—especially Dušan, Sara and Arthur. Miloše, thank you for proofreading my texts—the prevailing errors, if any, were introduced afterwards—and the many conversations we had. Lennert, thank you for the debugging sessions for our codes and ideas. Thomas, thank you for the never-ending discussions on becoming healthy, wealthy and wise. And Alan, thank you for the thought-provoking debates—especially the rare ones you chose to concede.

My dear family and friends, thank you for the countless hours spent talking over the internet. You were always there when I needed you the most, supporting and understanding—no matter how far apart we were. Whether you were in my home Belgrade, Jakovo, Novi Sad, Berlin, Lausanne, Madrid or elsewhere in the world, thank you for always patiently waiting for my return. I love you all!

Lastly, I want to express my gratitude to the European Commission for funding this research through the Horizon 2020 research and innovation programme Marie Skłodowska-Curie ITN ECRYPT-NET (Project Reference 643161).

Danilo Šijačić
Leuven, October 2020

Abstract

> Superhuman effort is not worth a damn unless it achieves results.
>
> Sir Ernest Henry Shackleton

Cryptographic algorithms authenticate and encrypt data to protect communication between parties. From a mathematical perspective, the implementation details at sender and receiver are not considered and modeled as a "black box". In the black-box security model algorithms are designed to resist cryptanalysis, constituted by mathematical and statistical methods. Many such algorithms are publicly available. One prominent example is the Advanced Encryption Standard (AES).

However, the black-box approach is insufficient against attackers with physical access to cryptographic devices. In this case, sensitive data can be revealed by monitoring physical emanations inherent to device operation. Information can leak through variations in execution times, power consumption, electro-magnetic radiation and so on. Such physical attacks, called side-channel analysis (SCA), have become the weakest link in the black-box security model.

In response to SCA, a plethora of countermeasures are proposed in the literature. The most researched ones induce structural or algorithmic modifications to the target cryptographic implementation. They fall into two categories: secure logic styles and masking schemes. They can be implemented using the standard complementary metal-oxide semiconductor (CMOS) technology and in the latter case provide formal security models and proofs under sets of mathematical assumptions. As such they could be easily integrated in modern digital design flows and are suitable for mass adoption. Nevertheless, electronic design automation (EDA) tools and models for reliable and efficient SCA security evaluations prior to manufacturing are lacking. Consequently, it is difficult to gauge the level of SCA security prior to chip manufacturing.

This dissertation addresses design time evaluation of SCA security for cryptographic implementations along three lines.

Firstly, we study the state-of-the-art EDA tools and simulation models used to facilitate the standard-cell application-specific integrated circuit (ASIC) design flow and how they can be used for design time evaluation of SCA security. We design and implement a framework to bridge the gap between digital design and SCA evaluation, putting together best practices from both communities. Thus, we integrate SCA evaluation into the standard-cell ASIC design flow in a wholesome and efficient manner. Lastly, we demonstrate our approach by evaluating representative cryptographic circuits. In doing so, we detect a vulnerability in a published peer-reviewed design.

Secondly, we show how to supplement theoretical considerations of masking countermeasures using detailed physical models incorporated in our framework. In particular, we demonstrate the resilience of glitch-resistant Boolean masking schemes against fault sensitivity analysis (FSA). We design an experimental setup, based on post-layout digital simulation, and argue it is the best-case scenario for FSA attackers. Under these assumptions, we experimentally verify that glitch-resistant Boolean masking schemes resist FSA.

Thirdly, we investigate the discrepancies between the analog behavior of CMOS chips and the independent leakage assumption that underlies all masking schemes. We design an experimental setup, based on transistor-level simulation, to capture the impact of layout parasitics on the SCA security of masking schemes. This setup includes the first model for co-simulation of the power distribution network and the logic core, in the context of SCA security. Thus, we provide novel insights into the potential sources of SCA leakage, that escape the abstract mathematical models of the masking schemes.

This dissertation enriches the knowledge base of SCA-resistant cryptographic implementations. We develop tools and methods to support the successful manufacturing of SCA-resistant chips, starting from the first tapeout. As a result, we enable adoption of SCA-resistant cryptography into real-world applications.

Beknopte samenvatting

In klassieke cryptografie en gegevensbeveiliging wordt communicatie met andere partijen beveiligd door middel van versleuteling. De implementatiedetails van de zender en de ontvanger worden niet beschouwd, ze worden gemodelleerd als een zogenaamde "zwarte-doos". In dit model worden algoritmen ontworpen om cryptanalytische aanvallen te weerstaan, die berusten op mathematische en statistische methoden. Deze algoritmen zijn publiek beschikbaar en een vooraanstaand voorbeeld is de Advanced Encryption Standard (AES).

Het zwarte-doos model is niet bestand tegen aanvallers die fysieke toegang hebben tot de systemen die de cryptografische bewerkingen uitvoeren. In dat geval kunnen gevoelige gegevens van cryptografische operaties worden verkregen via metingen van hun fysische kenmerken. Geheime informatie kan lekken door variaties in o.a. uitvoeringstijden, stroomverbruik en elektromagnetische straling. Deze categorie van aanvallen heet nevenkanaal analyse (NKA) en vormt de zwakste schakel in beveiliging.

Als reactie op NKA werd in de literatuur een reeks tegenmaatregelen voorgesteld. De meest onderzochte beroepen zich op structurele of algoritmische aanpassingen van de cryptografische implementatie. Ze vallen onder te delen in twee categorieën: veilige logische schakelingen en maskerschema's. Beide kunnen geïmplementeerd worden met complementaire metaaloxide-halfgeleidertechnologie (CMOS). Onder een verzameling van wiskundige veronderstellingen bieden maskerschema's bovendien formele veiligheidsmodellen en bewijzen. Dit maakt dat ze eenvoudig te integreren vallen in de moderne digitale ontwerpprocessen. Desondanks ontbreken *electronic design automation* (EDA) tools en modellen voor nevenkanaalevaluaties van elektronische chips tijdens het ontwerpproces. Bijgevolg is het moeilijk om de veiligheid van een chip tegen NKA te meten voordat de chip wordt geproduceerd.

Dit proefschrift handelt over de evaluatie van nevenkanaal analyse tegen cryptografische implementaties in de ontwerpfase in drie delen.

In het eerste deel van het proefschrift bestuderen we de huidige stand van EDA tools en simulatiemodellen die worden gebruikt om het ontwerpproces voor applicatie-specifieke geïntegreerde schakelingen (ASIC) te vergemakkelijken en hoe ze kunnen worden gebruikt voor evaluaties van NKA-beveiliging. We ontwerpen en implementeren een raamwerk om de kloof te overbruggen tussen de commerciële EDA-tools voor digitaal ontwerp en nevenkanaal evaluatie. We beschouwen zowel de front-end als de back-end stadia van het ASIC-ontwerpproces met standaardcellen. We hebben de beste praktijken van de EDA en de NKA gemeenschappen op een praktische en efficiënte manier samengevoegd door de nevenkanaalevaluaties op een methodische manier in de ASIC-ontwerpstroom met standaardcellen te integreren. Onze aanpak en ons raamwerk richten zich op de concrete implementatie en evaluatie van a priori geconstrueerde tegenmaatregelen. We tonen de haalbaarheid en de relevantie van onze simulaties door representatieve cryptografische circuits te evalueren, en door een kwetsbaarheid te detecteren in een gepubliceerd ontwerp.

In het tweede deel van het proefschrift tonen we hoe de gedetailleerde fysieke modellen van ons raamwerk de maskerschema-theorie kunnen aanvullen. In het bijzonder demonstreren we de bestendigheid van glitch-resistente Booleaanse maskerschema's tegen foutgevoeligheidsanalyse. We maken gebruik van het ruisvrije karakter en de hoge precisie van digitale simulaties om aanvallers te bevoordelen met metingen die in de praktijk onmogelijk te verkrijgen zijn. Onze experimenten tonen de bestendigheid tegen foutgevoeligheidsanalyse aan in een meest optimistisch geval voor een aanvaller en dus een pessimistische geval voor de tegenmaatregel. We voeren de experimenten uit in het back-end stadium van het ASIC-ontwerpproces met standaardcellen.

In het derde deel van het proefschrift onderzoeken we de discrepanties tussen het analoge gedrag van CMOS-chips en de onafhankelijkheidassumptie die de grondslag vormt voor maskerschema's. Verschillende recente werken tonen aan dat ondanks de bewezen veiligheid van Booleaanse maskerschema's, gevoelige informatie toch kan lekken door de layout van het ontwerp. We ontwerpen een experimentele opstelling om de impact van parasitaire effecten van de layout op de NKA-veiligheid vast te leggen. Naast de gebruikelijke verdachte koppelcapaciteiten stellen we een model voor co-simulatie van het stroomdistributienetwerk en de logische kern voor. Zo bieden we nieuwe inzichten aan in de mogelijke bronnen van NKA-lekken die ontsnappen aan de abstracte wiskundige modellen van de maskerschema's.

De bijdragen van dit proefschrift hebben tot doel de kennis rond NKA-resistente implementaties te versterken. De voorgestelde tools en methoden dienen tot het verkorten van de ontwerpcycli en het sneller tot de markt brengen van NKA-resistente chips. Als resultaat maken we een bredere adoptie van NKA-resistente cryptografie in applicaties mogelijk.

List of Abbreviations

AES Advanced Encryption Standard

AFD Analyzed Frame Data

ASIC Application-Specific Integrated Circuit

CASCADE Computer-Aided Side-Channel Analysis Design Environment

CCS Composite Current Source

CMOS Complementary Metal-Oxide Semiconductor

CMS Consolidated Masking Schemes

CPA Correlation Power Analysis

CPU Central Processing Unit

DFA Differential Fault Analysis

DOM Domain Oriented Masking

DPA Differential Power Analysis

DRC Design Rule Check

ECC Elliptic Curve Cryptography

EDA Electronic Design Automation

EDPC Exhaustive Dynamic Power Capturing

FA Fault Analysis

FI Fault Intensity

FO4 Fan-Out of 4

FPGA Field Programmable Gate Array

FS Fault Sensitivity

FSA Fault Sensitivity Analysis

FSDB Fast Signal Data Base

GE Gate Equivalent

GLN Gate-Level Netlist

GPU Graphical Processing Unit

HD Hamming Distance

HDL Hardware Description Language

HW Hamming Weight

iMDPL improved Masked Dual-rail Precharge Logic

IoT Internet of Things

IP Intellectual Property

LEF Library Exchange Format

LIB Liberty Characterization Format

LMDPL LUT-based Masked Dual-rail Precharge Logic

LUT Look-Up Table

MAC Message Authentication Code

MCU Microcontroller Unit

MDPL Masked Dual-rail Precharge Logic

MIA Mutual Information Analysis

MSM Marching Sticks Model

MtD Measurements-to-Disclosure

NLDM Non-Linear Delay Model

PAR Place and Route

PDN Power Distribution Network

PFF Power Frame File

PT Synopsys PrimeTime

RAM Random Access Memory

RFID Radio-Frequency IDentification

RSA Rivest Shamir Adleman

RTL Register Transfer Level

SABL Sense Amplifier Based Logic

SCA Side-Channel Analysis

SCADA Supervisory Control and Data Acquisition

SDC Synopsys Design Constraints

SDF Standard Delay Format

SI Signal Integrity

SNR Signal to Noise Ratio

SoC System on a Chip

SPA Simple Power Analysis

SPEF Standard Parasistic Exchange Format

SPICE Simulation Program with Integrated Circuit Emphasis

TI Threshold Implementations

TVLA Test Vector Leakage Assessment

VCD Value Change Dump

WDDL Wave Dynamic Differential Logic

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet of Things (IoT) is the emerging pinnacle of the information era we live in. It encompasses a myriad of devices, largely different in their technical capabilities and intended use. IoT devices are made to be capable of communicating together, revolutionizing the way we experience life. Its roots date back to the 1970s' idea of identifying things using bar codes, and the *pervasive computing* paradigm put forward in the 1980s. Enabled by the mass adoption of the Internet in the late 1990s, IoT is leading the civilization into a world of unfathomable, best described with a question: "What happens when things start to think?" [54]. The current estimated value of the IoT market of US $190 billion is a small fraction of the 2026 prediction by Bloomberg [21] surpassing US $1.1 trillion. A similar study [138] shows that the 26.66 billion connected devices in 2019 are going to be nearly tripled until 2025 to 75.44 billion. Such a staggering growth rate would not be possible without the IoT pervading into every aspect of societal needs. It presents a natural evolution of Supervisory Control and Data Acquisition (SCADA) systems widely used for control of industrial processes. Production and distribution of goods varying from ballpoint pen tips to nuclear reactors are made more efficient, cheaper and safer by the swarms of IoT sensors and actuators. Nevertheless, industrial automation is just a tip of the "IoT-berg". Vehicle-to-vehicle and vehicle-to-infrastructure communication is an invaluable aspect of autonomous driving. Home automation applications, or *domotics*, can provide anyone with services previously available only to the fortunate ones to have living-in butlers, maids and security personnel. Various wearable devices can be used to track our vitals and gain intimate knowledge of our behavior, biology and health, better than any doctor. When this is not enough, IoT can reach into our physical selves via sub-dermal and cranial medical implants. The line between treating humans and things

grows thinner when Radio-Frequency IDentification (RFID) tags can be used for identifying small items, shipping containers as well as humans [7]. In addition to physically attached devices, hand-held devices such as (smart) mobile phones and tablets seem to be bound to humans via digital umbilical cords. Constantly fed with media inputs humans became a valuable part of this internet. Humans are given access to digital commerce, banking, entertainment and communication. Internet allows sharing both physical (*e.g.* Airbnb, Uber) and digital (*e.g.* Amazon Web Services) assets, for the betterment of business and pleasure. In exchange, this internet demands their constant involvement, attention and streams of personal data. To sum up, combining all IoT devices—ranging from miniscule ones, spread around the globe, to gargantuan cloud servers and data centers—together with its human users forms an entity better described as *internet of everything*. An entity, with an unparalleled leverage over humanity; baring fictional characters. It stands to reason to keep this ubiquitous entity benevolent. Doing so requires *dependable* and *secure* operation of all of its electronic devices, throughout their entire *lifecycle*.

1.1 Dependability and Device Lifecycle

Dependability[1] starts with a steady supply of devices to the global market, sustaining uninterrupted service. Devices need to be manufactured with high levels of assurance to prevent recalls and consequent service congestions. Last but not least, for a reliable service devices must adhere to the ever-growing demand for computational power. Across the spectrum of computational devices constraints are getting tighter too. High-end devices flourished following Moore's law and increased in clock speeds during the past decades. As further transistor scaling becomes nearly impossible and clock frequencies above 3–4 GHz range easily lead to thermal meltdowns, alternatives for further growth must be found. In their Turing Award winner lecture [65], Hennessy and Patterson point out that designing novel hardware architecture is the best path to take. Although low-end devices are manufactured in older technologies and use clocks far below the GHz range, they are subject to stringent monetary and resource utilization (*e.g.* area, power and energy) constraints. Therefore, Hennessy and Patterson's argument easily extends to the entire electronic landscape and we can expect the increase in the volume and rate of production.

The majority of modern electronic devices are built as Application-Specific Integrated Circuit (ASIC) devices. ASICs include but are not limited to: Central Processing Units (CPUs), Graphical Processing Units (GPUs), Microcontroller Units (MCUs), Field Programmable Gate Arrays (FPGAs) and dedicated

[1] This definition of dependability is not to be confused with fault tolerance.

devices. Modern embedded systems often feature a combination of said units integrated into a single System on a Chip (SoC). We outline the chronological steps of the ASIC hardware lifecycle in Figure 1.1.

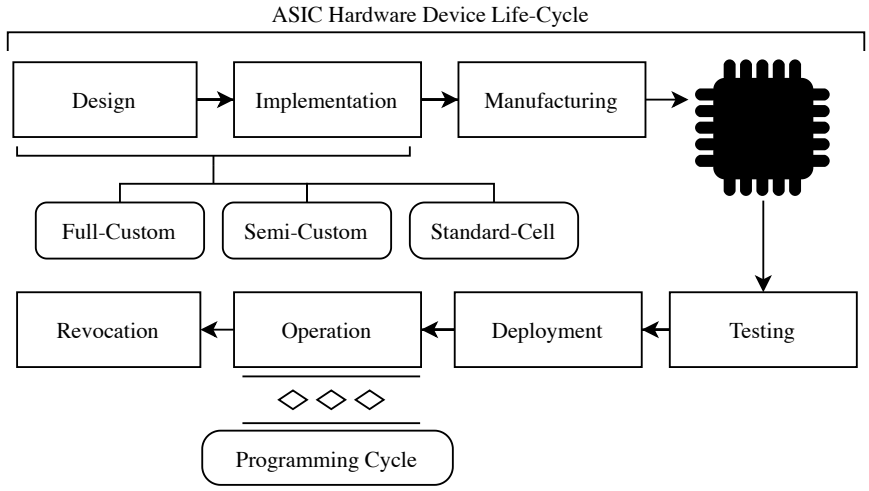


Figure 1.1: Lifecycle of ASIC devices.

Design and Implementation. The design and implementation steps are intertwined in a number of ways to ensure short time-to-market and high level of quality assurance. Electronic Design Automation (EDA) is essential during each of these steps. EDA tools and methods use different levels of abstraction, *i.e.* models, allowing fast prototyping and pre-silicon testing using simulations. Full-custom ASIC design involves creating each system component "from scratch". It yields the optimal circuits, at the cost of design time and immense level of expertize needed. Standard-Cell ASIC design is based on using building blocks or *cells* from pre-made full-custom digital libraries (Section 2.1). It dominates digital industry for increased reliability and ease of design that ensure market presence, outweighing the costs and benefits of a full-custom approach. Semi-custom ASIC design is a compromise between the latter two approaches. Regardless of the approach, masks for photolithography of the chip, or *layouts*, are exported in standardized formats and sent to the manufacturing facilities for *tapeout*[2].

[2]Historically, the name originates from shipping physical tapes containing photolithography masks out to the manufacturing facilities.

Manufacturing. Manufacturing consists of silicon wafers fabrication, processing and packaging. Chip layouts can be designed and implemented anywhere in the world and sent securely over the internet to the manufacturing facilities or *foundries*. The majority of foundries are spread around East Asia. Therefore, despite the global development and deployment, most chips are manufactured in this locale. As political and economic currents fluctuate, ensuring an uninterrupted manufacturing and a reliable supply chain is a non-trivial issue. Common threats include Intellectual Property (IP) theft and the insertion of *hardware Trojans* [140].

Testing. Newly manufactured chips need to be functionally tested and classified based on their performance. Testing is highly automated and performed directly at the manufacturing facility to save the testing time, as it impacts the price of production significantly.

Deployment. Chips that pass the testing are passed down the supply chain to trusted vendors where they can be purchased and deployed for a specific application. Deployment includes initial programming, configuration and integration with the desired product, followed by the launch to market.

Operation. An ASIC device can be incorporated into a myriad of products in the market. Depending on the type of device and its purpose, the operation step consists of programing cycles. Devices running user software, such as desktop CPUs and mobile SoCs, run a variety of software applications under one or more operating systems. They are updated many times during the operation step. Industrial microcontrollers on the other hand often run a single routine for many years. FPGA devices can be programmed, *i.e.* reconfigured using different bitstreams.

Revocation. Revocation or removal of electronic devices from the field is the final, often overlooked, step. Consumer-grade devices simply run until failure. Different industrial-grade devices must guarantee service for a number of years (*e.g.* 20 years for automotive, 50 years for aeronautical). As devices contain increasingly sensitive data, proper policies for handling them after the end of operation need to be set.

In summary, production steps represent a small fraction of the device lifecycle. Unlike software that can easily be patched after deployment, bugs in hardware design persist until revocation. Modern design flows and EDA tools are built around reliable detection and prevention of hardware bugs that would impede

the functionality and the performance before they go viral in the market. However, this is not the case for security vulnerabilities. Recent attacks by Lipp *et al.* [89] and Kocher *et al.* [80] exploit such hardware vulnerabilities in Intel processors.

1.2 Security and Cryptography

Security is never a standalone product, but an enabling technology. As such, it is often squeezed out from the profit margins or left as an afterthought under the pretext of "what could go wrong". The large influx of pervasive electronic devices combined with neglect of security imposes risks to physical health and safety—as shown by Miller and Valasek [98, 99] and Marin *et al.* [93].

Securing electronic devices during their entire lifecycle encompasses technical and social measures. Social measures become important after deployment, as humans can easily be manipulated to circumvent the protection mechanisms using *social engineering*[3]. Technical measures are the backbone and must be set in place during the design step and present during all subsequent steps. They center around *cryptology*, a science conceived shortly preceding the invention of the written language. Cryptology enables secure communication and computation in an adversarial environment. For thousands of years it has almost exclusively been the matter of the secret societies, military and diplomatic corps. In 1883, Kerckhoffs postulated six principles [73, 74], the second of which today remains known as "Kerckhoffs' principle". Given a keyed algorithm, *i.e.* a *cipher*, security is preserved if and only if the key is kept secret. All other details including the design rational and cipher functionality can be made public, without impacting security[4]. Kerckhoffs' principle is the foundation for modern cipher design and the *black-box* security model. In this model ciphers are viewed as purely mathematical constructs. Cipher outputs are sufficiently large and computationally indistinguishable from uniform random data to adversaries with finite computational power. Cryptography is the synthetic part of cryptology. Traditionally, cryptographic primitives are used to protect messages in transit, *i.e.* communication, and messages at rest, *i.e.* storage. Said primitives are commonly divided into three branches:

- symmetric-key primitives: communicating parties are in possession of the same pre-shared *secret* key,

[3] All social aspects of security, ranging from inadequate policies and improper defaults to industrial espionage, are outside of the scope of this dissertation.

[4] Algorithms that deviate from this principle cannot be vetted through rigorous public scrutiny. Hence, such algorithms should be avoided, no matter who the authors are.

- public-key primitives: each party possesses a pair of mutually dependent keys, the party's *private* key and a *public* key known to everyone,
- keyless primitives.

Symmetric-key primitives. Symmetric-key primitives, *i.e.* ciphers, *encrypt* an input message, called *plaintext*, into a *ciphertext* using a secret key. The reversed operation is called *decryption*. Secret key sizes of 80, 128 and 256 bits are recommended for short-, moderate- and long-term security, respectively [58]. They include three types of primitives. Block-ciphers operate on words of constant size, *i.e.* block size, including the plaintext, the ciphertext and the internal state. Notable examples include the Advanced Encryption Standard (AES) [34] and PRESENT [24]. Stream-ciphers use the internal state of fixed size to create a pseudo-random bit-stream used to encrypt the arbitrary-size plaintext. A notable example is Chacha20 [37]. Permutation-based primitives permute a relatively large state acting as an entropy pool, absorbing plaintext blocks into the state between permutations to create ciphertext blocks. A notable example is Ascon [40]. Symmetric-key primitives excel at efficiency but require a secure channel for key exchange.

Public-key primitives. Public-key primitives excel at key management and establishing secure channels, but lack efficiency. Each party has to generate and manage its own private key, while the public key can be broadcasted freely over an insecure environment. Advances in the area of *quantum computing* in conjunction with the Shor's algorithm [135] pose a potential future threat for the public-key algorithms currently in use. Notable examples of public-key primitives are the Rivest Shamir Adleman (RSA) [126] cryptosystem and Elliptic Curve Cryptography (ECC) [100, 79, 96].

Keyless primitives. Keyless primitives involve cryptographic checksums, also known as *hash functions*. They are versatile primitives used in combination with both symmetric- and public-key primitives. Notable hash functions include: RIPEMD-160 [39], SHA2 [111] and SHA3 [11].

Primitives from different branches of cryptography are used to fulfill *security goals*. While we list the fundamental ones, an interested reader can find a more comprehensive list in [97].

Confidentiality. Data confidentiality or secrecy is the oldest security goal. It prevents any unintended recipients from reading the contents of a message. It

serves to protect everything from private information of individuals to industrial and governmental trade secrets. Confidentiality is commonly achieved using symmetric-key primitives due to their computational efficiency.

Authenticity. Authenticity of data and entities has to be ensured to prevent unauthorized players from assuming roles in a system. In case of data its origin can be authenticated using *digital signatures* or Message Authentication Codes (MACs). The former are built using public-key primitives. The latter are built using different *modes of operation* of stream or block ciphers *e.g.* AES-GCM [95], dedicated *authenticated encryption* [127] constructs *e.g.* AEGIS-128 [150] and AES-OCB [83] or using hash functions *e.g.* HMAC [10]. Entity authentication confirms their identity to match one of authorized players in the system. Similarly to the data authenticity, it can be achieved using MACs and *digital certificates*. The latter are digitally signed identifiers issued by *certificate authorities*.

Non-repudiation. Non-repudiation is a form of a misbehavior detection mechanism. It prevents legitimate entities from denying performing their previous actions. Asymmetric knowledge in public-key cryptosystems achieves non-repudiation using digital signatures.

1.2.1 Grey-Box Security Model

The black-box security model for computational security suffices when cryptographic primitives are regarded as purely mathematical constructs adhering to the Kerckhoffs' principle. Black-box adversaries can only collect plaintext-ciphertext pairs and try to make mathematical inferences about the key. Secure cryptographic primitives require attackers to perform prohibitively many evaluations of the target primitive before making a successful inference.

For example, the best known attack against AES-256 [23, 139] requires $2^{254.27}$ evaluations of the primitive and 2^{40} plaintext-ciphertext pairs. This is considered an attack as it can recover the key with less evaluations on average than $2^{255.00}$ evaluation needed for the exhaustive search[5].

However, black-box security quickly falls apart once adversaries attain physical access to cryptographic devices. In this case they can leverage physical manifestations that inevitably accompany cryptographic computations to gain access to the cryptographic keys behind the abstract mathematical constructs.

[5]For perspective, the number of particles in the known universe is $2^{267.47}$.

Said attacks are called *physical attacks*. The black-box security model is expanded into the *grey-box* security model to reflect the capabilities of physical attackers as depicted in Figure 1.2.

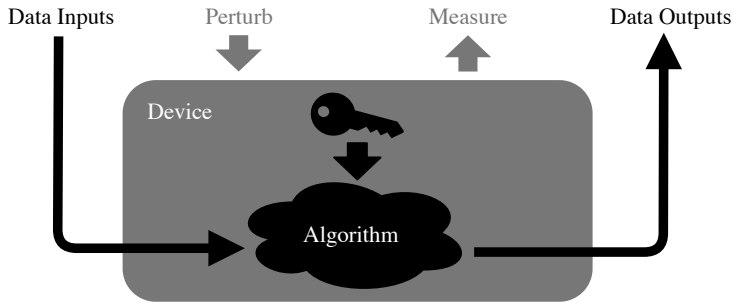


Figure 1.2: Grey-box security model.

Grey-box attackers require a trivially small number of input-output data pairs, compared to black-box attackers. This is enabled by the advantages of physical access, the ability to *perturb* and *measure* device operation in order to collect additional key-related information.

- Perturbation can be caused through manipulation of supply voltage, clock signal or operating temperature. More sophisticated attackers can use lasers and focused ion beam cannons for a finer-grained control. Regardless of the technical venue attackers take, perturbation causes devices to produce faulty outputs. Thus-obtained outputs can be exploited using Fault Analysis (FA) techniques.
- Physical emanations, such as power consumption, electromagnetic radiation and execution timing, are inherently present and correlated to device operation and data being processed. Consequently, when measured they form a communication channel that *leaks* sensitive information. Side-Channel Analysis (SCA) techniques exploit information embedded in these channels.

FA and SCA are powerful attack techniques on their own. However, in practice perturbations and measurements can be used together to mount even more devastating *combined* attacks.

1.3 About This Dissertation

In this dissertation we study design-time, *i.e.* pre-silicon, evaluations of physical vulnerabilities. We focus on the earliest—*design* and *implementation*—steps of the ASIC hardware lifecycle, Figure 1.1. We aim to preempt vulnerabilities that could be exploited during the device *Operation* step, while enabling fast time to market. Furthermore, our approach allows designers to catch vulnerabilities before the lengthy and expensive *Manufacturing* step, leading to more reliably secure designs released to the market. In particular, we focus on design-time evaluations of *side-channel* vulnerabilities for cryptographic hardware implemented using the *standard-cell* ASIC design flow.

The contributions of this dissertation can be placed along three lines of work. In the first part of this dissertation we focus on a systematic and wholesome integration of pre-silicon SCA evaluation in the standard-cell ASIC hardware design flow. In the second part, we move to demonstrate the power of experimental pre-silicon evaluations in supporting theoretical SCA security claims. Lastly, in the third part we use low-level hardware models to uncover physical phenomena escaping mathematical models for SCA security.

The remaining chapters of this dissertation are organized as follows.

Chapter 2. In this chapter we present the necessary background for this dissertation. We start by introducing the standard-cell ASIC hardware design flow, the design approach dominating digital design for decades. We detail abstractions and actions typically involved in different stages of said design flow. We describe modeling and tooling practices necessary for the success of standard-cell designs. Next we introduce physical attacks, as a major threat for secure cryptographic implementations. We briefly outline their classification, based on the attacker capabilities and socioeconomic resources. We then focus on SCA, for we deem it highly risky as it is available to a broad range of potential attackers. We outline state-of-the-art SCA attack methodologies, countermeasures and methods for accessing SCA security. Boolean masking schemes are on the forefront of the countermeasure collage, with properties that promise compliance with standard-cell design. Additionally, we accent the gap between the areas of standard-cell design and SCA. SCA practices and theoretical insights are relatively recent. In stark contrast, standard-cell design and EDA tools are field-tested veterans of many decades of research and practice in the digital electronics industry.

Chapter 3. In this chapter we consolidate state-of-the-art models and practices of the standard-cell ASIC hardware design flow with the state-of-the-art SCA evaluation methods and metrics to propose a SCA-aware hardware design flow, capable of detecting physical vulnerabilities starting from the earliest design stages. We experimentally verify the computational efficiency and efficacy of models by running design-time evaluations on several representative cryptographic circuits. We focus on instantaneous power consumption as the preferred side-channel. However, our methodology can be applied to any other channel. To further the practical significance of this approach, we design and implement Computer-Aided Side-Channel Analysis Design Environment (CASCADE). CASCADE bridges the gap between the commercial EDA tools and SCA evaluation methods, in a manner that can be easily adopted in a designer's toolbox. Lastly, we discuss the design rationale and implementation details of CASCADE, following with performance benchmarks.

This work was originally published in [155]. An extended version of this work was published in [153].

Chapter 4. In this chapter we use the EDA tools and models incorporated in CASCADE to evaluate resistance of glitch-resistant masking schemes against fault sensitivity analysis. We start from theoretical considerations and experimentally verify the findings. We also demonstrate the versatility of CASCADE, extending it to evaluations beyond passive SCA.

This work was originally published in [156].

Chapter 5. In this chapter we investigate the potential causes of the "out of model" leakage present in digital circuits protected using Boolean masking schemes. We employ detailed backend simulations using Simulation Program with Integrated Circuit Emphasis (SPICE) to capture the physical effects that are abstracted away in the mathematical models used to craft Boolean masking schemes. We provide valuable insights into the impact of the Power Distribution Network (PDN) and a plethora of parasitic components inevitably present in digital circuit layouts. We show the qualitative difference in the impact of different parasitic components to the SCA security.

This work was originally published in [154].

Chapter 6. In this chapter we conclude the dissertation, summarizing its contributions. We provide several insights that follow from the obtained results. Accordingly, we propose several directions for future research.

Chapter 2

Background

> Whenever you find yourself on the side of the majority, it is time to pause and reflect.
>
> Mark Twain

In this chapter we introduce the standard-cell ASIC design flow and Side-Channel Analysis (SCA) attacks on cryptographic implementations.

Standard-cell ASIC design flow is extensively documented in the literature [120, 70, 14]. As the most widely used design flow for digital circuits its facilitation can be grouped together differently, to cater to different optimization targets. In this chapter we delineate and summarize its aspects relevant for this dissertation. Without the loss of generality we focus on the design of cryptographic Intellectual Property (IP) cores. Cryptographic IP cores may include hardware-accelerated instructions set extensions (*e.g.* AES-NI), trusted computing cores (*e.g.* Sancus [114]) and the entire secure enclaves (*e.g.* Apple T2 Security Chip). Any system-level design and integration with other IP blocks are out of scope. As we are setting the stage for designing SCA attack resistant IP cores, we emphasize the *logic simulation steps* across all levels of abstraction.

SCA is a class of physical attacks on cryptographic implementations. In gist SCA attackers look for a relation between a physical measurement of the cryptographic implementation (*e.g.* power consumption) and the data it is processing in hopes of retrieving the secret information from the device. In this chapter we introduce physical attacks, classify them and discuss the

state-of-the-art and SCA evaluation techniques.

SCA upsets the well-established digital design practices by requiring a new design criteria, namely SCA resistance. We conclude this chapter by delineating the contrast between the systematic and methodical practices of the standard-cell ASIC design flow and the missing measures for design time SCA evaluation for cryptographic implementations.

2.1 Standard-Cell ASIC Design Flow

Standard-cell ASIC design flow is based on assembling *a priori* designed logic gates into the desired circuit. Said gates, or *cells*, are organized in collections, or *libraries*, for a designated technology node. Each standard-cell has a pre-designed layout and electrical characteristics. Physical properties of a standard-cell are abstracted away through *cell characterization* and replaced with piecewise linear models. Thus, standard-cell ASIC design flow greatly reduces designers' effort, as it decouples logic functionality from physical design. It is therefore a "divide and conquer" approach across increasingly complex abstraction levels. Standard-cell libraries typically include: combinatorial, sequential and physical cells. Combinatorial cells include simple gates such as inverters, `AND` or `XOR` gates; and more complex ones such as multiplexers and adder slices. Sequential cells include latches and flip-flops with different control amenities such as synchronous or asynchronous reset, write enable signal or a multiplexer for scan insertion. Physical cells include signal distribution cells such as clock buffers and filler cells, necessary to ensure uninterrupted well doping. Most of modern commercial libraries are based on the Complementary Metal-Oxide Semiconductor (CMOS) logic style and include between 300 and 500 cells. Cells come in different sizes, allowing different driving strengths for the same logic functionality.

Gate Equivalent (GE) is the most common metric of circuit complexity for standard-cell ASIC designs [70]. The number of GE expresses the circuit complexity normalized to the complexity of the two-input `NAND` gate with minimal driving strength from the target library. The complexity of the two-input `NAND` gate is measured by either its area or number of transistors (four). The number of GE is a good preliminary estimate for comparing designs for the same standard-cell library. Nevertheless, as a normalized metric, GE is not entirely adequate for comparing designs implemented using different libraries and technology nodes. An alternative metric, focused on normalizing timing is Fan-Out of 4 (FO4) [62]. It represents the propagation delay of a cell with

fan-out equal to four, *i.e.* with a loading capacitance four times larger than its input capacitance.

2.1.1 Design Stages

Figure 2.1 depicts the stages of the standard-cell ASIC design flow. Each stage consists of *synthetic*, *analytic* and *corrective* actions, depicted using rectangles rounded rectangles and dashed lines, respectively. We note the outputs of different design stages using ellipses. After an iteration of a synthetic step, designers must analyze thusly obtained design. Based on the feedback from the analysis tools, designers proceed to the next synthetic step or perform corrective actions. Corrections must be made if the design does not satisfy the functionality and predetermined constraints from the product specification. Constraints include but are not limited to: timing (*e.g.* setup and hold times), area, power consumption and energy per operation. As this often results in numerous tedious iterations, automation is an invaluable designers' asset.

Electronic Design Automation (EDA) tools aid synthetic and analytic actions, while the corrective actions are left for the designers[1]. The analysis includes extraction, simulation and formal verification of designs. The level of physical detail about the design increases as the flow progresses. Design information becomes more complex, requiring steep increases in the number of analysis steps and computational power for each step. Moreover, the separation between steps in practice is rarely as s"clean" as Figure 2.1 depicts. As different design steps are performed by different designer teams, additional iterations incur scheduling conflicts that can further delay the design. Delays to market are especially prominent if a design needs to be reverted to one of the previous stages. While pre-silicon design stages can cause delays in days or weeks, the highest potential delays come from manufacturing. Foundries commonly schedule tapeouts every three to four months and the process itself takes several weeks. Therefore successful tapeout is paramount for minimizing time-to-market. Pre-silicon design evaluations are invaluable for this.

Behavioral Modeling (BEH). During this stage a design specification is replaced by a black-box model. In other words, the design is represented using a mathematical relation between inputs and outputs. Behavioral modeling abstracts away all physical and logical structure of the design. Specification, normally given in C/C++ programming languages or via MATLAB or Python models, can be used to simulate the design behavior.

[1] If a corrective action can be automated it becomes a part of synthetic actions.

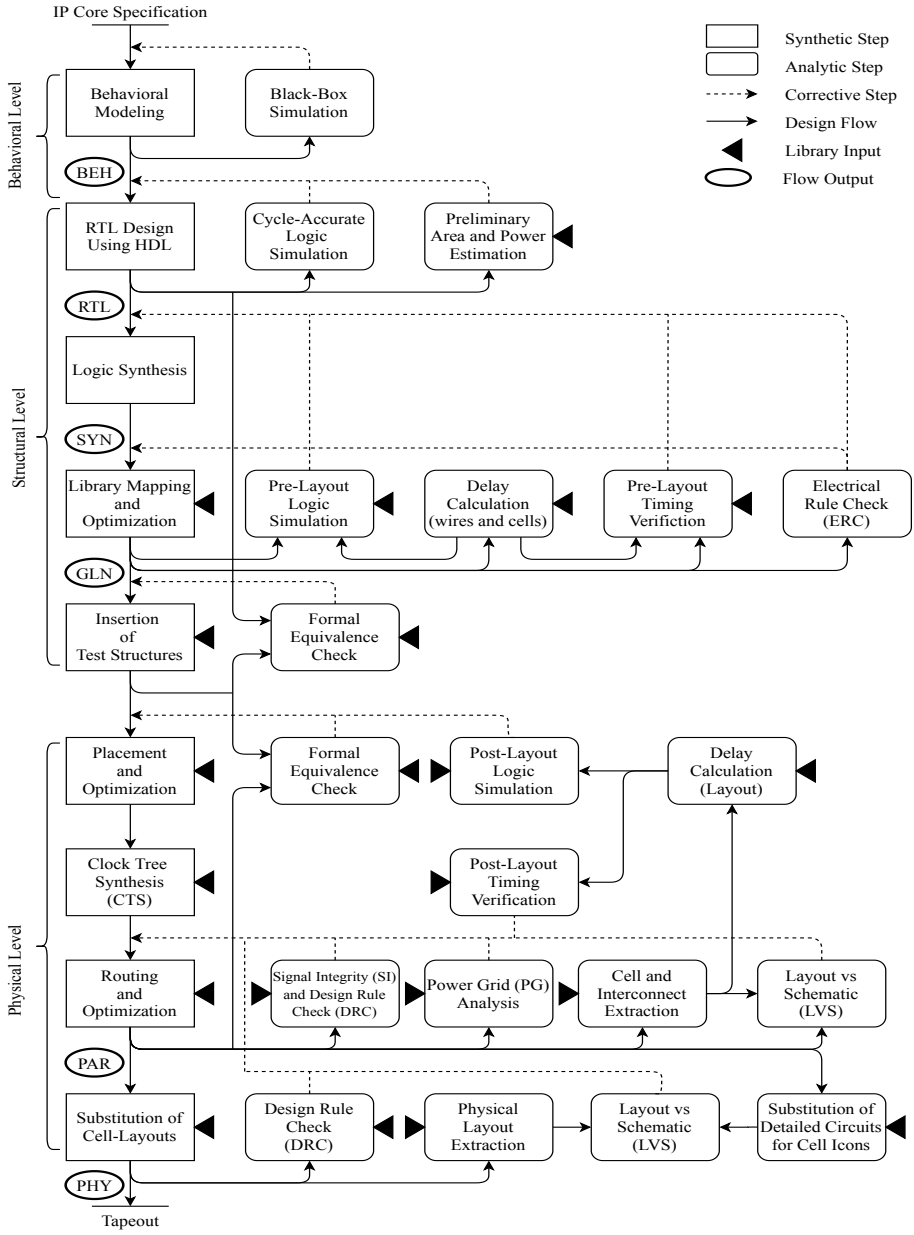


Figure 2.1: Standard-cell ASIC Design Flow.

Register Transfer Level (RTL). Structural model of a design is captured using a Hardware Description Language (HDL), such as VHDL, Verilog or System Verilog. A HDL description introduces the clock signal and segments design behavior into a sequence of transformations separated by clock cycles. State after each transformation is stored in sequential logic elements that form registers. Hence, design behavior consists of a series of data transfers between registers. No information is given on the combinatorial networks facilitating said transformations. Therefore, an RTL design can be viewed as a *zero-delay* network of registers operating in a number of consecutive cycles. Cycle-accurate, *i.e.* zero-delay, simulations are used to validate the functionality of RTL designs. The logical structure introduced in the RTL model divulges the storage requirements and the mathematical complexity of each combinatorial transformation. This allows rough estimations of the area and power costs.

Synthesis (SYN) of the Gate-Level Netlist (GLN). The Gate-Level Netlist (GLN) represents the structure of the design as a network of standard cells. It is obtained via a two step process: logic synthesis and library mapping. Logic synthesis converts the abstract inter-register transformations to a combinatorial network of generic logic gates, *e.g.* `AND`, `XOR`, `INV`. Abstract registers are replaced with generic storage cells, *e.g.* D flip-flop, SR latch, or Random Access Memory (RAM). The design obtained at this point, SYN, could be simulated using logic delta-delay, *i.e.* Δ-delay, simulation whereby each gate is assigned the same delay. However, this step is commonly skipped as synthesis and library mapping are commonly bundled together as designers often have a target library in mind. SYN output is independent of standard-cell libraries and highly portable. Thus obtained design, is then mapped to concrete cells of a particular standard cell library, constituting the GLN. Cell models include piecewise linear models for timing and power behavior and electrical rules, *e.g.* the outputs of two cells must not be connected together. Therefore, a structural view of the design is preserved while allowing physical assertions. As physical placement and routing is unknown, delays between cells are estimated based on *statistical wire-load models*. Namely, for a given standard-cell library and design size, average wire length per fanout node is determined by the library manufacturers. Each such wire is replaced using an RC network and an Elmore delay is calculated. Pre-layout delay calculation therefore allows timing verification considering statistical worst-case scenarios. Both steps are followed by logic optimization. Modern EDA tools completely automate both logic synthesis and library mapping, along with necessary checks. Consequently, a single designer script often facilitates both steps. EDA tools also fully automate addition of test structures, *e.g.* scan chains to the GLN.

Placement and Routing (PAR.) The physical placement and routing of designs translates the structural GLN into the physical layout. Firstly, placement assembles cells on a two-dimensional plain according to the geometric cell-properties from the library. The clock tree, consisting of clock buffers, is then inserted to minimize the clock skew between registers. Next, the previously abstract connections between cells are replaced by metal wires in the process of routing. EDA tools facilitate said three steps and allow various optimizations. For example, signals routed along long lines need to be buffered, or cell placement needs to be rearranged. As this can cause structural design changes, formal equivalence checks need to be performed. After routing, a plethora of metal wires distributed across multiple metal layers (over ten metal layers exist in modern libraries) is placed on top of the silicon wafer. This may cause issues with Signal Integrity (SI) or violate Design Rule Check (DRC). Issues of Power Grid (PG) need to be resolved as well, minimizing the voltage drop across power rails to ensure performance. When everything is set in place, RC parasitics surrounding cells and interconnect wires are extracted and post-layout delays are calculated. Thusly obtained delays are used to drive detailed event-drive logic simulation and timing verification. The Place and Route (PAR) stage belongs to the physical design level, as there can be many different mappings of the same GLN to the physical layout. Nevertheless, standard cells are thus far represented by structural piecewise linear models.

Physical Extraction (PHY). After the the PAR stage, standard-cell abstracts are replaced with detailed cell layouts[2]. A transistors-level schematic is then obtained from the layout and verified against the original structural schematic to ensure the same functionality. To the best of our knowledge all parasitic extraction is performed for the purpose of capturing the circuit performance. Side-channel attacks are not taken into consideration. Standard-cells with extracted parasitics are then represented using Simulation Program with Integrated Circuit Emphasis (SPICE) models called *phantom-cells*. Phantom-cells obscure the cell-layout to protect the IP of the library manufacturers. Hence even the computationally extensive SPICE simulation based on phantom-cells, while paragon of precision for performance, can not guarantee side-channel security.

[2]This step is often performed by the party entrusted by the library manufacturer with these low-level models. This is mostly not the designer.

2.1.2 Contemporary EDA Tools and Data Formats

The multi-million transistor complexity of modern digital circuits makes the EDA industry an indispensable asset of digital designers. Many open-source tools exist but the majority of the market is captured by "the big three" ASIC EDA tool vendors for design: Cadence®, Mentor Graphics®, and Synopsys®; listed in alphabetical order. Each of the big three develops tools for various design steps, initially aiming for vertical integration of design flow under a single suite or framework. The digital design market converged towards the "best in class" approach. Designers retain the discretion to compose their own toolboxes cherry picking from different vendors, resulting in a myriad of possible variations. Creating an effective and efficient design flow thusly becomes a separate issue, often an overlooked nightmare in practice [70]. To ensure coherency among each other, as well as with library manufacturers, digital designers rely on standardized data formats to interchange design information. Further interoperability is secured through a common use of Tcl shell as the control interface across the majority of EDA tools[3].

Table 2.1 outlines some of the frequent data formats used in the standard-cell design flow. In addition, a number of tools supports exports into generic text files, such as comma separated value (CSV) format.

2.1.3 Standard CMOS Cells and Library Characterization

The CMOS logic style was invented in 1963 by Wanlass and Sah [147]. Starting with the IBM's 20 μm technology node from 1961, CMOS is the predominant logic style for digital electronics in modern 7 nm technology nodes introduced by TSMC in 2018[4]. Transistor sizes shrunk over *ten thousand* times and their geometries changed from *planar MOSFET* to *FinFET* but the efficiency and robustness of CMOS remains unparalleled. Figure 2.2 shows the transistor-level schematic (left) of the CMOS inverter gate, the fundamental building block of all CMOS gates. The idealized waveforms shown in Figure 2.2 (right) demonstrate the operating principle and advantages of CMOS logic.

Firstly, CMOS cells are very energy efficient. They draw significantly smaller supply currents from the power supply when idle, compared to currents drawn when transitioning between logic states. Therefore, each gate contributes to the power consumption during a short interval of time compared to the clock period, leading to high energy efficiency and leaving enough time for

[3]Although EDA tool designers occasionally make small "improvements" of their Tcl interpreters, resulting in curious bugs.

[4]Smaller technologies are in development, but are not available in commercial products.

Table 2.1: List of common EDA data formats.

Ext.	Format	Typical use
`.v`	Verilog source	Design capture or netlist representation
`.sv`	System Verilog source	Testbench generation or verification
`.vhd`	VHDL source	Design capture or testbench generation
`.vcd`	Value Change Dump (Value Change Dump (VCD))	Output of logic simulation
`.sdf`	Standard Delay Format (Standard Delay Format (SDF))	Delay annotation for cells and interconnections
`.sdc`	Synopsys Design Constraints (Synopsys Design Constraints (SDC))	Exchange of design constraints among front-end and back-end tools
`.fsdb`	Fast Signal Data Base (Fast Signal Data Base (FSDB))	Synopsys' proprietary format for continuous waveform storage
`.spef`	Standard Parasitic Exchange File (Standard Parasistic Exchange Format (SPEF))	IEEE standard for exchange of parasitic RLC components
`.lef`	Library Exchange Format (Library Exchange Format (LEF))	Physical cell layout for PAR tools
`.lib`	Liberty Characterization Format (Liberty Characterization Format (LIB))	Human readable characterized cell information; timing, power and noise
`.db`	Compiled binary of the `.lib` file	Compiled for logic synthesis and PAR tools

the heat to dissipate. The former supply current, often referred to as *static* or *leakage* current causes *static power consumption*. The latter, *dynamic*, supply current causes the *dynamic power consumption*. Although static leakage currents increase with the technology nodes shrinking, dynamic currents remain predominant. The power consumption of each CMOS gate is highly correlated with the logic value of its data inputs as a consequence of this behavior.

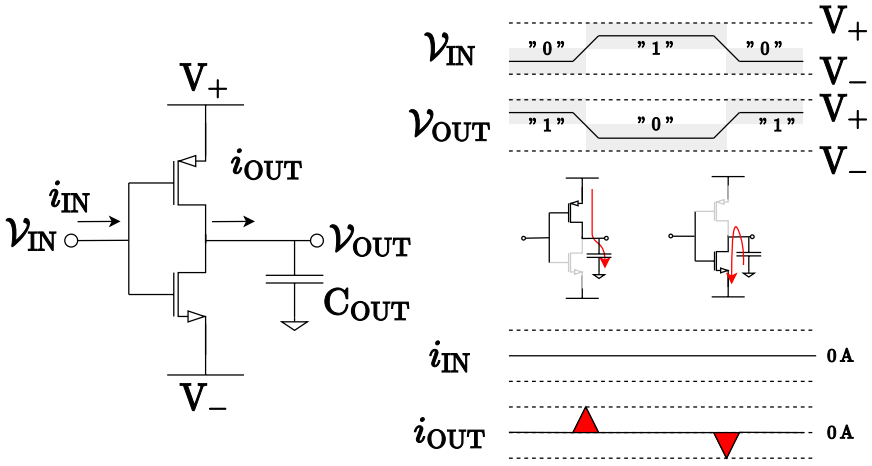


Figure 2.2: CMOS inverter transistor schematic (left), idealized voltage and current waveforms waveforms (right).

Secondly, the robustness of CMOS gates comes from amplification and level-restoration effects. As both transistors are connected in the *common-source* topology they amplify the input signal. Amplification maps a range of input values to a high or low output level, allowing digitization in the presence of noise. An oxide layer separates input from output stages of CMOS gates, preventing any significant conducting paths between the logic terminals. The voltage level at the output of CMOS gates is restored by drawing current directly from the power supply. Combined, said properties make CMOS gates easily composable. Moreover, the behavior of CMOS gates is greatly determined by the supply voltage, slope of the input signal (V_{IN}), the loading capacitance at the output ($\mathrm{C_{OUT}}$) and the operating temperature. Consequently, the complex non-linear behavior of underlying transistors can be modeled very accurately using a "handful" of linear parameters, allowing powerful models for structural cell representation. Such parameters are obtained through *characterization* of CMOS gates, focusing on timing, power and noise behavior.

Cell characterization is performed by library manufacturers. Firstly, a full-custom design of each cell is made and transistors-level schematics are obtained directly from technology parameters. Said technology parameters are closely guarded trade secrets of foundries providing standard-cell libraries. Therefore, characterization results are often the only interface between designers and technology. Algorithm 1 describes the characterization procedure using SPICE simulations, for a given standard-cell. The characterization procedure is repeated under different operating conditions, such as temperature and supply voltage.

This is how different case-corners are obtained for a given library. For example, the *worst-case* entails high temperature, commonly 125°C, and lowered supply voltage, commonly 80–90% of the nominal value [78, 67, 3].

Algorithm 1 Standard-cell characterization.

```
1: procedure CellCharacterization(n data inputs, set of p input slopes,
   set of q loading capacitances)
2:     for input in n data inputs do
3:         for state in 2^(n−1) states of other inputs do
4:             for slope in set of p input slopes do
5:                 for load in set of q loading capacitances do
6:                     analog response ← InputRise(input, state, slope, load)
7:                     pwl ← SamplePieceWiseLinear(analog response)
8:                     PopulateLUT(rise, pwl, input, state, slope, load)
9:                     analog response ← InputFall(input, state, slope, load)
10:                    pwl ← SamplePieceWiseLinear(analog response)
11:                    PopulateLUT(fall, pwl, input, state, slope, load)
12:                end for
13:            end for
14:        end for
15:    end for
16: end procedure
```

For each of the n cell-inputs, a pair of Look-Up Tables (LUTs) is created for each of the 2^{n-1} states the remaining data inputs can be in. LUTs are paired to independently capture circuit response to rising and falling stimuli. Simulations produce analog (continuous) simulations of the response to the stimuli. Analog responses are then sampled and represented using a piecewise linear representation. Thus obtained vectors are stored in the aforementioned, three-dimensional, LUTs indexed with the input slopes and capacitances of the output capacitances. Typically, the number of input slopes p and output capacitances q is set to 7.

Older technologies, between 90–130nm, were well characterized using Non-Linear Delay Models (NLDMs). NLDM record cell output voltage to characterize cell behavior. Deep sub-micron nodes, below 90nm, are better characterized using current drawn from the supply. In particular, Composite Current Source (CCS) models are considered "best in class" yielding results within 3% from SPICE [67, 110, 3]. In all cases, models are stored in the Liberty format [22]. Figure 2.3 depicts the CCS characterization of a two-input `NAND` gate; recording rising edge of one input while the other input, *i.e.* the state, is fixed.

Similarly, timing, power and noise tables are formed and stored into library

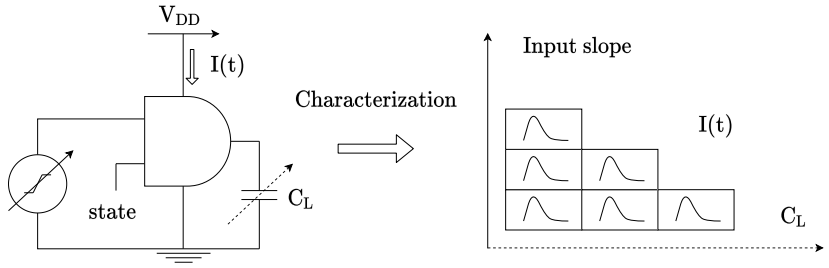


Figure 2.3: Liberty characterization of the`AND` gate supply current. Simulation setup is on the left. Visualization of the library LUT creation is on the right. Different LUT is created for each state and each transition (rising and falling) combination.

files. For a given design and constraints, these curves are used to estimate timing, power and signal integrity, respectively. Therefore, modern standard-cell libraries contain a plentitude of information about the current waveforms. The main concerns of digital designers are meeting performance criteria (timing closure) and estimation of average and peak power consumption. Average power is important to determine cooling requirements and energy consumption, while the peak consumption determines the power electronics circuitry.

2.2 Physical Attacks

Physical attacks surfaced in the academic literature in the late nineties. They allow adversaries to circumvent cryptography by exposing key material from the target device, as demonstrated by [128]. The grey-box attacker model is devised to capture the capabilities of physical attackers. The threat of grey-box attackers greatly depends on their social and monetary resources. Accordingly, physical attackers can be classified as per guidelines introduced by IBM [2].

Class I: Clever Outsiders. Intelligent individuals trying to find and exploit existing weaknesses. They are constrained by their personal budget and publicly available information about the target.

Class II: Knowledgeable Insider. Individuals with in-depth technical knowledge of the target. Often (former) employees of the organization producing the target. They have partial or complete access to sophisticated and often

target-specific tools and documentation. Nevertheless they face temporal and personal budget constraints.

Class III: Funded Organization. It is difficult to upper-bound the threat of these adversaries, as they can range from privately funded syndicates to nation state agencies. They employ many Class II attackers and supply them with state of the art professional tools. They aim to exploit existing and to craft new vulnerabilities for their goals.

Furthermore, physical attacks can be broadly classified based on the *activeness* and *invasiveness*. Of course, in practice there exists a spectrum of attacks that borrow and combine methods from different classes of attacks. Such attacks are referred to as *hybrid* or *combined* attacks. Standards such as the NIST 140-3 [1] address the multifaceted problem that is security of cryptographic modules. Considerations such as tamper evidence, resistance and response are made to minimize the danger of physical attacks.

Activeness. An *active* attack consists of two steps: perturb and conclude. Naturally, fault injection attacks belong to this category. Active attacks often require more costly equipment to carry out precisely. Also, a higher level of expertize if usually required as injecting faults, *i.e.* perturbing the device operation, can be destructive. A non-active, or *passive* attack consists of two steps as well: measure and infer. Naturally, side-channel attacks belong to this category. The measured physical emanation data are processed and statistical inferences are made on said data to extract the key. The price of side-channel attack setups and the level of expertize varies greatly between targets. Targets running at high clock frequencies, at the order of GHz, require very sensitive, expensive, equipment and knowledgeable handling to attack.

Invasiveness. A *non-invasive* attack requires no mechanical modification of the target devices, including its casing, circuit board and packaging. They are entirely carried out using existing interfaces. As such their power is somewhat limited, but they are virtually impossible to detect. They typically require low-cost equipment, but may require high level of expertize to mount. Therefore, non-invasive attacks can be carried out by all three classes of physical attackers. A *semi-invasive* attack allows attackers to open device casings and to remove them from their circuit boards. Thereby, semi-invasive attackers may gain access to additional interfaces, *e.g.* probing onboard busses between the Central Processing Unit (CPU) and onboard storage. They can also remove onboard decoupling capacitors to obtain better side-channel measurements, or tamper with the board power supply and external clock to inject faults. Note that, here

we assume that the target of an attack is a wholesome electronic contraption, device or product. Should the target be defined as the chip itself, the latter two examples would be considered non-invasive. A semi-invasive attack gives attackers a wider attack surface, but demand an increase in the quality of setups and level of expertize. Nevertheless, it is difficult to cover the evidence of physical tampering especially because some of the modifications introduced can be destructive. Lastly, fully *invasive* attacks give the attacker complete power over every aspect of the device. In addition to the advantages of semi-invasive attackers, invasive attackers have access to the insides of the target chip packaging. This allows them to probe CPU busses and caches, or even to make physical modifications to the chip metal layers using Focused Ion Beam (FIB) cannons. The price of the equipment and the interdisciplinary knowledge required to carry out these attacks make them exclusively available to Class III attackers.

2.3 Side-Channel Analysis

Unlike conventional cryptanalysis techniques that stem from mathematics, SCA leverages information that leaks through inherent physical channels. These physical magnitudes carry within information about the values and operations internally processed by a circuit, including cryptographic keys. SCA attacks are passive attacks, while they can be either non-invasive or semi-invasive. A degree of invasiveness can help the measurement acquisition and is not an issue as adversaries are often legitimate owners of the target devices. Unprotected implementations can trivially be attacked using simple setups consisting of a shunt resistor and a low-end oscilloscope. Therefore, SCA is recognized as a major threat to cryptographic implementations. The most prominent exploitable physical side channels include execution timing [81], power consumption [82], and electromagnetic emissions [52, 119]. More exotic side-channels include acoustic [53] and photonic [131] emanations. More recent hybrid schemes such as *actively triggered passive* Fault Sensitivity Analysis (FSA) [88] allow attackers to constitute new side-channels by injecting faults into the device. So far mentioned side-channels can be used to attack both hardware and software implementation of cryptographic algorithms. Powerful novel side-channels [89, 80] emerged, tailored to attack high-end software platforms that rely on cache memories and speculative execution. Nevertheless, the principles of SCA techniques are invariant of the underlying physical channel that leaks the information. They leverage the correlation of the recorded physical magnitude with the performed operations and intermediate data being processed. We focus on the power consumption, as the most widely used side channel. To be precise, we deal with the *instantaneous* power consumption waveform, recorded during

the target device operation. Such waveform corresponding to a single execution of the target algorithm (*e.g.* AES) is called a *power trace* or simply a *trace*. At each point of the trace, the instantaneous power consumption can be modeled as a linear combination of three components as shown in Equation (2.1):

$$P_{total} = P_{dynamic} + P_{static} + P_{noise} \ . \tag{2.1}$$

Here, $P_{dynamic}$ represents the power consumed during state transitions, *i.e.* when the logic gates are toggling. In the older technology nodes $P_{dynamic}$ accounted for the majority of the power consumed. In comparison, the P_{static} consumed when the gates are in the steady state was negligible. Thus it was often considered constant [91] and of no use for the SCA attacker. However, with the technology scaling below 100 nm, static power consumption becomes significant and dependent on the circuit state. Moradi [105] demonstrated the practical implications of leakage currents to SCA security on Field Programmable Gate Arrays (FPGAs), followed by the work of Pozo *et al.* [118]. In the more recent works, Moos *et al.* [103, 104] demonstrate a SCA attack using P_{static}. In the context of SCA on software targets, data-dependent $P_{dynamic}$ and P_{static} (on newer platforms) are broken down as $P_{dynamic} + P_{static} = P_{operation} + P_{data}$, as each operation is performed by a different piece of hardware in the arithmetic-logic unit [91].

The noise component P_{noise} represents the inevitable noise introduced by the target device and the measurement setup. Independent electrical noise sources include thermal noise, random telegraph noise and flicker noise. Thermal noise is a white noise that follows a nearly Gaussian distribution and is the predominant noise source of the three. Additionally, algorithmic noise introduced by the transitions of the circuit state follows a binomial distribution. Assume an n-bit state of the cryptographic implementation, where each bit flips with probability $p \approx 0.5$. Probability $P(k)$ of k bits flipping, is given in Equation (2.2).

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k} \approx \binom{n}{k} p^n \ . \tag{2.2}$$

For $n = 128$ we compare the binomial distribution $\mathcal{B}(n = 128, p = 0.5)$ with a normal distribution $\mathcal{N}(\sigma^2 = 32, \mu = 64)$ in Figure 2.4. The two distributions are nearly identical. Even if this were not the case, Lyapunov's variant of the central limit theorem [49] states that adding independent distributions, even if they are different, results in the normal distribution. Mangard *et al.* [91] experimentally show that the measured P_{noise}, *i.e.* with all noise sources added together, indeed follows a normal distribution.

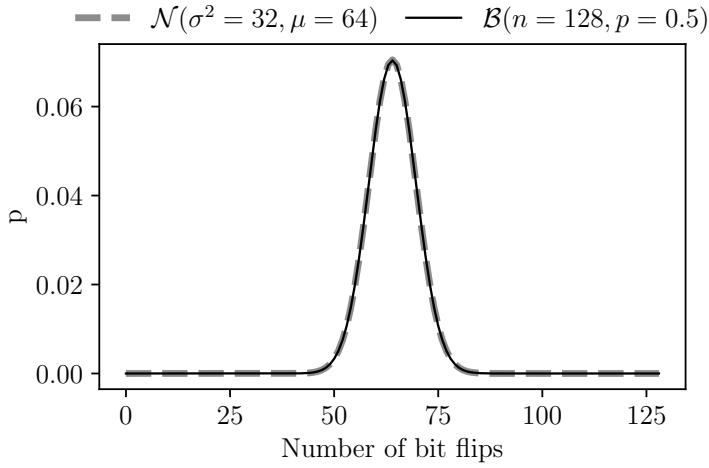


Figure 2.4: The distribution of state transitions.

2.3.1 Attack Techniques

The seminal Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [82] attacks were soon followed by techniques such as Correlation Power Analysis (CPA) [27], and Mutual Information Analysis (MIA) [57]. These attacks have been used to break security features of commercial devices [43, 115, 8, 128]. The starting point of each attack is trace acquisition. The principal components of a trace acquisition setup are presented in Figure 2.5 [91]. Additional components such as filters and amplifiers can be used to enhance the quality of acquired traces.

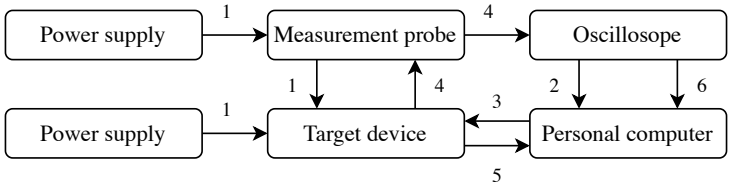


Figure 2.5: Typical side-channel measurement setup [91]. The numbers indicate the order in which components interact to obtain a trace.

Attackers start by supplying a stable power supply and clock signal to the target device, preparing it it for operation and trace acquisition (1). Next, attackers configure and arm the oscilloscope (2) via a personal computer, followed by issuing the start command to the target device (3). The measuring

probe records the instantaneous power consumption waveform and transfers it to the oscilloscope (4). Once the personal computer receives the output of the cryptographic operation (5) it reads the recorded power trace from the oscilloscope (6). For each new trace steps 2–6 need to be repeated. Traces can be accumulated, or processed on-the-fly, adhering to one of the SCA techniques.

2.3.2 Countermeasures

Side-channel countermeasures aim to make traces independent of the execution of the underlying cryptographic algorithms. They can be classified based on two active principles, namely *masking* and *hiding*. Both types of countermeasures aim to reduce the Signal to Noise Ratio (SNR) for the SCA attackers. Figure 2.6 [91] depicts this classification as:

- masking countermeasure aim to make traces independent of the intermediate values of the cryptographic algorithm—they introduce randomized algorithmic noise, $P_{rnd} = P^{rnd}_{dynamic} + P^{rnd}_{static}$; while
- hiding countermeasures aim to make traces independent of the processing of intermediate values—they reduce the data-dependent variations in $P_{dynamic} + P_{static}$ or increase the P_{noise}.

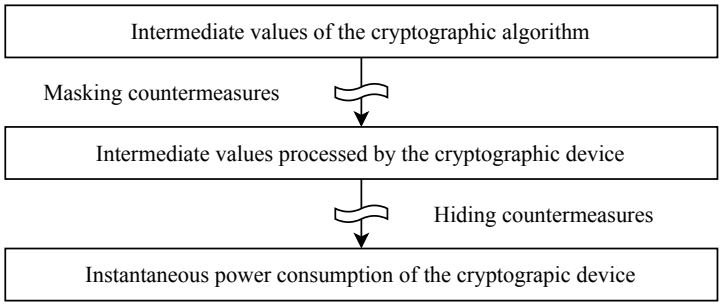


Figure 2.6: Classification of side-channel countermeasures [91].

Side-channel countermeasures can be applied to both hardware and software implementations. In this dissertation we focus on the protection of ASIC hardware implementations, leaving any software-related discussions out of scope. Countermeasures can be deployed across different levels of abstraction. For example, at system level, the number of key uses can be limited before rekeying. This limits the amount of traces attackers can obtain for a single key, at the cost of a more elaborate key management scheme or terminating

the device operation. Alternatively, key regeneration can be a feature of the system [42]. In both cases, it is difficult to make guarantees without making assumptions about the devices and protocols in the system. On the other hand, low-level countermeasures tackle the leaky physical channels directly, by adding dedicated physical structures. For example, a pair of capacitors and a diode bridge[5] can decouple computation from the external power supply [134]. These countermeasures may incur resource overhead, as well as overheads in design time and level of expertize needed. Other countermeasures can include causing trace misalignment using jittery clock or dedicated noise sources. Using such low-level functionalities to secure digital logic would require a more vertically integrated design flow to implement and evaluate SCA security reliably. Consequently, a series of different mid-level countermeasures are popular in the literature. They abstract the low-level, physical, behavior of the cryptographic devices while staying in the domain of digital design. This approach resonates with the well-established practices of the standard-cell ASIC design flow. We discuss some of the popular mid-level countermeasures below.

Hiding countermeasures

The correlation between the instantaneous power consumption of the CMOS logic style and the processed data makes it highly vulnerable to SCA. Several novel logic styles have emerged, with the idea of securing cryptographic implementations by making the instantaneous power consumption constant. As the underlying cryptographic algorithm remains unchanged, secure logic styles typically incur no latency penalty. At first authors of *secure logic styles*, such as Sense Amplifier Based Logic (SABL) [141], resorted to designing custom logic cells. This approach gained limited traction as manufacturability of custom cells would impede the mass adoption. Secure logic styles based on standard-cell libraries, such as improved Masked Dual-rail Precharge Logic (iMDPL) [116], LUT-based Masked Dual-rail Precharge Logic (LMDPL) [84] and Wave Dynamic Differential Logic (WDDL) [142] alleviate the manufacturability issue.

Wave Dynamic Differential Logic (WDDL)

Introduced by Tiri and Verbauwhede [142], WDDL is a differential logic style based on standard CMOS cells. Differential rails host complementary logic according to De Morgan's law, as per Equation (2.3):

$$c = a \cdot b \iff \overline{c} = \overline{a} + \overline{b} \ . \tag{2.3}$$

[5]Designing power electronics circuitry differs greatly from digital design.

For example, Figure 2.7 depicts two WDDL gates. Assuming an all-zero initial value, *i.e.* $x_0 = \overline{x_0} = y_0 = \overline{y_0} = 0$, such constructions ensure two properties:

- each logic gate in a WDDL gate toggles state at most once,
- the total number of toggles in a WDDL gate is independent of the input values.

Thus, WDDL logic style ensures a constant power consumption of the combinatorial logic, independent of the processed data.

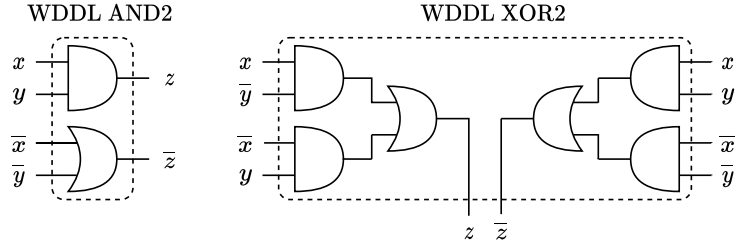


Figure 2.7: WDDL AND2 gate (left) and XOR2 gate (right).

Side-channel security of WDDL gates is predicated on a reliable way of setting all combinatorial inputs to zero, *i.e.* $x_0 = \overline{x_0} = y_0 = \overline{y_0} = 0$ in our example. Figure 2.8 depicts control signals, logic and registers required to do so.

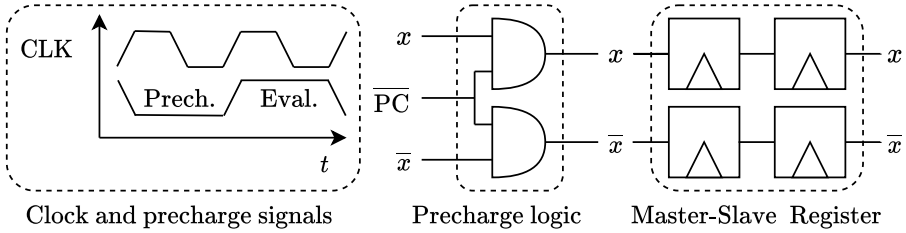


Figure 2.8: WDDL precharge generation and the dynamic dual rail flip-flops.

WDDL logic operates in two phases, namely precharge and evaluation. If a two-cycle master-slave register is used precharge signal $\overline{\text{PC}}$ needs to be applied only to the inputs to the WDDL-secured core. This will in turn create a wave of all-zero values and data values that propagate through all register stages; hence the name. Alternatively, a regular register layer could be used with precharge generation logic after each register layer. However this results in a global $\overline{\text{PC}}$ signal drawing a significant peak current that may harm the signal integrity. Therefore we favor the approach depicted in Figure 2.8.

In addition to the SCA resistance, WDDL provides redundant data coding scheme, presented in Table 2.2. Therefore, it allows fault-detection at no additional cost.

Table 2.2: WDDL redundant encoding.

x	$\overline{x}$	Encoded value
0	0	Precharge
0	1	Logic 0
1	0	Logic 1
1	1	Invalid/Alarm

WDDL is fully compliant with the standard-cell ASIC design flow, including custom routing techniques needed to balance the differential rails [145]. Technology node scaling however, increases the impacts of the manufacturing process variation. Thus the symmetry, required for the security of such differential styles, is upset. The manufacturability of secure logic styles is again hindered, leaving it up to digital designers to ensure security against SCA. Fortunately, recent advances in secure logic styles design enable easily manufacturable low-latency secure implementations [129].

Masking countermeasures.

Masking was introduced by Chari *et al.* [28] and Goubin *et al.* [60], as an algorithmic countermeasure against SCA. It is based on randomization of the intermediary values of sensitive variables processed during cryptographic computation. By doing so, masking schemes decorrelate the processing of sensitive variables from physical emanations. Masking schemes target security against SCA at the algorithmic level, abstracting physical properties of the underlying hardware with a number of assumptions. This makes masking schemes suitable for implementation on a variety of platforms, including standard-cell ASIC design flow.

Early proposed schemes such as the masked AND gate of Trichina [146] and the masking scheme of Ishai, Sahai and Wagner (ISW) [68] made succinct sets of assumptions about the underlying hardware. Said assumptions did not capture the finite propagation delays of digital circuits, thus failing to account for *glitches*. A *glitch* is a hazardous logic transition that occurs due to finite propagation delays in standard CMOS circuits. We say that a digital logic gate *glitches* if its output switches more than once per clock cycle. As these additional transitions are data-dependent they leak information about the

sensitive intermediaries of the cryptographic computation, thus breaking the SCA security.

Threshold Implementations (TI) introduced by Nikova *et al.* [112, 113] account for the impact of glitches. Follow up research gave rise to a myriad of boolean masking schemes, improving on efficiency and security [16, 35, 17, 18, 4, 26]. Other notable Boolean masking schemes include Domain Oriented Masking (DOM) by Gross *et al.* [61] and Consolidated Masking Schemes (CMS) by Reparaz *et al.* [123]. The provable security of masking schemes is further supported by theoretical advances [123, 36] that consolidate and unify masking schemes under theoretical frameworks. Although assumptions vary slightly among different schemes, most of them are easily attainable at the structural level of abstraction that dominates the standard-cell ASIC design flow. The latter two properties of masking schemes are driving their prevalence in the research field of SCA countermeasures.

Masking security models and d^{th} order security

Several security models, sub-models of the gray-box model, are used to formalize the relation between the adversarial power and the level of SCA resistance [68, 9, 48, 125]. Such models allow a mathematical framework for reasoning about the SCA security of digital hardware. Without going into the details of different SCA security models, we rely on the notion of d^{th}-order SCA security as originally introduced by Chari *et al.* [28]. In practice, an attacker targeting a masked implementation always tries to uncover the unshared values of sensitive variables. To do so, the attacker makes d observations of the preferred side channel. Said observations can be of the same point in the trace (univariate attacks) or a set of different points of the trace can be targeted (multivariate attacks). In either case, if d observations do not suffice for an attacker to retrieve the unshared value, we say that the implementation is d^{th} order secure. Implementations crafted for d^{th}-order security can always be broken in the next, $(d+1)^{th}$-order. Nevertheless, each security order increase rises the level of randomized algorithmic noise introduced in the measurements. As the law of large numbers dictates, the number of measurements (traces) needed to suppress a noise level grows exponentially with the increase of the noise level [91]. In other words, attacking in the next order requires collecting an exponentially larger number of traces. Therefore, d^{th} order security formalized using different models [68, 9, 48, 125, 36] is consequential for achieving practical security. In practice, achieving d^{th} order security for an m-bit sensitive variable $x \in \{0,1\}^m$ requires sharing x into $n \geqslant d+1$ shares. The fundamental assumption of all masking schemes is that shares operate, thus leak information, independently.

Henceforth, we focus on Boolean masking schemes. Algorithm 2 describes the initial sharing procedure using Boolean masking, $\oplus$ denotes the XOR operation.

Algorithm 2 Sharing $x \in \{0,1\}^m$ into n shares using Boolean masking.

Input: variable $x \in \{0,1\}^m$, number of shares n
Output: n shares $x_i \in \{0,1\}^m$, where $1 \leqslant i \leqslant n$ and $x = \bigoplus_{i=1}^{n} x_i$

1: **for** $i = 1$ **to** $n-1$ **do**
2: $\quad r_i \leftarrow^{\$} \{0,1\}^m$
3: $\quad x_i \leftarrow x \oplus r_i$
4: **end for**
5: $x_n \leftarrow x$

Threshold implementations.

Given a Boolean vector function $f : x \mapsto y$, where $x \in \{0,1\}^{m_i}$ and $y \in \{0,1\}^{m_o}$, such that $y = f(x)$, the computation of $f(x)$ can be shared into n shares $f_1(x_1), f_2(x_2), \ldots f_n(x_n)$, where $x_1, x_2, \ldots x_n$ are obtained using Algorithm 2. Such sharing is a TI [112, 113] if it fulfills three properties: *correctness*, *uniformity* and *non-completeness* [112]. TI schemes are provably secure under the share independence assumption. For a TI to be correct Equation (2.4) must hold. In other words, for each unshared input x, unsharing the shared outputs y_i must give the desired output y.

$$y = \bigoplus_{i} y_i = \bigoplus_{I} f_i(x_i) = f(x) \ . \tag{2.4}$$

For a TI to be uniform Equation (2.5) must hold. In other words, all values of shared inputs x_i and shared outputs y_i must be equiprobable.

$$\begin{cases} \forall a \in \{0,1\}^{m_i}, \forall x_i, Pr(x_i = a) = \dfrac{1}{2^{m_i}} \ , \\ \\ \forall b \in \{0,1\}^{m_o}, \forall y_i, Pr(y_i = b) = \dfrac{1}{2^{m_o}} \ . \end{cases} \tag{2.5}$$

The former two properties are common to all Boolean masking schemes. They impose relations between the input and output values of sharing, without any concern of the internal structure of each share f_i. TI introduces non-completeness, more precisely d-th order non-completeness, to handle the unwanted effects caused by hardware artifacts such as glitches. It states

that each combination of up to d shares f_i must be independent of at least one input share x_i. In other words, when observing any d shares, an attacker can never retrieve all information about the shared secret x. Therefore, as long as the fundamental assumption of share independence holds, implementation details of individual shares can be fully abstracted thanks to non-completeness. A generic example of a TI scheme for $n = 3$ is depicted in Figure 2.9.

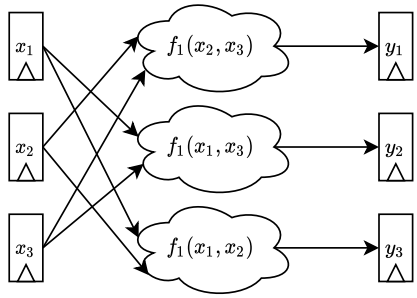


Figure 2.9: Generic structure of a three-share TI scheme.

Masking linear operations is trivial, as it amounts to the initial sharing Algorithm 2. Masking non-linear operations quickly becomes complicated. Figures 2.10 and 2.11 depict sharing of the two-input AND gate $z = a \cdot b$, using first-order-secure TI sharing by Bilgin *et al.* [19] and a first-order-secure sharing by Faust *et al.* [48], respectively. The numbers on each of the gates denote belonging to a particular share.

The circuit in Figure 2.10 splits the inputs a and b into three shares, a_1, a_2, a_3 and b_1, b_2, b_3, respectively. It computes the shared outputs z_1, z_2 and z_3 in a single clock cycle. The circuit in Figure 2.11 splits the inputs a and b into two shares, a_1, a_2 and b_1, b_2, respectively. It is significantly smaller than the previous one. However, it requires two register layers to compute the shared outputs z_1, z_2. Both circuits require one bit of randomness r per evaluation to satisfy the uniformity property. Clearly, a smaller area and a shorter critical path can be gained at the price of the increased clock cycle latency. In general, uniform random bits r can be added to satisfy uniformity, and flip-flops can be added to stop the propagation of signals from other shares that would break non-completeness. The morale is that many tradeoffs can be made, and a thorough body of research exists addressing them [16, 35, 17, 18, 4, 26, 48, 19].

Therefore, TI—as well as the other (Boolean) masking schemes—are a valuable technique for securing standard-cell ASIC hardware devices.

Interestingly, the circuit in Figure 2.11 from [48] is equivalent to the DOM-*indep* multiplier by Gross *et al.* [61] with the addition of the two output registers

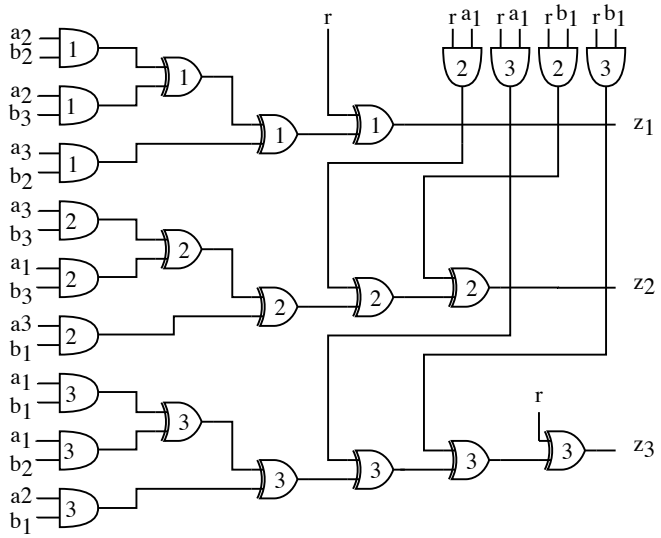


Figure 2.10: First-order secure, 3-share TI multiplication by [19].

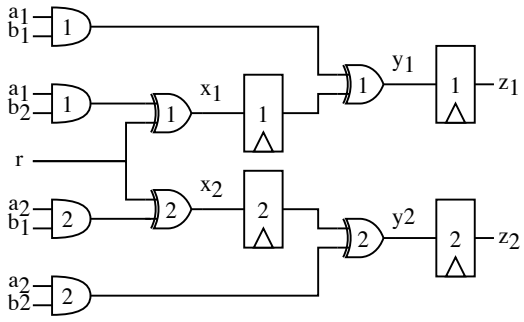


Figure 2.11: First-order secure, 2-share TI-like multiplication by [48].

needed to ensure the composable non-completeness. While TI presents a "top-down" approach to masking, dealing with high-level Boolean vector functions as a whole (*e.g.* S-Boxes), DOM embraces the non-completeness property and deploys it at the gate-level, calling shares "domains".

A closer look at first-order security.

Circuits in Figure 2.10 and Figure 2.11 are first-order secure masked circuits. Therefore they can be freely composed without negative security implications

nor additional assumptions. The former is purely combinatorial, hence as long as the shares are separated its first-order security is invariant to logic synthesis. However, the register placement is crucial for the first-order security of the latter. We use the timing diagram in Figure 2.12 to illustrate this.

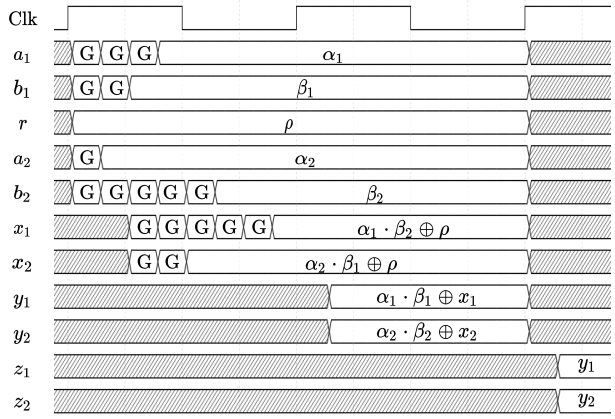


Figure 2.12: Signal waveforms, 2-share TI-like multiplication by [48].

The inputs can be driven either by flip-flops or combinatorial gates, *e.g.* trivially shared XOR gate. Consequently, glitches may cause the input bits to glitch; denoted using the label "G".

Firstly, without the first flip-flop layer storing x_1 and x_2 the circuit would be equivalent to the Trichina AND gate [146], secure only in the zero-delay model. The signal $y_1 = a_1 \cdot b_1 \oplus a_1 \cdot b_2 \oplus r$ would reveal information about the unshared b, despite the masking. Once the $r = \rho$ settles, information about the unshared input b may leak through multiple glitches. Once x_1 is registered, $r = \rho$ bit masks the cross-share product term b_2 and thus protects the compression in the second clock cycle. Analogously, without the x_2-register y_2 would leak a.

Secondly, as mentioned, without the second flip-flop layer the circuit from Figure 2.11 reduces to the DOM-*indep* multiplier [61]. The DOM-*indep* multiplier assumes that all input bits are shared independently. Any dependence of $a_{1,2}$ and $b_{1,2}$ would directly affect $y_{1,2}$, propagating the bias along a purely combinatorial path. Consequently, y_1 and y_2 needs to be registered to prevent the composability issues, as pointed out by Faust *et al.* [48].

Therefore, multi-cycle masked designs must be synthesized with great care. The separation of purely combinatorial shares is trivial to achieve. However, optimization algorithms, such as retiming [85] can relocate or entirely remove register layers in order to improve performance. Such algorithms are at the

heart of commercial EDA tools. Consequently, SCA evaluation must be performed after every quality-of-results-improving design step. Despite such design difficulties, to the best of our knowledge the first-order secure Boolean masking schemes remain secure when implemented (in academia) on the real hardware. However, designing and implementing them is a laborious and time-consuming process that requires high level of expertize. Additionally, recent work of De Cnudde *et al.* [30] raises concerns. Namely, authors show that the level of side-channel resistance of the same gate-level design varies depending on the physical placement and routing. This implies the existence of potentially harmful physical effects overlooked by the mathematical masking theory, in particular the independent leakage assumption.

2.3.3 Side-Channel Security Metrics

Attack-based evaluation.

Initially, attack techniques were used to determine the quality of a side-channel resistant implementation. Batteries of known attacks would be launched against the target implementation. Prominent attacks include DPA, CPA and MIA. The Measurements-to-Disclosure (MtD), or the number of traces before the key is revealed, was the prevalent evaluation method. Nevertheless, such testing suffers from several pitfalls. Firstly, such tests are very time consuming as numerous attack techniques must be considered and each requires a high number of traces for a reasonably protected implementation. Secondly, as attack techniques aim to extract the cryptographic keys from the wholesome implementations, it is difficult to facilitate such tests in the early design stages. And thirdly, known attacks typically depend on a specific leakage model. The MIA is an exception, being an information-theoretic method, but it is still computationally demanding. Therefore, using MtD as a security metric is bound to the leakage models and specific attack techniques used. Should novel attacks or leakage models be introduced, MtD obtained through attack-based testing can yield no security guarantees against the attackers' improved arsenal.

Leakage-detection based evaluation.

An attractive alternative to attack-based evaluations was introduced by Goodwill *et al.* [59]. In addition to naming said approach Test Vector Leakage Assessment (TVLA), Cooper *et al.* [32] discuss numerous practical details, inducing the approach a *de facto* standard for SCA testing. TVLA has since been addressed from the computational [132], performance [124] and practical

"know-how" [136] aspects. It is a generic statistical method based on the Welch's two-tailed t-test. Given two sets of cardinality N_1, N_2, with mean values μ_1, μ_2, standard deviations σ_1, σ_2, the first-order t-statistic is computed as per Equation (2.6):

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{(\sigma_1)^2}{N_1} + \frac{(\sigma_2)^2}{N_2}}} \ . \tag{2.6}$$

The first-order t-statistic quantifies the probability of distinguishing the means of the two sets. Therefore, $|t|$ values larger then a certain threshold are associated with a level of confidence that the two sets are distinguishable. In practice, values $|t| > 4.5$ correspond to the two sets being distinguishable with probability $p > 0.99999$, *i.e.* over 99.999%.

The t-statistic in Equation (2.6) is the first-order t-statistic for the mean is the first-order statistical moment. Schneider and Moradi [132] delineate how to compute higher-order t-test statistics. Notable higher-order moments t-statistics are: standard deviation (2^{nd}-order), skewness (3^{rd}-order) and kurtosis (4^{th}-order).

The TVLA procedure can be summarized in the following steps:

1. Choose a sensitive m-bit variable $x \in \{0,1\}^m$ and decide on a partitioning function $\mathcal{P} : \{0,1\}^m \mapsto \{0,1\}$.
2. Generate two sets of data inputs $\mathbb{D}_1$ and $\mathbb{D}_2$, such that $|\mathbb{D}_i| = N/2$ and $\forall d \in \mathbb{D}_i, \mathcal{P}(d) = i$, where $i \in \{0,1\}$.
3. Randomly interleave data inputs from $\mathbb{D}_1$ and $\mathbb{D}_2$, feed them to the target device and record n-sample long traces.
4. Partition traces into two sets $\mathbb{T}_1$ and $\mathbb{T}_2$, corresponding to data inputs from $\mathbb{D}_1$ and $\mathbb{D}_2$, respectively.
5. Compute the t-statistic, attempting to distinguish the two sets of traces $\mathbb{T}_1$ and $\mathbb{T}_2$.

For a large N, storing traces may require hundreds of gigabytes of storage space. To avoid storage issues, steps 3–5 can be performed on-the-fly, as explained in [132]. In addition to the efficient computation strategies put forward by Reparaz *et al.* [124], leakage-detection requires fewer traces to detect the leakage than to exploit it [59, 32]. Therefore, leakage detection is inherently more efficient than the attack-based evaluations.

Depending on the partitioning $\mathcal{P}$ a *specific* or a *non-specific* TVLA can be performed. A specific TVLA is similar to attack techniques as it focuses on the value of a specific intermediate. Inputs need to be crafted for each partitioning of the target variable. For example testing an 8-bit AES S-Box requires 256 specific partitioning schemes $\mathcal{P}$, one for each 8-bit value. Therefore it results in a large number of tests. An alternative is to perform the *non-specific* TVLA. The non-specific partitioning is also called the *fixed-versus-random test* , as one set of traces is generated using a single value of the partitioning variable, while the other is based on all other, randomly selected values. The non-specific tests converge more slowly, but always provide full coverage [122].

Lastly, depending on the manner in which the samples for the t-statistic are selected *univariate* or *multivariate* TVLA can be performed. The univariate TVLA assumes that the t-statistic is computed for each sample independently. Alternatively, the sample points of each trace can be combined to create "multivariate traces". Performing univariate TVLA on thus obtained traces is called multivariate TVLA.

Now let us briefly discuss the relation between the statistical moment order, the d-th order security and the number of physical probes. Assuming a non-invasive attacker, it is likely that only one physical probe measuring the supply current is used. However, given enough measurements each statistical moment of the distribution can be considered a separate observation of the side-channel, *i.e.* another probe. Consequently, higher-order TVLA is routinely used to evaluate the higher-order security of masked implementations(*e.g.* Bilgin *et al.* [16]).

More recently, Moradi *et al.* [108] introduce another statistical test for side-channel leakage detection based on the χ^2-statistic. Unlike the t-statistic that observes each statistical moment as if it were independent, the χ^2-statistic tests the independence of the two distributions including all the moments combined. Consequently, the χ^2-statistic can detect leakage spread across multiple higher-order moments and in the low-noise environments more efficiently, *i.e.* with fewer traces. As such the χ^2-statistic tests are a valuable complement to the t-statistic-based testing, valuable for the third-, or higher-order evaluations.

As seen above, the TVLA evaluation is independent of leakage models, attack techniques and even of the target architecture, as it only determines whether two sets of traces have distinguishable statistical moments. TVLA can quickly detect flawed implementations, as the number of traces required to detect leakage is often orders of magnitude smaller than the number of traces required to exploit the leakage [32]. However, given the leakage detected using TVLA it is difficult to assert whether said leakage can be exploited in practice. Potential pitfalls of TVLA-based evaluations are delineated by Standaert [136], but if used properly TVLA remains the predominant evaluation method. Lastly, for

the TVLA desired negative results (no leakage) to be reasonably dependable a sufficient number of traces must be collected. Looking at the academic literature, dependable number of traces may range from several millions to hundreds of millions, or even a billion by Sasdrich *et al.* [129].

2.4 Conclusions

In this chapter we first introduced the standard-cell ASIC design flow that dominates the modern digital design. We explained its stages and the iterative manner in which they are traversed. Each design stage represents the target design in a different, increasingly complex, level of abstraction. From the manufacturability and timely deployment perspectives, standard-cell ASIC design flow is largely automated using, hence dependent on, EDA tools and standard-cell library models.

Next we introduce physical attacks, as a great threat to modern cryptographic implementations. We delineate how physical attackers are classified based on their actions and the amount of socioeconomic resources at their disposal. Out of the plethora of possible attacks, we focus on the passive, non-invasive, or semi-invasive side-channel attacks. Side-channel attacks deserve this focus as they present a high risk, being available to a wide-range of attackers using low-end equipment.

Furthermore, we described the practical principles of mounting side-channel attacks on existing devices, along with the countermeasures used to prevent them. We focus on countermeasures that can be deployed by a standard-cell ASIC designer during the implementation stages of the device lifecycle. Thus we introduce hiding countermeasures in form of secure logic styles compliant with standard-cell CMOS libraries, and masking countermeasures. Masking schemes claim provable side-channel security under rudimentary assumptions about the underlying hardware. These assumptions are compliant with the structural view of standard-cells and are therefore appealing for wide-spread adaptation—as long as the models for the underlying hardware remain valid. Lastly, we explain the state of the art techniques for assessing the level of side-channel security.

As it can be seen from this chapter, there exists a clearcut line between the areas of digital design and side-channel security. Seeing how cryptography and data security were mostly software concerns for the better part of digital EDA industry lifetime, this gap is not surprising. Nevertheless, the gap must be bridged in order to keep up with the omnipresent collage of digital devices that needs to be secured.

Chapter 3

SCA-Aware Standard-Cell ASIC Hardware Design Flow

А у руке Мандушића Вука,
свака пушка биће убојита!

Петар Петровић Његош

Content Source

This chapter is largely based on material published in:

D. Šijačić, J. Balasch, B. Yang, S. Ghosh and I. Verbauwhede, "Towards efficient and automated side-channel evaluations at design time", *PROOFS 2018*, Amsterdam, The Netherlands, September 13, 2018 (2018), L. Batina, U. Kühne, and N. Mentens, Eds., vol. 7 of Kalpa Publications in Computing, EasyChair, pp. 16–31.

D. Šijačić, J. Balasch, B. Yang, S. Ghosh and I. Verbauwhede, "Towards efficient and automated side-channel evaluations at design time", Journal of Cryptographic Engineering, Volume 10, Issue 3, Published June 22, 2020.

Contribution: Principal author.

In this chapter we present how the pre-silicon SCA evaluations can be incorporated directly on top of the standard-cell ASIC design flow. Figure 3.1 depicts how this SCA-aware flow corresponds to the traditional one. We keep the notation from Figure 2.1 for the design steps that we consider in this chapter. The remainder of the traditional steps is represented using grey color. More importantly, we use the red shade to denote where we introduce additional side-channel analytic steps in parallel to the logic simulation. Clearly the approach is fully compatible with the existing flow. We argue each of the introduced steps in great detail and experimentally verify the rationale.

In order to show the feasibility, and the practicality of this approach we design and implement Computer-Aided Side-Channel Analysis Design Environment (CASCADE). CASCADE is an efficient, flexible and easily extensible framework that adds the design-time side-channel evaluations vertically, spanning front-, and back-end stages of the standard-cell ASIC design flow. It is based on commercial EDA tools, providing ease of integration with the existing designer's toolboxes.

3.1 Motivation

As discussed in Chapter 1, SCA is acknowledged as a major threat to cryptographic implementations. In turn, a plethora of SCA countermeasures has emerged in the literature. Nevertheless, the implementation of countermeasures is a non-trivial task that requires a high level of expertize. As a result, in addition to the overhead in the required circuit complexity, power consumption and performance penalties, incorporating SCA countermeasures in a product incurs additional costs of highly specialized experts. Moreover, many countermeasures require the use of dedicated logic styles or special libraries, further increasing production costs and time-to-market. But worst of all, it is difficult to determine the efficacy of countermeasures prior to chip manufacturing. SCA vulnerabilities disclosed at post-silicon stages can cause major set backs, potentially requiring a complete redesign.

In this context, hardware simulations rise as an attractive alternative to evaluate the SCA security at design time. Simulation techniques for typical hardware design constraints are long-studied and well integrated into EDA tools. Models used by the EDA tools capture physical manifestations of the intended silicon, giving a simple and composable representation of the individual logic gates. As a result, they can provide remarkably accurate circuit complexity (*i.e.* area), delay, and power estimates even in the earliest design stages. The plentitude of physical information contained in the EDA models makes them potentially

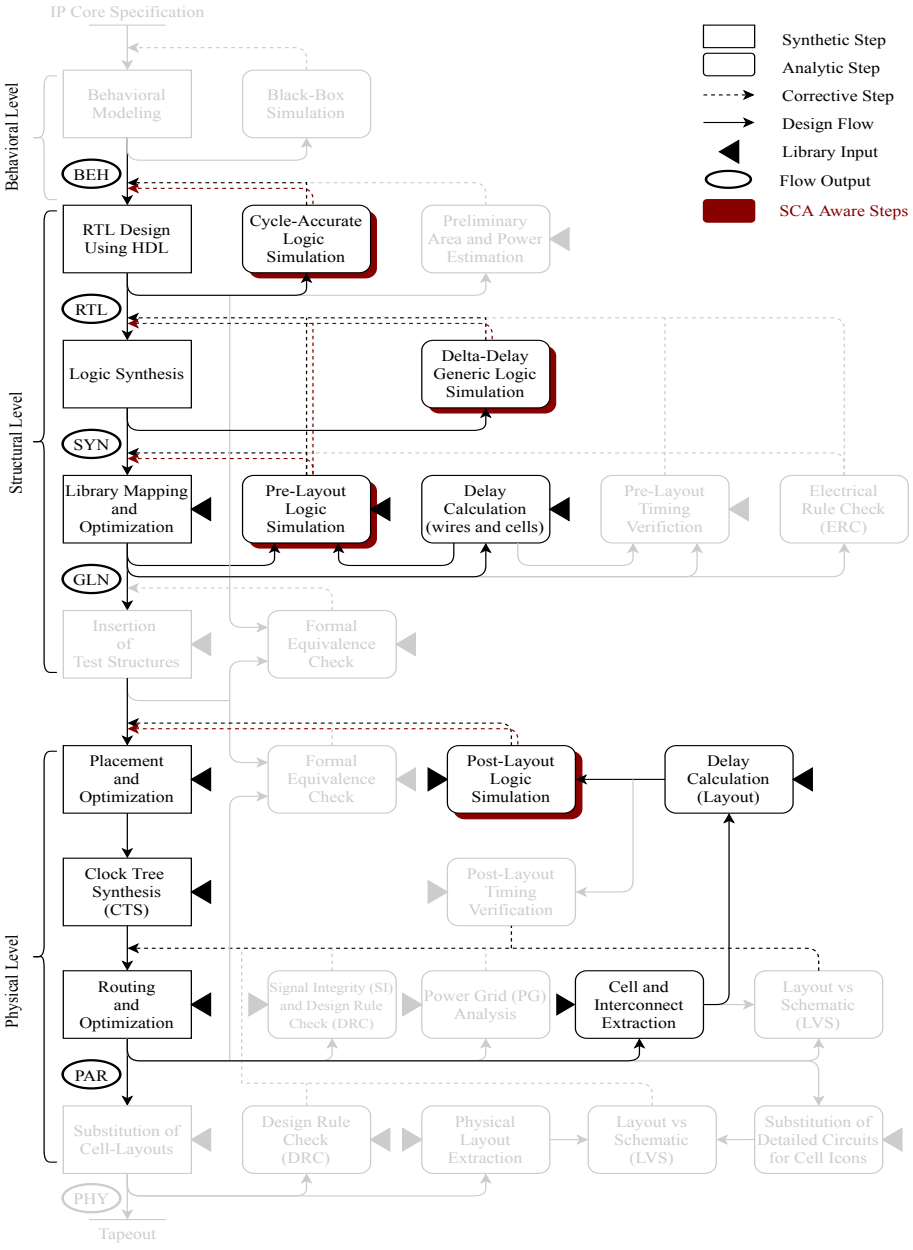


Figure 3.1: SCA Aware steps in the standard-cell ASIC design flow.

viable for capturing side-channel information leakage in the early pre-layout stages. Nevertheless, said models are devised to represent the behavior relevant for the designs to operate correctly within the given specification, not to be SCA secure. For example, digital designers predominantly deal with average and peak power consumption during device operation. The former dictates battery life and cooling requirements, while the latter determines the rating of the power supply and the power distribution network. On the contrary, SCA security depends on the instantaneous waveform of the physical side-channel emitted for each data input. Therefore, dedicated models for simulating SCA evaluation are lacking.

In addition to the elaborate and efficient models, standard-cell ASIC design flow dominates the market for its generality and efficiency. Namely, design stages are invariant of the intended design functionality. In turn, design stages can be largely automated and incorporated into the EDA tools that would provide feedback at each design stage. For a designer implementing a side-channel secure cryptographic core no such tools exist. Moreover, cryptographic implementations vary greatly depending on the properties of the selected countermeasure. Dedicated tools can be developed for checking properties of each countermeasure, increasing the complexity and dependency of the development stack. Instead, generic evaluation tools would be more inline with the standard-cell design flow. Furthermore, unlike the monotonous operating cycle of non-cryptographic cores, side-channel secure cryptographic implementations need to withstand a variety of known and unknown attacks. Therefore, SCA evaluation methods embedded in the standard-cell ASIC design flow need to be generic and the evaluation needs to be automated to adhere to the flow.

Last but not least, there exists a rich body of work in the design of countermeasures based on standard-cell libraries. Such countermeasures, with the Boolean masking schemes at the forefront, have matured in the last decade and we deem them ready for mass adoption in the colorful collage of the IoT devices and beyond. Therefore, we believe it is worth investigating how the existing models and EDA tools can be used in conjunction with the SCA evaluation methods to provide efficient and reliable pre-silicon evaluations of the side-channel security. Bridging the gap between the two areas would yield synergetic benefits to both the digital design practitioners and the communities designing countermeasures. Thus the former would be able to deploy more reliably secure devices in shorter time to market, while the latter would have better understanding how the theoretical assumptions bare the impacts of the physical world. Consequently, we find this merger to be the necessary condition for deployment of the side-channel secure devices en mass.

3.2 Related Work

The issue of using simulations in the context of the design-time SCA evaluation has been previously addressed in the literature. In fact, detailed analog Simulation Program with Integrated Circuit Emphasis (SPICE) simulations are predominantly used to evaluate the side-channel security of the secure logic styles. SPICE is invaluable for design and evaluation of the secure logic styles based on full-custom cells. Regazzoni *et al.* [121] introduce MOS Current Mode Logic and use SPICE simulations to design and evaluate a side-channel secure instruction set extensions protected using this secure logic style. Similarly, Tiri and Verbauwhede [141] design and evaluate Sense Amplifier Based Logic (SABL). Furthermore, Kamel *et al.* [71] thoroughly examine the practical implications of evaluating side-channel security using SPICE simulations. They target an AES S-Box protected using Dynamic and Differential Swing-Limited Logic [63]. Similarly, standard-cell-based secure logic style designers rely on SPICE simulation. Tiri and Verbauwhede [142, 143] introduce the Wave Dynamic Differential Logic (WDDL) and evaluate a protected AES core using SPICE. Bhasin et al. [13] do the same for a WDDL protected PRESENT engine. While invaluable for small-scale observations, the computational requirements of SPICE simulations grow exponentially as the transistor count increases. Taking into account large number of measurements needed for a proper SCA evaluation, simulating entire designs in SPICE can take weeks if not months. Instead, Kirschbaum and Popp [76] resort to logic simulations to evaluate an 8-bit controller in Masked Dual-rail Precharge Logic (MDPL). Logic simulations can provide quick but rough information leakage estimates at early stages, relying on simple Hamming Weight (HW) and Hamming Distance (HD) models. It is understood from these works that different trade-offs between the simulation accuracy versus time influence the security assessment. Quite naturally, these models become more accurate as the flow approaches the actual layout, *i.e.* as more details on the target circuit are available. Nevertheless in all these cases authors use but one method, at an arbitrarily chosen stage in the design flow.

Furthermore, dedicated tools and design stages were introduced for specific countermeasures. Tiri and Verbauwhede [144, 145] tailor extensions to the standard-cell design flow to allow balanced placement and routing necessary for achieving symmetry of WDDL differential rails. Arribas *et al.* [5, 6] develop tools dedicated to checking properties of Threshold Implementations (TI). More recently Knichel *et al.* [77] introduce a tool for verifying compliance to security notions for different Boolean masking schemes. The latter two approaches, while efficient, rely on the abstract mathematical models. Mangard *et al.* [92] show how leakage of masked Complementary Metal-Oxide Semiconductor (CMOS) gates can not be adequately captured using cycle-accurate simulations.

Moradi *et al.* [109] discuss power models for SCA evaluation, but do not discuss the application nor scalability. Fujimoto *et al.* [51] introduce a model, similar to the ones used to capture the influence for power supply and substrate noise, to speed-up SPICE simulation. Yet, such models are still no match for the efficiency of gate-level models. Nevertheless all these works address a single stage of the design flow and are often applicable to a single type of countermeasures.

More recently, several frameworks for pre-silicon evaluations have emerged. He *et al.* [64] create a framework with custom methods and tools to evaluate the SCA security at the register transfer level. Other frameworks such as [66] proposed by Huss *et al.* incorporate targeted insertions of ad hoc countermeasures in an attempt to mitigate the detected leakage. Frameworks go as far as evaluating software implementations on ARM Cortex-M0 controllers without the physical device present, Mc Cann *et al.* [94]. However, there are no complete approaches in compliance with the tools and practices of the standard-cell ASIC design flow.

The additional, complementary, approaches include information theoretic and formal methods, along with the FPGA prototyping. Macé *et al.* [90] consider information theoretic evaluations as a complementary method to simulation. The formal verification methods such as [12, 20] are emerging as valuable alternatives that do not require collections of measurements. Nevertheless, they operate on rather high levels of abstraction, overlooking many properties of hardware, and are closely tied to certain types of countermeasures.

FPGA prototyping is a very straightforward method, unparalleled in that it yields of actual silicon evaluations. Hence, evaluations include physical effects due to noise, thermal drift, measuring setup, etc. It is especially advantageous for masking techniques, as they claim independence from the implementation of individual shares. A limitation of this approach is however that FPGA implementations can only be computational equivalents of ASICs. The fundamentally different structure of the FPGA configurable logic blocks and the ASIC gates can therefore make such evaluations incomplete. Moreover, the specifics of the FPGA structure and the physical layout are proprietary to its vendor. This hinders the identification and fixing of the issues identified in a security analysis. An example is given by De Cnudde *et al.* [30], who investigate the impact of coupling effects on protected designs implemented on FPGA platforms.

3.3 Contributions

Although the topic of SCA evaluations based on simulations has been investigated in earlier works, to the best of our knowledge it has not yet been made an integral part of the design process. In this chapter we address this matter in a wholesome and methodical manner, spanning the entire design flow—from behavioral to layout stages. We tackle the practical aspects of the implementation and the evaluation of cryptographic circuits. We also provide performance and scalability figures to show the practical viability of this approach. Our goal is to enable a methodology that allows circuit designers to assess the security of their implementations at different stages, similar to what is currently done for other design targets, *e.g.* timing constraints. The contributions of this chapter are placed along the following lines:

1. We discuss the rationale behind the integration with the standard-cell ASIC design flow.
2. We dissect the physical gate-level models used by the EDA industry up close and examine how they can be used for SCA evaluation.
3. We apply our approach to a set of representative cryptographic circuits, with known SCA security properties in order to validate its functionality. We strengthen our arguments by demonstrating a flaw in a recently proposed masked design of an AES S-Box.
4. We design and implement a flexible and extensible framework to support practical design-time SCA evaluation. We build on decades of experience by relying on commercial EDA tools, instead of designing our own simulators. We enrich this set of tools with optimized parsers and analysis tools written in C. Our framework strings them together according to categorized sets of parameters, to allow a high degree of automation of design and SCA evaluation.
5. Lastly, we benchmark our approach and discuss the practical details, feasibility and the utility to the designers.

The rest of this chapter is organized as follows. In Section 3.4 we discuss the rationale behind the added design steps dedicated to the SCA. In Section 3.5 we present a framework that implements the approach from Section 3.4. We detail the design of tools and data formats necessary for tackling practical issues. In Section 3.6 we give experimental results of evaluating several representative cryptographic circuits, using different models. In Section 3.7 we discuss and benchmark our approach and its implementation embodied by our framework. Lastly, we present our conclusions in Section 3.8.

3.4 SCA-Aware Extensions to the Standard-Cell ASIC Design Flow

In our view, the practical viability of SCA evaluations at design time is bound by three aspects. Evaluations must:

1. be available as early in the design flow as possible,
2. be fast and scalable in terms of circuit sizes and
3. guarantee a reasonable level of confidence in the security of the end device.

We focus on the first two aspects in this chapter. Studying the latter key aspect requires dedicated tapeouts and comparisons against chip measurements for a number of different scenarios.

3.4.1 The Preferred Side-Channel

We focus on the instantaneous power consumption waveform as the target side-channel. More precisely, the waveform of the supply current drawn by the cryptographic core. As the power supply voltage is approximately constant, *i.e.* Equation (3.1), the two can be used interchangeably for this purpose.

$$P(t) = I(t) * V_{DD} \,. \tag{3.1}$$

There are multiple advantages to observing this side-channel at design time. Firstly, as CMOS circuits draw a significant dynamic current only when switching, data-driven logic simulations—present throughout the standard-cell ASIC design flow—can be used to represent it. Moreover, observing the supply current discloses the timing of the computation with a higher resolution than the clock cycle count. Secondly, the current is localized in the gates and the interconnects. By contrast, simulating the electromagnetic emissions requires spatial distribution of the circuit components and is more computationally demanding. Therefore, it can not be applied in the pre-layout stages. Thirdly, more exotic emanations such as acoustic [53] and photonic [131] are more dependent on the printed-circuit board and the manufacturing technology. As such they can not be applied at the early design stages.

Therefore, we choose the instantaneous power consumption as the preferred side-channel in the remainder of this chapter.

3.4.2 The Systematic Use of Simulations

We argue that the systematic use of simulations along the EDA flow can greatly decrease efforts of designers, while yielding more reliably secure designs prior to manufacturing. At design time it is easy to focus on the critical hardware blocks (*e.g.* S-Boxes), prior to evaluation of the entire designs. The absence of noise and the high simulation precision allows us intimate observation of the target circuit, unattainable using measuring equipment. Simulations also provide fully aligned traces, removing the need for the costly pre-processing. We can treat the effects of controllers, data path and all physical circuitry (*e.g.* clock buffers) using the same physical models and tools, without any additional manual input. Moreover, traditional design constraints, *i.e.* timing (setup and hold violations) and average power consumption, are already being evaluated in this manner.

In contrast, the models for side-channel security are completely lacking. It may be possible to use the existing models (for timing or power), but it is unclear whether they will give upper or lower bounds or indicate trends on SCA security. Therefore we investigate this in this chapter.

Furthermore, unlike the FPGA evaluations, we directly model the physical structure of the target ASIC circuit. In other words, logic implemented using FPGA Look-Up Tables (LUTs) has the same output as the one implemented using standard ASIC cells, but the transient behavior (*e.g.* glitches) varies. Compared to the inherently serial nature of data acquisition from a chip, simulating multiple power traces in parallel is trivial.

Lastly, we stress that models, as simplifications of physical phenomena, can never fully capture reality. Hence, simulations are only as accurate as the models they use, and they can not account for artifacts of the manufacturing process. Therefore, we do not propose design time evaluations as a replacement to post-silicon measurements, but as a design technique aimed at shortening the time-to-market and more reliably secure designs from the first tapeout.

3.4.3 SCA Evaluation Methods

Simulated side-channel traces obtained at each design stage need to be evaluated. Since we aim to quickly assess an arbitrary piece of design, we prefer generic methods, independent of the underlying countermeasure, over batteries of attacks. One possible approach is to use information-theoretic metrics such as the one proposed by Standaert *et al.* [137]. While certainly useful and possible

to integrate in our setting, estimating probability distributions may be too computationally and memory intensive.

Instead, we focus on the SCA evaluation by means of leakage detection methods described in Section 2.3.3. In particular we focus on the Test Vector Leakage Assessment (TVLA) methodology [59, 32], as we mostly evaluate the first- and second-order secure designs. However, χ^2-based testing can easily be added for the higher-order designs.

The evaluation of the designs in their final stages, *i.e.* an entire AES core as opposed to a single S-Box, can be additionally fortified by applying some of the attack techniques. Nevertheless, at multiple fast iterations of the leakage detection is beneficial for the debugging purposes.

3.4.4 Simulation Models

Analog SPICE simulation, albeit the pinnacle of electronic modeling in terms of accuracy, features exponential increases in run times with the transistor count increase. We do support SPICE in our framework, as they are useful as a reference for smaller validation circuits. For practical reasons we focus on timing and power models that have the potential to scale efficiently. In the remainder of this section we delineate the models used by the EDA and SCA communities. Note that for all of the models discussed in the remainder of this chapter, the approximation shown in Equation (3.1) holds exactly.

Models from the SCA community.

The secure logic style community predominantly relies on detailed SPICE simulations to evaluate their designs. In addition to computational inefficiency, Tiri and Verbauwhede [143] raise concerns regarding the effects different parasitic extraction methods in the backend stages of the design. We address the issues of parasitic components in Chapter 5.

On the contrary, the masking community relies on minimal assumptions when modeling the underlying hardware. Early works, such as [146], retain their security only in the zero-delay model, *i.e.* they can be broken due to the effects of glitches caused by the propagation delay in the standard CMOS circuitry.

Modern masking schemes prevail in the presence of glitches, *e.g.* due to the non-completeness property of threshold implementations. Splitting sensitive values in multiple shares and performing independent computations ensures that no glitch, in whichever share it may occur, leaks information about the

unshared secret values. Consequently, the modeling of the circuit timing under these assumptions is considered to be of less concern. This allows for the use of generic, library independent, Δ-delay models where each gate is assigned a fixed delay.

Leakage models often employed by the SCA community are based on the HW and the HD of the processed data. Both are based on the predominance of dynamic power consumption in the standard CMOS logic. The HD model maps every toggle with a Dirac-like pulse of unitary amplitude. Multiple toggles that happen at the exact same time are simply added together. The HW model maps the number of logic ones to the amplitude of the Dirac-like pulse, without considering previous states. It is particularly useful for evaluating software implementations, where periodically pre-charged buses are the main source of leakage.

Models from the EDA community.

Timing parameters determine performance constraints, *e.g.* setup and hold times. Hence, the models for the timing simulation (closure) are at the heart of the EDA tools. Standard-cell libraries contain detailed information on how to extract timing parameters for the Gate-Level Netlist (GLN) and the Place and Route (PAR) stages. In the GLN stage, interconnect delays are extracted from statistical wire load models embedded in the standard-cell libraries. In the PAR stage, delays are extracted from the actual physical layout. Therefore at the PAR stage the impact of concrete parasitic elements is taken into account. These are detailed models for timing and power consumption, in the Open-Source Liberty [22] format, compatible across the EDA vendors.

We focus on the Composite Current Source (CCS) models [67], as the "best in class" for technologies below 90 nm. Figure 3.2 depicts the entries of the four LUTs that characterize the output current of the `XOR2_X1` gate from NanGate 45 nm standard-cell library [78]. The gate has two input pins, A and B. Output current waveforms depend on the pin initiating the transition as well as its direction, *i.e.* rising or falling. As the underlying Boolean function is commutative, this asymmetry can not be detected without observing physical properties of the cell. As such, CCS models are an industry standard used for "golden" sign-off estimations.

Figure 3.3 depicts the CCS power consumption waveform of the `XOR_X1` gate for five transitions. Transitions are separated using vertical dotted lines. We use Synopsys PrimeTime (PT) with the PX add-on, and henceforth refer to these simulations as PTX.

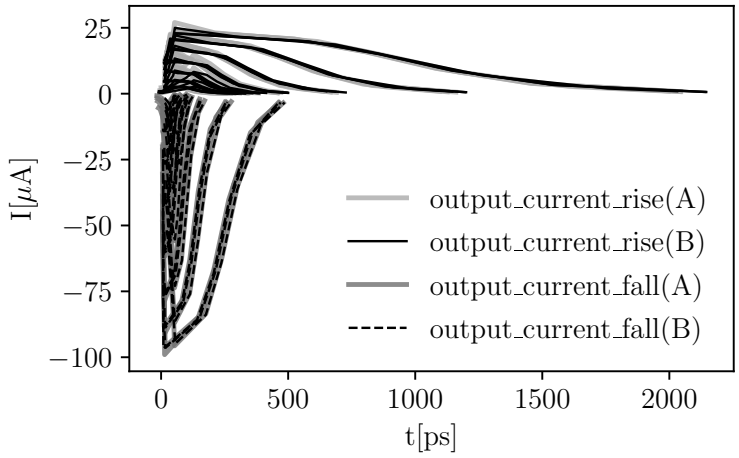


Figure 3.2: CCS output current waveforms of a `XOR_X1` gate.

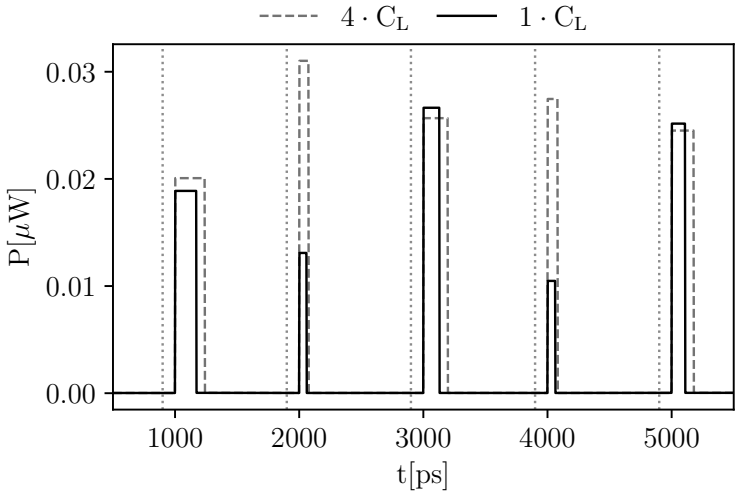


Figure 3.3: CCS power waveform of the `XOR2_X1` gate for five transitions.

In both cases `XOR_X1` is driven using `DFF_X1`. The solid line is obtained when the loading capacity of the output pin is set to the capacity of the `DFF_X1/D` pin C_L. The dashed line is obtained by increasing this capacity four times. When evaluating small isolated circuits using PTX it is important to set proper

design constraints. These representations contain more information than the unitary pulses normally used in the SCA community. However, the information is much less detailed than what CCS models can capture, Figure 3.2. The output waveforms are clearly designed with the average power consumption in mind.

Consolidating timing and power models.

We use the parametrized Δ-delay model to bridge the gap between SCA and EDA worlds. This model is useful to provide assessments before a specific library is introduced. Equation (3.2) states the general form of the parametrized Δ-delay model.

$$\Delta = \delta(1 + F\theta) \ . \tag{3.2}$$

Here, δ is the fixed propagation delay, F is the fanout and θ is the scaling factor. By choosing $\delta = 0$ the model is reduced to $\Delta = 0$, *i.e.* the zero-delay model. By choosing $\delta > 0$ and $\theta = 0$ the model is equivalent to the fixed $\Delta > 0$ delay model. Lastly, for $\delta > 0$ and $0.05 \leqslant \theta \leqslant 0.20$ we define the fanout dependent delay Δ_F. We choose the range for θ based on empirical observations of several modern libraries and use these values in our experiments.

We expand the HD model into the Marching Sticks Model (MSM), named for its graphical interpretation. MSM power for a standard-cell gate is described by Equation (3.3). MSM power of a standard-cell design is a sum of contributions of all gates. The parameter $-1.0 \leqslant \alpha \leqslant 1.0$ addresses the asymmetry of the rising and falling edges. When $\alpha = 0$, MSM power model is symmetrical and identical to the HD model. MSM evaluations need to be computed from logic simulations orthogonally to the underlying timing model.

$$\mathrm{P}_{\mathrm{MSM}} = \begin{cases} 1 \,, \text{output transition } 1 \to 0 \,, \\ 1 - \alpha \,, \text{output transition } 0 \to 1 \,, \\ 0 \,, \text{in steady state} \,. \end{cases} \tag{3.3}$$

We can relate MSM to CCS power models in the same manner as the Δ-delay models relate to the timing ones. The quality and performance of the SCA evaluation using MSM versus CCS power models at different design stages needs to be determined. Note that the MSM estimations are adjunct to the logical simulations. They are a precursor for the event-driven instantaneous

power consumption estimations using CCS power models. Hence, they are the common case in terms of performance and scalability.

Lastly, MSM model accounts only for the dynamic power consumption. As static power consumption poses a threat to sub 100 nm technologies [103], this must be addressed in the context of design-time evaluations. MSM is valid, albeit simple, power model for all standard CMOS gates always consume more power during transitions than in the steady state. Therefore, a strong correlation exists between the logic state transition and the dynamic power consumption. Static power consumption depends on the logic values too. Table 3.1 shows the static power consumption of three different cells from the NanGate 45 nm library [78], for the slow process corner. Clearly, little correlation exists between the input state and the static power consumption. It is much more dependent on the way the cell is designed internally. The correlation further decreases when more complex cells are considered. Therefore, we see no good way of making a generic, library-independent model for assessing the static power consumption. Luckily, as soon as a standard-cell library is selected, CCS models offer detailed information about the static power consumption.

Table 3.1: Static power consumption in nW, captured by CCS power models.

Input state	`AND2_X1`	`XOR2_X1`	`OR2_X1`
00	10.17	19.55	20.78
01	17.20	33.33	15.92
10	12.42	24.31	16.59
11	20.82	22.54	17.64

3.4.5 Simulation Methodology

Our methodology is closely coupled with every stage in the traditional standard-cell design flow. While these stages are alike to the functional simulations for timing closure, the rationale behind them is completely different. In traditional design flow designers care about the values in the steady state, *i.e.* after all transitions have settled, and when does the steady state occur, *i.e.* the critical-path delay. We rather focus on the transitions, observing changes in the instantaneous power consumption caused by an input change. Since we make no assumptions about the functionality of the target circuit—other than it being a digital circuit—this allows us to apply this approach to any standard-cell design. Consequently, we can analyze implementations of masking schemes, standard-cell secure logic styles or any other block of digital hardware in the same manner. In addition to the increased level of physical detail available

at the later design steps, evaluation must be repeated after each synthetic design step to ensure that the EDA tool did not introduce a vulnerability. An example of such vulnerability could be the violation of the non-completeness of the threshold implementations due to the re-timing algorithm optimizing the register locations for performance reasons.

Capturing all transitions of a circuit with n input bits requires simulating $2^{2n} - 2^n$ non-trivial transitions. We call this simulation sequence Exhaustive Dynamic Power Capturing (EDPC). We use Algorithm 3 to ensure the traversal of all non-trivial transitions without repetition. The exponential complexity of the EDPC makes it infeasible for circuits with the large number of input bits. Our tests indicate that the EDPC is feasible for circuits with 16 input bits or less. This is suitable for rigorous evaluations of smaller, but SCA critical, blocks such as the cryptographic S-Boxes. In case of the larger designs, we generate the input vectors in a pseudo-random fashion. This is analogous to the acquisition in the laboratory settings.

Algorithm 3 EDPC Sequence Generation.

function EDPC($nbits$)
  $init \leftarrow 1$, $jump \leftarrow 1$, $node \leftarrow 0$, $space \leftarrow 2^{nbits}$
  for $i \in [0, 2^{2 \cdot nbits} - 2^{nbits})$ **do**
    yield $node$ ▷ EDPC sequence value.
    $node \leftarrow (node + jump) \mod space$
    $jump \leftarrow (1 + jump) \mod space$
    if $node = 0$ **then**
      $init \leftarrow (init + 1) \mod space$
      $jump \leftarrow init$
    end if
    if $jump = 0$ **then**
      $jump \leftarrow 1$
    end if
  end for
end function

Test benches must be carefully crafted to account for the activity of all nodes truthfully. Hierarchical designs may cause the port nets of the individual sub-modules to be annotated multiple times. To avoid counting the contribution of these nodes multiple times two paths can be taken. First, the hierarchical netlist can be flattened after the synthesis. Second, the logic simulator can be instructed to optimize away the redundancy. In QuestaSim this can be facilitated using `-voptargs="+acc=prn+<testbenchModuleName>"` argument of the `vsim` command.

3.5 Computer-Aided Side-Channel Analysis Design Environment (CASCADE)

The goal of CASCADE is to incorporate the design-time SCA evaluations into the standard-cell design flow. We aim to combine the knowledge of both the EDA and the SCA community to develop a tool easily applicable in practice. CASCADE is built around the commercial EDA tools and associated data formats. We adhere to the standard-cell design flow by using the EDA simulators to obtain instantaneous power consumption estimates, starting at the earliest stages of the design. In order to embed the SCA evaluations in all of the standard-cell design stages, we design and implement additional software components that bridge the gap between the EDA tools and the SCA evaluations at design time. This requires tackling several challenges:

- There exists a gap in the current timing and power models used in the standard EDA applications and the SCA evaluations. The timing models are primarily targeted for performance, while the power is primarily a concern for heat dissipation and battery life. In contrast, it is not known how these models can be used for SCA evaluations concerned with the instantaneous power consumption.
- There is a gap in the handling and interpretation of the simulations outputs. SCA evaluations often require processing of millions data-dependent simulations. Therefore, enabling mechanisms to efficiently generate and cope with the sheer volumes of data is of critical significance for practical applications.
- Given the history of the EDA development, and the favorable "best in class" approach, it is important to keep CASCADE flexible, extensible and away from the vertical-integration approach. As the SCA evaluations need to be performed along the full-stack of the front-, and the back-end design stages interoperability must be ensured through adherence to existing data formats and the development of new ones in a transparent manner.

CASCADE bridges these gaps and allows automated and efficient SCA evaluation during all stages of the standard-cell design flow. While it is easily extensible, its principal architecture is depicted in Figure 3.4.

CASCADE is available via a Command Line Interface (`CLI`). The Session Manager (`SM`) is the central part of the framework. Every time a new session is started, a set of `Parameters` is configured and stored within the `SM`. Categories of the parameters are shown in Table 3.2.

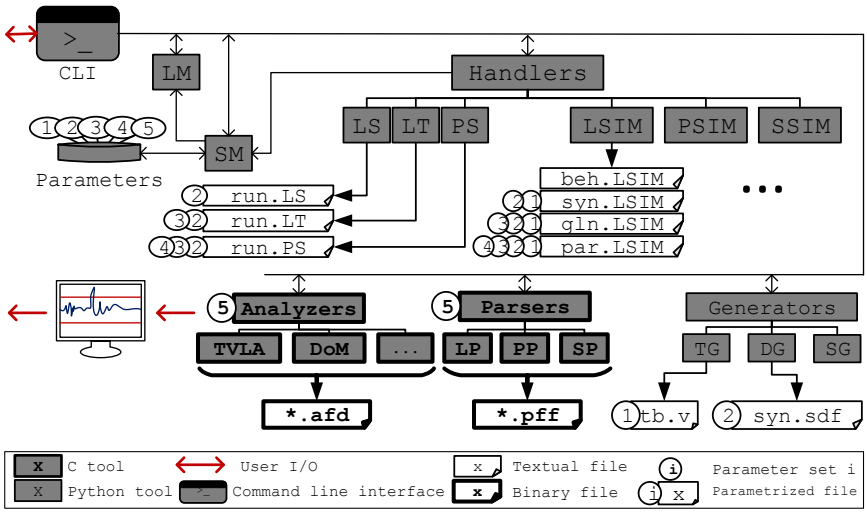


Figure 3.4: High level architecture of CASCADE.

The `SM` centrally manages all configuration parameters. There are other ways of achieving such centralization, *e.g.* using CMake or similar software. However, `SM` does not only evaluate the parameters, but often involves sanity checks and conversions to formats catered to target tools. We opt for this centralization to ensure coherency between tools, thus avoiding time-loss due to the error prone manual handling. The Library Manager (`LM`) parses and handles standard-cell library files. When working with a single library, `LM` data can be easily hardcoded in the CASCADE configuration. The remaining components are: `Handlers`, `Generators`, `Parsers` and `Analyzers`. CASCADE components relate to the typical side-channel measurement setup shown in Figure 2.5 as per Figure 3.5.

Table 3.2: Configuration parameters.

	Category	Examples
①	Simulation	*precision, duration*
②	Design constraints	*critical path*
③	Resources	*library resources*
④	Physical constraints	*placement constraints*
⑤	Power	*model parameters*

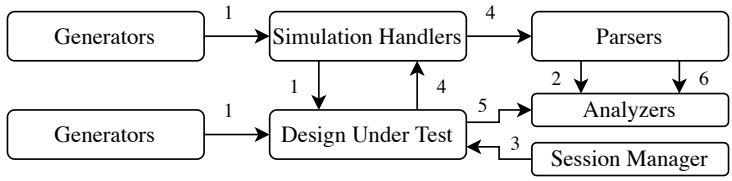


Figure 3.5: CASCADE in relation to a SCA measurement setup.

Handlers.

`Handlers` wrap the commercial EDA and the custom CASCADE tools alike, abstracting their functionality, vendor, and software version. Each handler can be modified, and the new ones can be created, independently from the rest of the framework. This makes CASCADE easily adaptable to any changes in the underlying tools or the flow itself. `Handlers` facilitate a synthetic or an analytic step in a streamlined and automated manner. They produce TCL scripts (*e.g.* `gln.LSIM` for the logic simulation at the GLN stage) used to drive the underlying tools. Depending on the point in the flow, TCL scripts are associated with different categories of parameters. Any change in the session parameters is automatically propagated along the entire flow. The set of the commercial EDA tools we currently use is given in Table 3.3.

Table 3.3: List of commercial EDA tools used.

Acronym	Function	Tool
`LS`	Logic synthesis	Synopsys Design Compiler
`LT`	Library translation	Synopsys Design Compiler
`PS`	Physical synthesis	Cadence Innovus
`LSIM`	Logic simulation	MentorGraphics QuestaSim
`PSIM`	Physical simulation	Synopsys PrimeTime, PX
`SSIM`	SPICE simulation	Synopsys HSPICE

The traversal of the design stages is depicted in Figure 3.6. We start from the Register Transfer Level (RTL) code of the design. The behavioral modeling can be also simulated in the flow, but as it is a black box representation of the mathematical algorithm it is more efficient to simulate it in software. The logic functionality is synthesized (SYN) using the generic logic gates. This functionality is then mapped to library cells of a concrete library to form a gate-level netlist (GLN). The placed-and-routed (PAR) design stage comes before the physical extraction and the tapeout. CASCADE enables the SCA

evaluation at the every stage of the design flow, providing feedback to the designers. Similarly to the timing closure, proceeding to the next stage is allowed once the security requirements are fulfilled for the current stage. We perform these simulations using models described in Section 3.4.4.

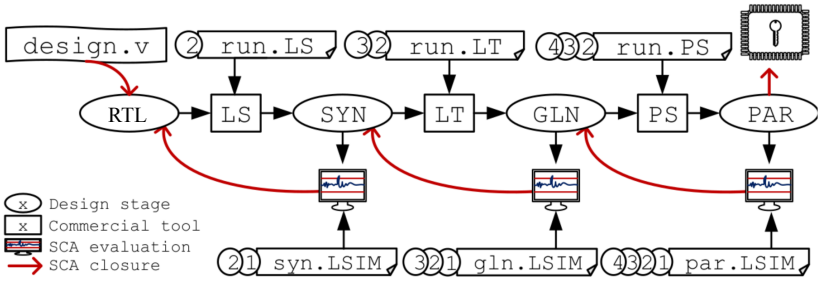


Figure 3.6: Standard-cell design stages using CASCADE.

Generators

`Generators` aid the automation by synthesizing scripts and data files in conformance with the session manager.

Test bench generator (`TG`) produces test benches based on the Hardware Description Language (HDL) code of the design (*e.g.* `tb.v`) and parameters obtained from the `SM`. `TG` parses the Verilog netlist, wires the design and facilitates the control signals for data input and capturing (*e.g.* `trigger` signal that indicates when to record power consumption). All status and control signals used to configure and run a particular design should be handled manually. The testbenches for different configurations can be easily added. Input data is read from a binary file, resulting in otherwise unchanged structure for each testbench. Depending on the desired test `TG` generates different data vectors (*e.g.* user-supplied functional tests, (pseudo-)random inputs or the EDPC sequence.) In the case of pipelined designs, such as Boolean masking schemes, to observe the worst case we keep the inputs stable until data has finished propagation through all pipeline stages. Feeding subsequent data would introduce noise as the pipeline would be computing on multiple statistically independent inputs at the same time.

Delay Generator (`DG`) is used to annotate generic netlists at the SYN design stage (*c.f.* Δ-delay in Section 3.4.4). Delay annotations are stored in the Standard Delay Format (SDF), compliant with modern EDA tools. SPICE Generator (`SG`) includes a translator from Verilog to SPICE netlists, as well as an analog version of the test bench generator.

Parsers

Similarly to data acquisition tools used in measurement setups, we design optimized `Parsers` to process and store data in a SCA-friendly manner. We design and implement them in in C. Regardless of the type of data we parse, Logic Parsers (`LP`), Power Parsers (`PP`) and SPICE Parsers (`SP`) output a Power Frame File (PFF). PFF is a custom binary format designed for storage and fast processing of the simulated instantaneous power consumption traces. As SCA evaluations require orders of magnitude more traces to be recorded than the traditional digital design, we believe that storage savings from binary encoding compared to human-readable formats often used for the EDA tools, can be immense. For the detailed PFF specification see Appendix A. We refer to the part of the simulation that corresponds to one power trace as a simulation *frame*. Each frame starts with data associated with the frame transitions followed by time-value pairs of discrete digital events. A number of data vector can be associated to each of the frames. We use input, output and target data vectors. CASCADE configuration allows binding these three vectors to arbitrary nodes. The latter allows us to leverage the native simulators to perform any post-processing, *e.g.* unsharing the sensitive variable. Frame-associated data also supports the functional validation of the design and the SCA processing, *e.g.* frames can be partitioned on the fly. The associated data vectors may as well be stored in separate files, as they require orders of magnitude less memory than the corresponding trace-data. Therefore there is no need in handling them on the fly. Nevertheless by attaching them to each frame we achieve a level of integration that minimizes human error during evaluation. Additionally, we can acquire logic values of all intermediary nodes and use if for debugging and verification purposes.

Analyzers

Lastly, we use the `Analyzers` to process PFF files. We design and implement them in in C. Each analyzer implements a specific SCA evaluation technique, *e.g.* TVLA, or an attack, *e.g.* Correlation Power Analysis (CPA). Focusing on the TVLA, we follow the roadmap put forward by Schneider and Moradi [132]. We abstain from applying the faster leakage assessment of Reparaz et al. [124] because of the prohibitive storage costs. We discuss this topic further in Section 3.7. The analysis consists of three steps that are performed on the fly for each frame:

1. A continuous power waveform is reconstructed from the frame data, PFF header information and desired parameters.

2. Context of the analyzer is updated using this waveform, according to the associated data.

3. The context is evaluated and the output is written to an Analyzed Frame Data (AFD) file, a custom binary designed for providing visual feedback to the designers.

For the detailed AFD specification see Appendix A. The latter step is mandatory after the final frame, but can be done periodically to observe the evolution of the chosen SCA metric. AFD files can preserve the analyzer context, so that on-the-fly evaluation can be continued at will. This allows a simple way to split an evaluation into multiple batches of traces.

3.6 Experimental Validation

In this section, we validate CASCADE by applying it to representative cryptographic circuits. The security properties of these circuits are well established, and therefore allow us to check the capabilities of our framework.

We show how CASCADE can be applied to both masked designs instantiated to provide *first-order* security, *i.e.* devised to resist power analysis attacks that exploit information leakages in the first-order moment, as well as standard-cell-based secure logic styles. Evaluation in higher-order requires simply changing the analyzer used. Lastly, we benchmark CASCADE using several circuits of different input spaces and area to test the scalability of CASCADE. We perform all experiments using a 45 nm standard-cell library from NanGate [78].

3.6.1 A Motivating Example

We use the first-order Domain Oriented Masking (DOM)-*indep* multiplier, a masked AND gate from [61] depicted in the Figure 3.7 as the motivating example. In this simple circuit, input and output variables are split into 2 shares such that $a = a_1 \oplus a_2$, $b = b_1 \oplus b_2$ and $z = z_1 \oplus z_2 = a \cdot b$. The design consumes one random bit r per evaluation. Figure 3.8 shows power profiles (averaged instantaneous power consumption) obtained using the symmetrical $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power model, defined in Equation (3.3), combined with the different timing models. We show the principal timing diagram for this circuit in Figure 2.12 and discuss its operation and security properties in Section 2.3.2.

Given the five input data bits, the EDPC consists of $2^{2 \cdot 5} - 2^5 = 992$ input transitions. As the entire distribution of power traces is exhausted and there is

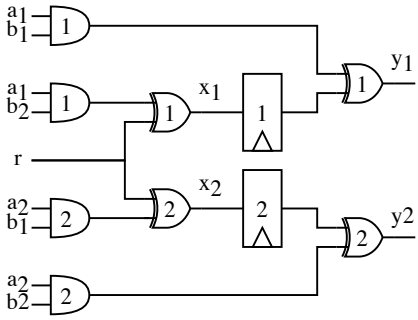


Figure 3.7: DOM-*indep* multiplier.

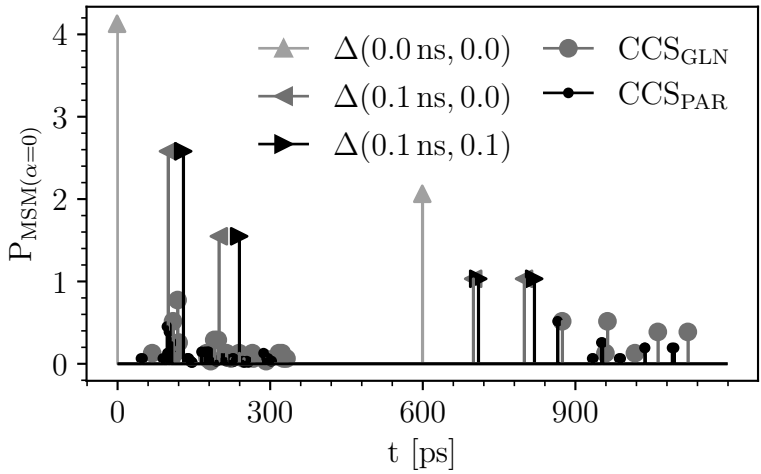


Figure 3.8: DOM-*indep* multiplier, MSM power profiles.

no noise, we can simply look at the difference of the distribution moments, and decide whether they are distinguishable with the 100% confidence. Figure 3.9 shows the difference of means (left) and of variances (right), partitioned based on the unshared output value $z = z_1 \oplus z_2$. The two sets are indistinguishable in the first-order moment, as the circuit is first-order secure, and vice versa. However, the difference of variances testifies to the vulnerability in the second-order, for both the Δ-delay and CCS timing models.

In order to validate how different timing models detect information leakage, we induce two flaws in the design. First, we break the input independence assumption, by setting $a_1 = b_1$. In practice, input dependency can occur during composition and outputs have to be registered to mitigate this as per Faust *et*

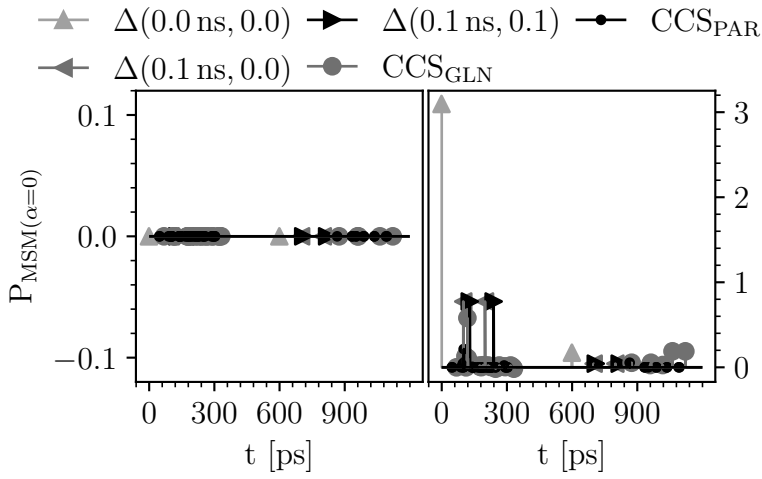


Figure 3.9: DOM-*indep* multiplier, difference of means (left), difference of variances (right).

al. [48]. In terms of threshold implementations, this is equivalent to breaking the input uniformity. Figure 3.10 shows the difference of means (left) and of variances (right), partitioned based on the unshared output value $z = z_1 \oplus z_2$ when $a_1 = b_1$. In addition to the second-order moment, the means of the two distributions differ. Note that the simple Δ-delay models can not differentiate the two distributions in the second clock cycle.

Next, we fix the bit $r = 0$, effectively disabling the masking. Figure 3.11 shows the difference of means (left) and of variances (right), partitioned based on the unshared output value $z = z_1 \oplus z_2$ when $r = 0$. This time the first-order moments are distinguishable only in the second clock-cycle. The indistinguishable first-order moments in the first-cycle are expected, as the remasking is used to secure the compression step after the register.

For all the models employed, second-order evaluations detect leakage, using even the simplest zero-delay model. However, for both flaws we introduce CCS models exhibit different behavior than the Δ-delay models. CCS timing models show four distinct points in which the first-order moments differ. These points of distinction are determined by the asymmetries of the compression XOR gates. Even for the fixed output load, z_i timing depends on the transition direction (rising or falling) and the $a_i \cdot b_i$ product value. As such timing differences affect the data-dependent distribution of glitches, they must be considered when evaluating Boolean masking schemes that claim glitch-resistance. Therefore, design-time evaluations benefit from CCS timing models. For the earlier, library-

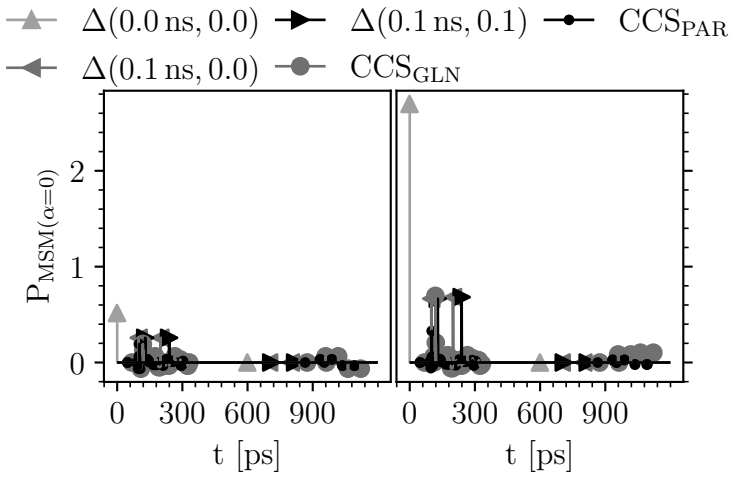


Figure 3.10: DOM-*indep* multiplier, when $a_1 = b_1$ difference of means (left), of variances (right).

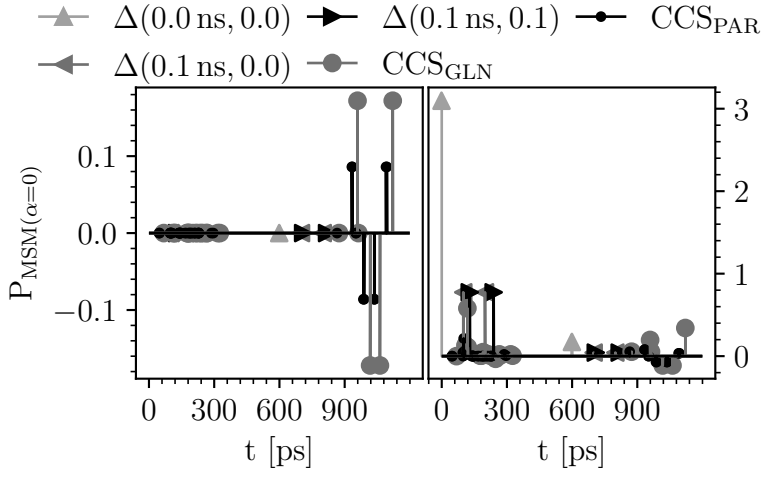


Figure 3.11: DOM-*indep* multiplier, when $r = 0$ difference of means (left), of variances (right).

independent, evaluations asymmetry can be introduced in the Δ-delay models for the glitch-resistant masking scheme evaluation. It suffices to mention that any standard-cell-based secure logic style, that relies on the symmetry, benefits from evaluations using CCS models.

So far we present only the non-normalized differential traces, as we exhaust all transitions. Input sizes of practical circuits prohibits such analysis, and therefore normalized t-traces have to be used. Moreover, all-zero non-normalized differential traces can be obtained using the $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power model equivalent to the Hamming distance power, *i.e.* by simulating the toggle distribution. If an asymmetry is introduced, either using $\mathrm{P}_{\mathrm{MSM}(\alpha\neq 0)}$ or detailed CCS power models, *i.e.* by simulating the physical behavior, the non-zero values—different from the floating point calculation errors—will occur in the non-normalized differential traces without endangering the side-channel security. Thus, it is beneficial to normalize the differential traces to obtain a confidence interval. Figure 3.12 shows the power profile and security evaluation using CCS power models to demonstrate this.

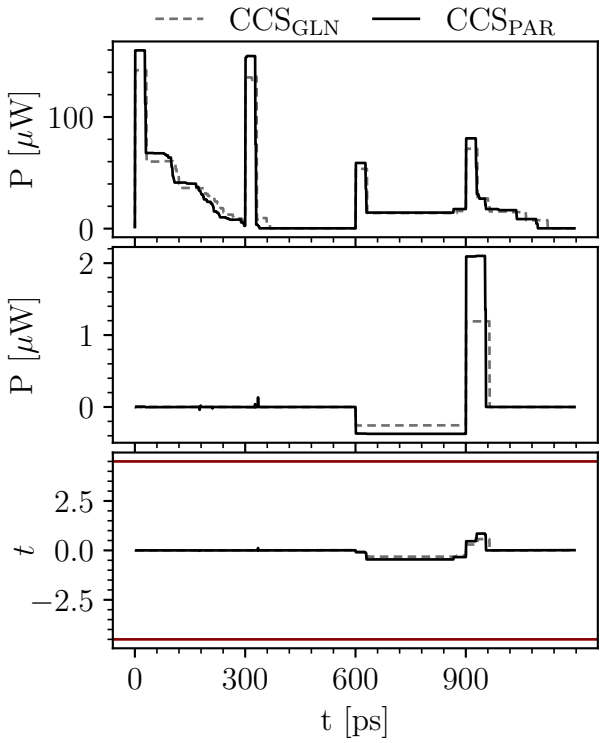


Figure 3.12: DOM-*indep* multiplier, power profile (top), difference of means (middle), first-order t-trace (bottom) using CCS power models.

The non-zero values in the differential trace occur only in the second clock cycles. Also, in the first clock cycle gates start right after the starting clock edge, while the switching is delayed by about 300 ps in the second one. The reason for this

is that the XOR gates in the second clock cycle are driven by a flip-flop, while inputs to the first cycle are supplied directly from the simulation. Normally, we would constrain the drive of the input bits to emulate drive of a flip-flop. We remove this constraint to show how it affects the physical simulation.

To make sure that the non-zero t-score in Figure 3.12 (bottom) is not a consequence of the limited number of traces, we show the evolution of the t-statistic in Figure 3.13 (left). The downwards trend testifies to the relevance of the result, as expected for the secure implementation of this circuit. In contrast, Figure 3.13 (right) shows the lack of the second-order security.

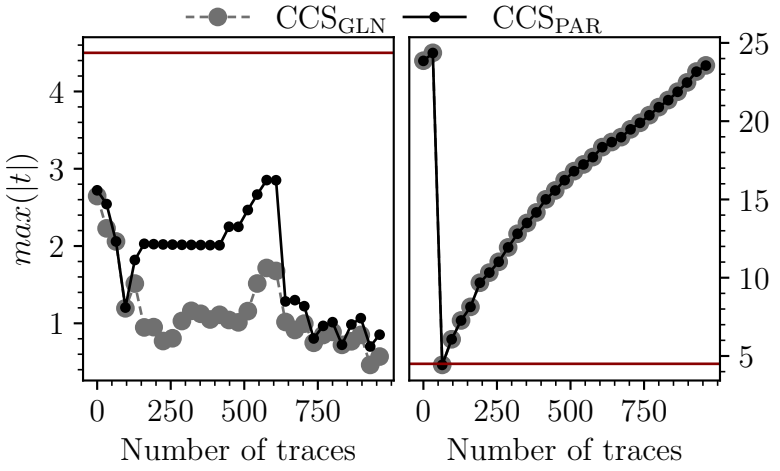


Figure 3.13: DOM-*indep* multiplier, the first-order (left) and the second-order (right) t-statistic evolution.

Lastly, we plot the t-statistic evolutions for the two flaws we introduce. We show results using both the $\mathrm{P}_{\mathrm{MSM}}(\alpha = 0)$ and the CCS power. We observe the t-statistics separately in each cycle for a more detailed discussion.

Figure 3.14 shows the t-statistic evolution in both clock cycles using the $\mathrm{P}_{\mathrm{MSM}}(\alpha = 0)$ power model. To set $a_1 = b_1$ we select a subset of 496 traces for which this holds. Despite the t-statistic staying below the $|t| = 4.5$ range, the sharp upwards trend testifies to the composability issue discussed in [48]. All t-values reach the 99.9% confidence to reject the null hypothesis. In the second cycle, the non-zero difference of means leads to a flat t-statistic trend and a low confidence that the two sets can be discerned.

Figure 3.15 shows the t-statistic evolution in both clock cycles using the CCS power models. Despite the equally small number of traces, increased level of

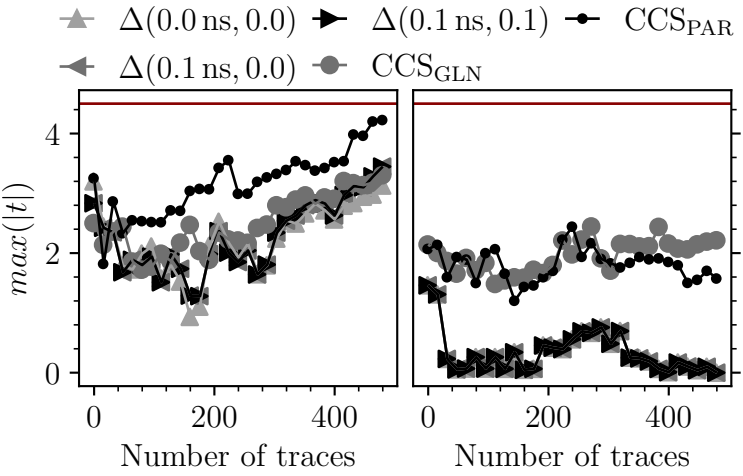


Figure 3.14: DOM-*indep* multiplier, when $a_1 = b_1$ first-order t-statistic evolution in the first-cycle (left) and the second-cycle(right), using $\mathrm{P}_{\mathrm{MSM}}(\alpha = 0)$.

physical detail increases the confidence well beyond the 99.999% margin typical for the TVLA.

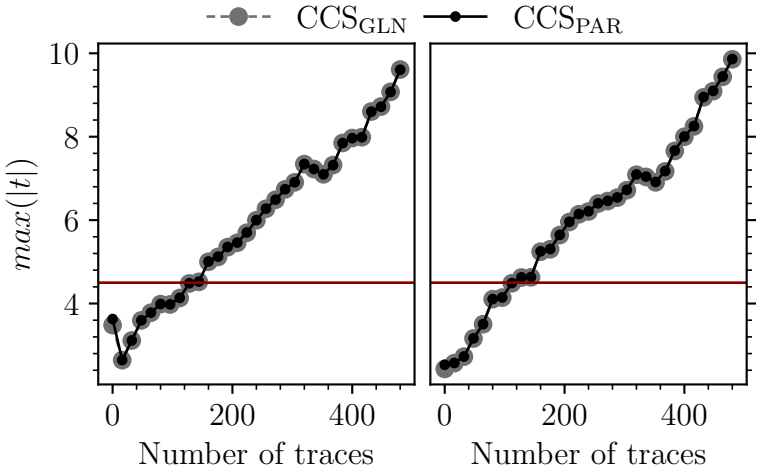


Figure 3.15: DOM-*indep* multiplier, when $a_1 = b_1$ first-order t-statistic evolution in the first-cycle (left) and the second-cycle(right), using CCS power.

Figure 3.16 shows the t-statistic evolution when the random bit is reset. This scenario results in a subset of 496 traces. In the first clock cycle, the t-statistic

quickly evolves to zero-confidence—inline with the security properties of the circuit and the difference of means shown in Figure 3.9. However, when $r = 0$ t-statistic in the second clock cycle sharply increases beyond the $|t| > 4.5$ interval for the evaluation using CCS timing.

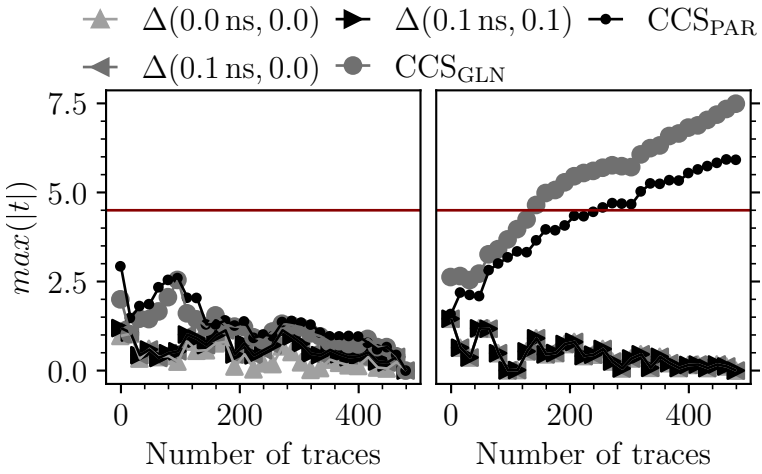


Figure 3.16: DOM-*indep* multiplier, when $r = 0$ the first-order t-statistic evolution in the first-cycle (left) and the second-cycle(right), using $\mathrm{P}_{\mathrm{MSM}}(\alpha = 0)$.

Figure 3.17 shows the t-statistic evolution in both clock cycles using the CCS power models. Both trends confirm the security properties of the DOM-*indep* multiplier, and are inline with the $\mathrm{P}_{\mathrm{MSM}}(\alpha = 0)$ power model with CCS timing.

Therefore, our framework evaluates the side-channel security of this masked design successfully and in accordance with the known security properties of the circuit [48]. We show how the increased level of physical detail, available in the later stages of the standard-cell ASIC design flow, can be leveraged to detect the security flaws with increased confidence. Nevertheless, some flaws in the implementation can be detected even using the simple Δ-delay models and the $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power.

Additional benefits of hardware simulation.

The precision and the discrete nature of models allow for another advantage of the simulated approach. Given a set of input vectors, for each gate G^i it is possible to annotate n^i discrete moments $t^i_1, t^i_2, \ldots, t^i_{n^i}$ relative to the clock edge at which the gate switches, in every trace. In very small circuits with Δ-delay models these moments will likely overlap. As the circuit grows and

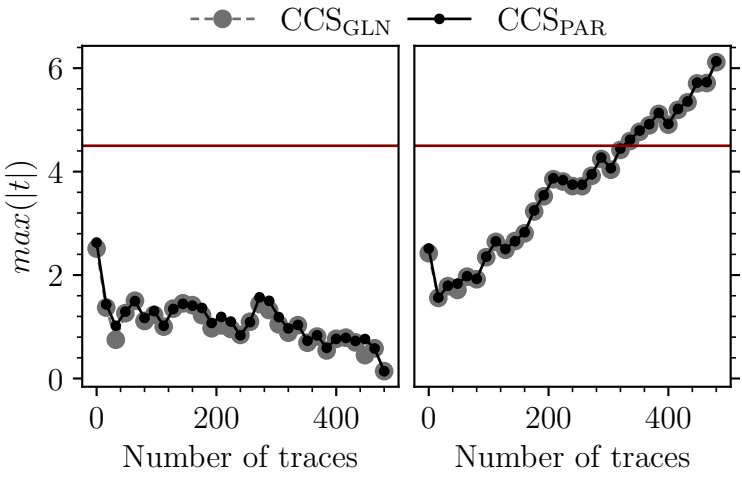


Figure 3.17: DOM-*indep* multiplier, when $r = 0$ the first-order t-statistic evolution in the first-cycle (left) and the second-cycle (right), using CCS power.

more detailed models are used $t^i_1, t^i_2, \ldots, t^i_{n^i}$ starts to vary for each G^i. For example, Δ-delay can be replaced with $\Delta + \delta^i$ such that $\delta << \Delta$, to allow for a more unique set of $t^i_1, t^i_2, \ldots, t^i_{n^i}$ per gate. Consequently, leakage detected in an interval $t_0 \leqslant t < t_1$ can be traced back to a subset of gates, possibly of cardinality one, in the design. By doing so, the sources of leakage are localized to the logical cones of the said subset of gates. This can be of great help during implementation debugging and it can be achieved using commercial EDA tools and a bit of scripting.

3.6.2 Protected S-Boxes

We now move to show how our approach and the CASCADE framework scale in terms of circuit sizes, number of traces and the security evaluation capabilities. To do so, we evaluate several protected S-Boxes protected using different standard-cell-based countermeasures. The S-Boxes are often the most SCA-vulnerable parts of the cryptographic algorithms due to their non-linearity, and as such they have to be rigorously evaluated starting from the earliest design stages.

TI Present S-Box

We target the first-order secure masked implementation of the PRESENT S-Box by Poschmann *et al.* [117], depicted in Figure 3.18. The design is decomposed into two quadratic S-Boxes F and G. They are split into three shares per variable in accordance with the TI principles. The total number of input- and output-bits is 12, resulting in $2^{2 \cdot 12} - 2^{12}$, *i.e.* approximately 16 million, transitions long EDPC sequence.

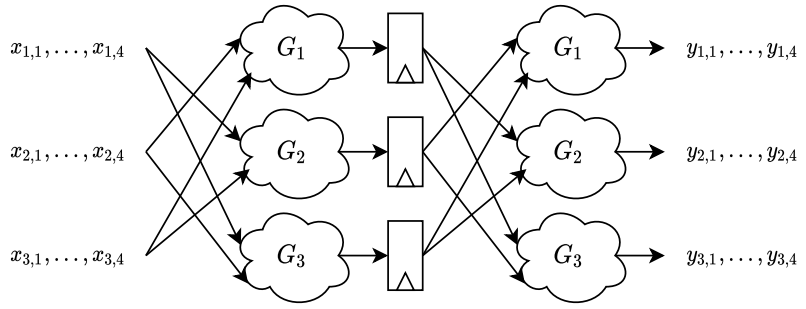


Figure 3.18: Architecture of the TI PRESENT S-Box.

As the circuit is designed to provide the first-order security, we first evaluate the second-order leakage. The results, depicted in Figure 3.19, are inline with the theory for all the timing and power models we consider. Figure 3.19 (left) shows the second-order $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power t-traces for several timing models. The second-order leakage is even more prominently detected using the PTX simulations based on the CCS timing and power models, as depicted in Figure 3.19 (right). For both the $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ and CCS power models the second-order is present in both cycles, as both decomposed stages are first-order secure TI.

Figure 3.20 shows the evolution of the first-order t-statistic using $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power with different timing models. As in the motivational example, traversing the entire EDPC sequence t-statistic evolves towards zero as the two distributions of the $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power samples get fully populated. Using $\mathrm{P}_{\mathrm{MSM}(\alpha\neq0)}$ results in the similar Lastly, note that the initial t-values outside the confidence interval are a statistical artifact due to the low number of traces processed. Hence, it is more important to observe the trend of the t-trace evolution, not a single evaluation with the small number of traces.

Figure 3.21 shows the first-order evaluation using CCS power models in PTX simulation. At the pre-layout GLN stage, there is no significant information leakage. However, t-value does not evolve to zero as all traces are exhausted.

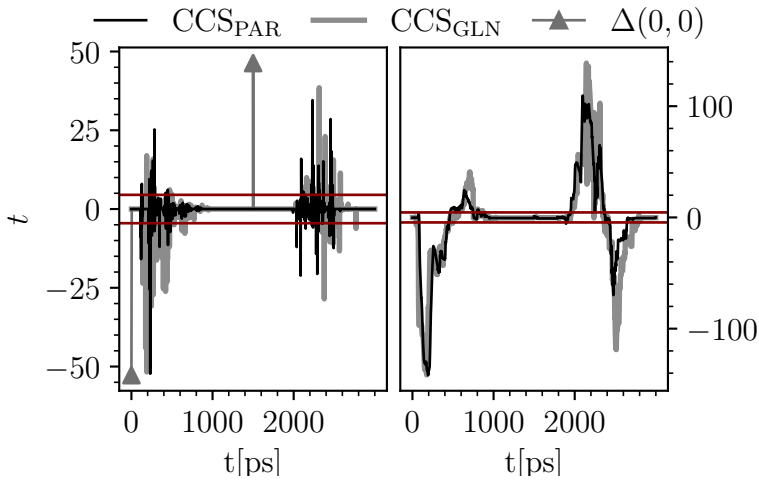


Figure 3.19: TI PRESENT S-Box, the second-order t-trace using $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power (left) and CCS power (right).

Compared to the $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$, physical asymmetries captured by the CCS models map identical toggles to different values. Consequently, the PTX simulation can yield non-zero values in the t-trace for a secure glitch-resistant masking

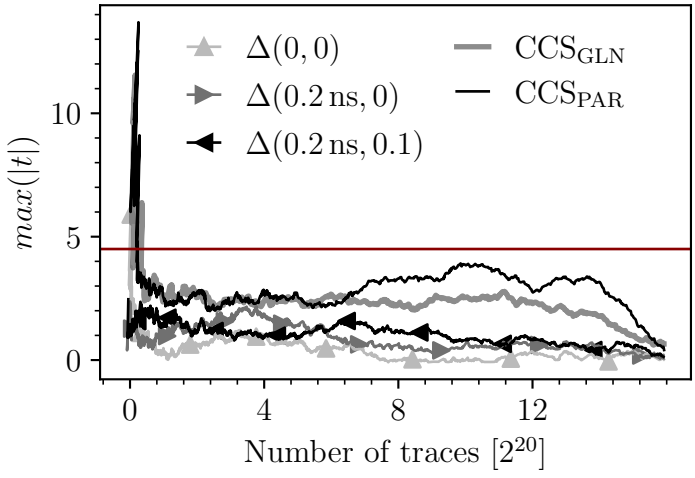


Figure 3.20: TI PRESENT S-Box, the first-order t-trace evolution using $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power.

scheme. Our initial evaluation at the PAR stage, indicated first-order leakage.

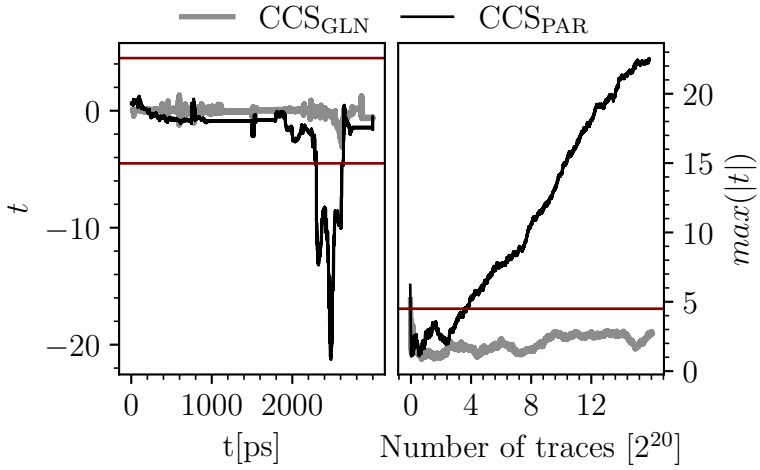


Figure 3.21: TI PRESENT S-Box, the first-order t-trace (left), and its evolution (right) for, using CCS power; false positive evaluation.

Both G and F are first-order secure Boolean functions, as per [117]. Consequently, the TI PRESENT S-Box should exhibit the first-order security in both clock cycles. We first test their individual security, running a separate experiment for each of the two functions. Then we simulate the EDPC sequence for both G and F, using $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ and CCS power models with CCS timing. In all cases, G and F are first-order secure. We then inspect the design files more closely. Upon inspection of the Synopsys Design Constraints (SDC) file, we find several negative time constraints applied to the register bits and the clock. Other design files unchanged[1], we remove these constraints and repeat the experiments to resolve this *false-positive* leakage assesment. Figure 3.22 shows the correct results of the first-order evaluation using CCS power models. This example shows the importance of properly constraining designs. We address this matter further in Section 3.7.

Lastly, we evaluate a known vulnerability using different models. Figure 3.23 (left) shows the first-order t-trace evolution using $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ when $x_{3,1} = x_{3,2} = x_{3,3} = x_{3,4} = 0$, *i.e.* one mask is turned off. Leakage is correctly detected even with the simplest $\Delta = 0$ model, roughly after processing 250 thousand traces. Other timing models, require as little as ten thousand traces. Results obtained when using the PTX simulation are shown in Figure 3.23 (right). In

[1]Also, the version of the PrimeTime changed to 2019.03.

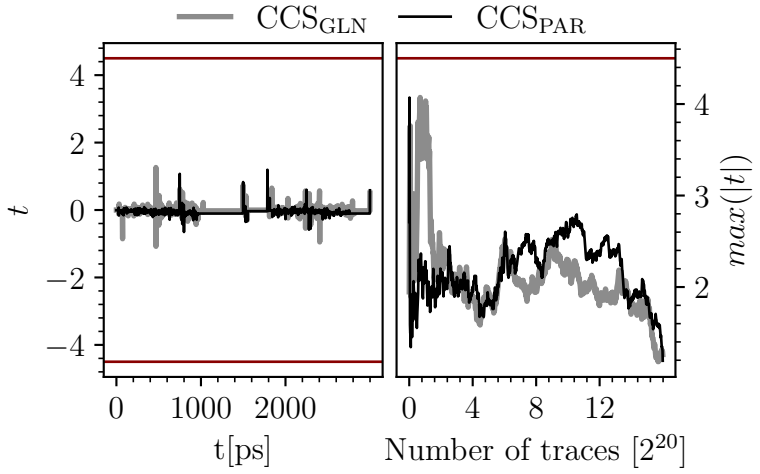


Figure 3.22: TI PRESENT S-Box, the first-order t-trace (left) and its evolution (right), using CCS power.

this case, the first-order leakage is visible after processing the first two thousand traces.

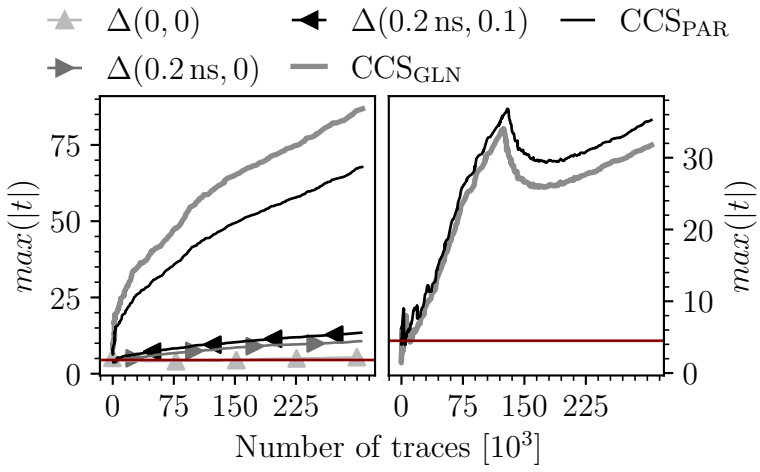


Figure 3.23: I PRESENT S-Box, the first-order t-trace evolution when $x_{3,1} = x_{3,2} = x_{3,3} = x_{3,4} = 0$, using $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power (left) and CCS power (right).

Masked Boyar-Peralta AES S-Box.

Ghoshal and De Cnudde [56] proposed a first-order secure implementation of the Boyar-Peralta (BP) AES S-Box, designed to consume no randomness. In a later work, Wegener and Moradi [148] showed that this design exhibits leakage due to a non-uniformity problem. Their experiments were carried using an FPGA setup and processing 10 million measurements. We validate the same vulnerability can be captured using CASCADE. In particular, our experiments indicate the presence of significant leakage starting from 400 thousand traces using $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ with the CCS timing models of the GLN. Such evaluation can be performed in 30 minutes, including the manual work, using a single desktop workstation.

WDDL Present S-layer.

We now move to evaluate a WDDL implementation of the PRESENT S-Layer, consisting of 16 S-Boxes in parallel. We use the same 45 nm library from NanGate [78] as before. As the PRESENT P-Layer consists of routing wires only, we effectively evaluate a round-based implementation of PRESENT. Since WDDL relies on balancing the differential rails, we observe multiple S-Boxes in parallel to capture the asymmetries introduced by the placement and routing better. Given the infeasibility of exhausting all $2^{2 \cdot 64} - 2^{64}$ EDPC transitions, in this experiment we perform a classical fixed-versus-random TVLA using 10 million traces. We partition based on the input value of one S-Box while the inputs of the remaining S-Boxes are random, hence generating algorithmic noise. Pre-charge circuitry and the data-splitting into the differential rails is done in the test bench, so the circuit has ideally aligned inputs.

Perfectly symmetrical $\Delta_{\delta=0,\theta=0}$ and $\Delta_{\delta>0,\theta=0}$ models yield an all-zero differential traces in the first- and the second-order TVLA as long as the logic structure is implemented correctly. As each pair of complementary gates has the same fanout, $\Delta_{\delta>0,\theta>0}$ models neither show leakage up to 10 million traces.

However, the first-order leakage appears very quickly when the asymmetrical CCS timing and power models are used. Figure 3.24 shows the t-trace evolution for the first 100 thousand traces. As we do not use the balanced routing strategy put forward by Tiri and Verbauwhede [145], this result is expected. Interestingly, the crude $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power model discerns the two distribution with significantly higher confidence than the detailed CCS power models. We address this matter further in Section 3.7.

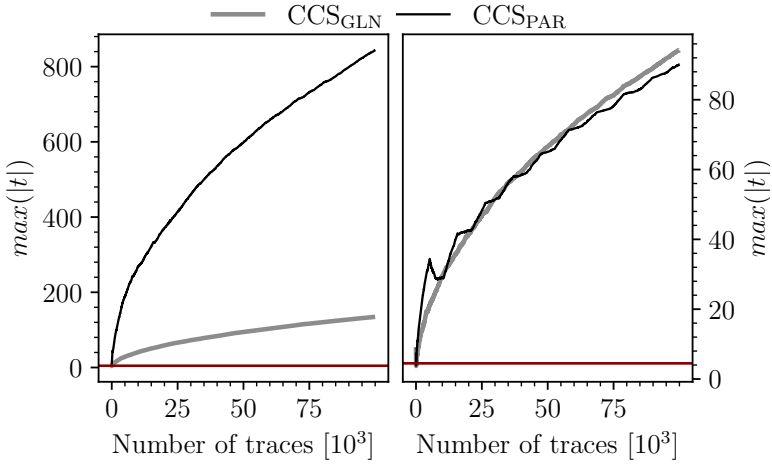


Figure 3.24: WDDL PRESENT S-Layer, the first-order t-trace evolution using $P_{MSM(\alpha=0)}$ (left) and CCS power (right) at 1 ps.

We repeat the experiment using a 10 ps simulation precision, and plot the results in Figure 3.25. The leakage is detected, although $P_{MSM(\alpha=0)}$ simulations show lower confidence.

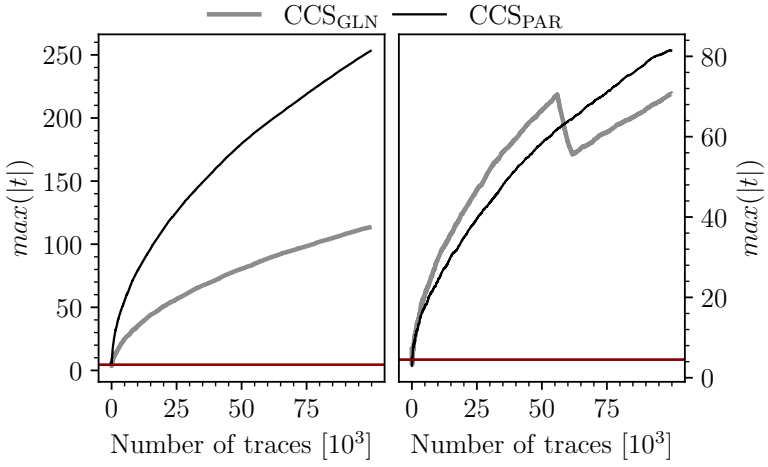


Figure 3.25: WDDL PRESENT S-Layer, the first-order t-trace evolution using $P_{MSM(\alpha=0)}$ (left) and CCS power (right) at 10 ps.

3.7 Discussion

In this section we discuss the design-time SCA evaluations using CASCADE. We start by arguing its utility for the digital designers. We then address several practical considerations important for the more reliable application. Lastly, we argue performance and the scalability of the approach, showing several benchmarks.

3.7.1 Utility to the Designer

CASCADE is designed in compliance with the commercial EDA tools and standard data formats. Therefore, it can be easily integrated in a designer's toolbox with little to no training overhead. It allows efficient early SCA evaluations of critical building blocks. Thus, vulnerabilities can be stopped from propagating to the post-layout and manufacturing stages of the design. Designers may pinpoint bugs and flaws, and proceed to fix them before moving on to the next stages. CASCADE can be used regardless of the target countermeasure, as long as the design is implemented using standard-cell libraries. All data and control paths along with any other auxiliary gates are treated uniformly and automatically, without the need for additional modeling. The previously mentioned analysis of the masked Boyar-Peralta S-Box implementation is a small example of CASCADE's practical utility. As a largely automated framework it can be used effectively, avoiding problems with measurement setups and saving time.

3.7.2 Models and Countermeasures

We demonstrate how to evaluate representatives of the two leading standard-cell based countermeasures using increasingly detailed models. Even the simplest zero-delay simulation can be used to uncover some of the vulnerabilities. Therefore it is possible to start the side-channel evaluation from the earliest design stages. We now discuss the advantages and potential shortcomings of the evaluations.

Glitch-resistant Boolean masking schemes.

Glitch-resistant Boolean masking schemes provide security by adding redundant logic that generates algorithmic noise independent of the processed secret. As

long as their assumptions hold they can offer mathematical proofs of side-channel security. Here we distinguish two types of assumptions. The "logic" assumptions dictate the properties of the Boolean functions, *e.g.* uniformity or non-completeness of the threshold implementations. The one "physical" assumption dictates that each share must leak information independently. While the assumptions hold, Boolean masking schemes are impervious to glitches. However, if one of the assumptions is violated, it is beneficial to use the timing and power models that instigate the highest glitching activity. By doing so, the potential vulnerability is captured with the smallest number of traces. More importantly, if too simple of a model is used, glitching of some parts of the circuit remains not captured by the simulation, *e.g.* the output XOR gate in the motivational example, even for the exhaustive coverage. Therefore, the high level of hardware asymmetries and the high simulation precision are advantageous. We present the results simulated at the 1 ps precision. However, using the 10 ps step we observe no loss in the quality of results. Further reduction to the 100 ps still yields consistent evaluations, only using a larger number of traces. Therefore, we find the gate-level simulation yield efficient and reliable pre-layout side-channel security evaluation of Boolean masking schemes—as long as the physical independent leakage assumption holds.

However, for the complete evaluation the independent leakage assumption must be considered too. More recent works [30, 86, 154] indicate that several layout and measurement factors can deteriorate the side-channel security of glitch-resistant Boolean masking schemes by affecting the independent leakage assumption. All the gate-level models used in this chapter assume a constant, ideal and independent power supply for each gate; recall Equation (3.1). We address this issue in more detail in Chapter 5. For now we conclude that gate-level simulators, as long as they are used properly, provide an optimistic prediction of the side-channel security.

Standard-cell-based secure logic styles.

Standard-cell-based logic styles such as WDDL provide security based on the physical symmetries. Similarly, like in the secure-logic styles simpler models can be used to detect flaws in the logic structure, *i.e.* whether the implemented logic is WDDL-compliant. However, the difficulty of implementing and evaluating WDDL designs lies in the backend. Notably, WDDL can provide a much higher level of security than the one we demonstrate, as we do not perform the balanced routing intended by the original authors. In any case, the evaluation of WDDL circuits depends on how accurately can the asymmetries be observed. Interestingly, the simpler $\mathrm{P}_{\mathrm{MSM}(\alpha=0)}$ power model produces a much starker difference between the two TVLA sets, compared to the detailed

CCS power. We believe the $P_{MSM(\alpha=0)}$ combined with the 1 ps precision give an overly pessimistic evaluation. We demonstrate this by lowering the simulation precision to 10 ps. This decreases the t-statistic threefold, while barely changing the evaluation using CCS power. Furthermore adding random jitter between 0 and 10 ps to the $P_{MSM(\alpha=0)}$ traces reduces t-statistic by two orders of magnitude. We believe that his jitter very difficult to achieve using measurement equipment. As WDDL does not rely on the algorithmic noise, and the gate-level simulators provide unrealistically stable and precise measurements, we believe the predictions to be pessimistic.

On the other hand, the gate-level simulations do not model physical phenomena, such as the variations in the manufacturing process that make every chip unique. Said variations are known to harm the SCA security of standard-cell-based secure logic styles. Similarly to the design-time evaluations of the Boolean masking schemes, additional layout phenomena certainly increase the gap between the measurements of an ASIC target and the simulated traces. However, this gap exists for all models as well as for the FPGA-based prototyping.

3.7.3 On the Importance of Design Constraints

In Section 3.6.2 we use a false positive assessment (*i.e.* the tools indicates leakage, where there is no vulnerability) to demonstrate the importance of design constraints. Namely, as delay calculation and power estimation in the commercial EDA tools are not designed for SCA evaluations they allow different settings that make little difference to the performance and average power of the designs. In this particular example, possibly through manual error, we synthesize a trivial single-buffer "clock tree". The physical synthesis tool infers negative timing constraints, that affect the traditional design parameters marginally. However, as SCA methods excel at detecting small data-dependent variations, this is enough to cause a faulty assessment.

Another important aspect are the default settings that simplify the evaluation in the early stages. For example, `high_fanout_net_threshold` determines the maximal fanout for which it performs the delay calculation. Global signals, such as clock and reset, as well as the multiplexer-control signals in wide data-paths, can have unrealistically high fanouts. Consequently the tools introduce a data-dependent non-linearity to optimize their performance. Again, this does not impact the traditional design goals significantly, but can create false leakage.

3.7.4 Performance

With 350 GE in size, the masked TI PRESENT S-Box is a small, but critical, design block. Its sufficiently long EDPC sequence makes the simulation efforts non-trivial and convenient for comparing performance given a fixed circuit. Table 3.4 summarizes runtimes of different CASCADE modules when processing $2^{12 \cdot 2} - 2^{12}$ traces using a single thread of an Intel i7-7700 desktop workstation

Table 3.4: TI PRESENT S-Box, benchmarks for tools, stages and models.

Stage	Models	Simulation	Parsing	1st-ord.	2nd-ord.
SYN	$P_{MSM(\alpha=0)}$, $\Delta^{1}_{\delta=0,\theta=0}$	4.85	1.65	0.03	0.05
SYN	$P_{MSM(\alpha=0)}$, $\Delta_{\delta>0,\theta=0}$	7.15	1.98	0.92	4.13
SYN	$P_{MSM(\alpha=0)}$, $\Delta_{\delta>0,\theta>0}$	7.30	2.13	0.97	4.16
GLN	$P_{MSM(\alpha=0)}$, CCS time	11.38	2.14	1.05	4.23
GLN	CCS power and time	57.75	5.15	1.53	4.85
PAR	$P_{MSM(\alpha=0)}$, CCS time	9.25	2.17	1.07	4.28
PAR	CCS power and time	60.07	5.66	1.64	4.87

Runtimes are given in minutes.
[1] Clock period is 10 ps, as opposed to 1500 ps in other cases.

In our experiments, the simulation clock is set to 1500 ps resulting in 3000 samples per trace. The exception is made for $\Delta = 0$ simulations, where we use the 10 ps clock. The simulation times are dictated by the total number of events. More complex models cause more different propagation delays, resulting in more glitches and different toggling times. This trend holds for both logic simulation, `LSIM`, and power simulation `PSIM`, with one exception. MSM simulation at PAR stage using CCS timing models produces more events (approximately 1.3 billion) compared to its GLN counterpart (approximately 1.2 billion). Also, PAR and GLN netlists differ in only a single (clock buffer) gate inserted during physical synthesis. Without looking at the implementation of the simulator, we can not give a certain reason for this discrepancy. One possible reason might be the way the extracted SDF data is presented. At the GLN stage, statistical wire load models are written to SDF as interconnect delays. At the PAR stage, wire delays are extracted from the layout and back annotated to the cell delays. Hence this subtle difference may lead to fewer instructions during simulation, causing faster runtime in the PAR stage. Runtimes of parsers and analyzers depend on the number of samples and the number of events they have to process. The former dependency is easily observable in the $\Delta = 0$ example. The latter is observable in the increasing runtimes with the increased complexity of models.

The CCS timing is characterized using the 1 ps precision. Therefore, all

Table 3.5: Runtimes for the acquisition and processing of 1 million PAR traces.

Target circuit	PRES. S-Box	PRES. S-Layer	BP AES S-Box	AES-128
Area [kGE]	0.35	2.98	5.45	127.18
Period [ps]	3000	3600	25000	30000
Logic simul. [h]	0.01	0.15	0.39	25.81
Logic parse [h]	<0.01	0.17	0.33	2.58
Power simul. [h]	0.06	0.31	1.13	52.61
Power parse [h]	<0.01	0.05	0.07	1.17
1^{st}-order eval. [h]	<0.01	0.02	0.05	0.14
2^{nd}-order eval. [h]	<0.01	0.04	0.16	0.25

calculations within the simulator are performed with that precision internally. Lowering the simulation precision, *e.g.* to 10 ps, does not yield significant performance increases. However, this causes fewer data output by the simulator and lowers the storage costs as well as the runtimes of parsers and analyzers.

We perform both the first- and second-order TVLA on-the-fly using the approach of Schneider and Moradi [132]. Fast computation strategy put forward by Reparaz *et al.* [124] uses kernel-based estimations of the t-statistic. Instead of the floating point arithmetic needed for adding each trace on-the-fly, this approach counts the occurrence of sample values in the form of histograms. It is very effective when working with modern oscilloscopes as they provide between 8 and 12 bits of resolution. Consequently, they require storing between $2^{2 \cdot 8}$ and $2^{2 \cdot 12}$ histograms, per sample, per set, to fully represent the measurement; assuming 32-bit counter values. Simulations produce the single precision floating point traces. Applying this approach would require storing $2^{2 \cdot 32}$ histograms instead. Simulated results can be quantized down to the 8-, or 12-bit range to allow the latter approach. Nevertheless, we believe the speedup is not worth the loss of precision.

To test the scalability of CASCADE, we apply it to a fully unrolled implementation of AES-128. We use a placed and routed design of 127.18 kGE with extracted layout parasitics. Table 3.5 shows the average runtimes for simulating, parsing, and analyzing 1 million traces at the PAR stage. We do so for the unrolled AES-128, along with the other circuits evaluated in this work.

Note that the logic simulation and logic parsing are necessary for the P_{MSM} power estimation. The logic simulation is the precursor for the power simulation using PT.

With the increase of circuit complexity, *i.e.* area, the cost of simulations becomes predominant. Simulations are done using sophisticated CCS models with a

precision of 1 ps, at the post-layout stage that includes extracted parasitic elements. With this level of detail they are akin to the "golden sign-off" simulations for the timing closure. The size of the unrolled AES-128 exceeds security-dedicated area budgets of many embedded devices. Still, single thread operation on a relatively modest desktop workstation can perform the $P_{MSM(\alpha=0)}$ evaluation in less than 30 hours. Additional 55 five hours is needed for the detailed CCS power evaluation. Additionally, simulation can be split into batches and run in parallel with negligible overhead. As all tools involved have a low RAM footprint, modern workstations—featuring 32 or more cores—can complete the entire evaluation in a few hours. Batching can also facilitate earlier estimates and alleviate storage issues. As shown in Section 3.6, the leakage can be detected much before all frames are analyzed. Therefore, in practice only a fraction of the batches may need to be processed if a flaw is present. Each batch is processed in the same manner as a whole simulation would be, updating the analyzer's context. As all computations are performed on the fly there is no need for storing terabytes of simulated data dumped by the logic simulators.

Therefore, the computational complexity and the scalability can not be an obstacle towards the adoption of this approach.

3.8 Conclusions

In this chapter we present SCA-aware extensions to the standard-cell ASIC design flow. We adhere to the iterative and systematic use of simulations across different design stages in the flow. We demonstrate how different models from the SCA and EDA communities can be used to detect SCA leakage starting from the earliest design stages. To do so effectively, we review the models, methods and metrics available in the SCA literature with those of the EDA world. Thus, we introduce the use of CCS models for the SCA evaluation.

To bridge the gap between the two large communities we design and implement CASCADE, a comprehensive framework for design-time evaluation of SCA security. CASCADE is built on the state of the art EDA tools and SCA evaluation and methodologies, combining them in a methodical and automated manner. We show how it can be applied in the early design stages regardless of the type of SCA countermeasure, as long as it is based on standard-cells. We benchmark the performance of selected modules in our framework to show its aptitude in testing realistic cryptographic designs, and argue its feasibility for real-world use even when relying on a single desktop workstation. Lastly, a snapshot of CASCADE has been released in the form of open-source software, available to the research community.

Chapter 4

Evaluating Glitch-Resistant Masking Schemes Against Fault Sensitivity Analysis

Content Source

This chapter is largely based on material published in:

V. Arribas, T. De Cnudde and **D. Šijačić**, "Glitch-Resistant Masking Schemes as Countermeasure Against Fault Sensitivity Analysis," 2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), Amsterdam, 2018, pp. 27-34.

Contribution: One of the main authors; designed and conducted the ASIC-simulation-based approach for evaluation.

In this chapter we use CASCADE to evaluate glitch-resistant masking schemes, such as threshold implementations, against the Fault Sensitivity Analysis (FSA). Figure 4.1 depicts the design stages of the standard-cell ASIC design flow applied for this evaluation. We denote parts of the flow irrelevant for this chapter using grey color.

We start from theoretical considerations based on the properties of the masking schemes and FSA introduced by Li *et al.* [88]. We then proceed to perform experiments using highly detailed CCS models of the logic gates and the discuss

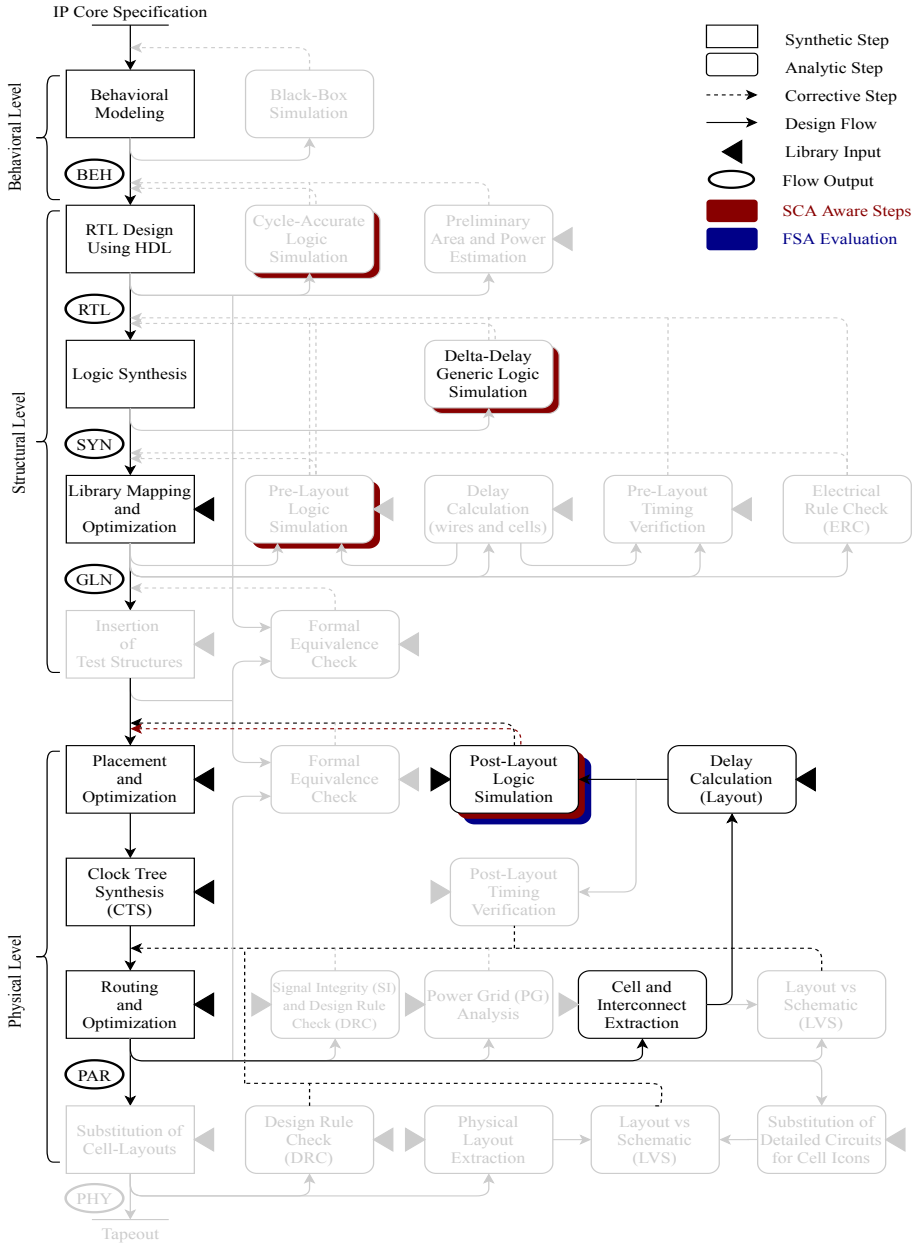


Figure 4.1: FSA in the ASIC design flow.

the practicality of said approach. We aim to give the attacker a high advantage of accessing detailed backend information. Hence we perform the analysis on top of event-driven logic simulation denoted with the blue color. Naturally, similar analysis can be performed in the earlier design stages. However, we prefer to capture as much physical detail to show the resistance against FSA provided by the glitch-resistant masking schemes.

Unlike the countermeasure implementation design-time evaluation performed in the previous chapter, we evaluate the countermeasure itself at design-time. We show how the extensive physical information provided by the CCS models can be used to evaluate a countermeasure. Another principal difference compared to the fully passive countermeasures is that there are no downsides to the 1 ps precision of CCS models. Namely, actively triggered and fully active attacks rely on the attacker's intervention. In practice it is not so easy to inject faults with high precision, therefore simulation allows evaluators to create worst-case scenarios, *i.e.* best-case for the attackers. Lastly, as active attacks often depend on the propagation of injected faults to the outputs, cycle-accurate simulation often suffices. However, in this particular attack, adversaries are trying to measure the data-dependent propagation delay within a clock cycle. For such physical observations, high-precision hardware models are invaluable as in addition to the side-channel security evaluation they allow deeper insights in certain aspects of the countermeasure.

In parallel to the reasoning behind the physical modeling and experiments targeting the ASIC platform, our co-authors have carried out experiments on an FPGA platform in parallel to strengthen the significance of the findings.

Note that the results of this paper hold true for the original FSA introduced by Li *et al.* [88]. Two years following the publication of this work at FDTC 2018, a new more powerful attack akin to FSA was put forward by Delvaux [38]. We address this finding in Section 4.8.

4.1 Motivation

We discuss masking schemes as a popular countermeasure against passive SCA in Section 2.3.2. They provide SCA resistance by randomizing the intermediate computation steps, in order to decorrelate them from the side-channel measurement. Alternatively, Fault Analysis (FA) [25] can be used to recover secrets from cryptographic devices. Through a physical manipulation, FA attackers disrupt the operation of the target implementation in a controlled manner. Thus they drive the devices to compute faulty ciphertexts. In the next step, FA attackers can infer the sensitive information by the cryptanalytic

comparison and processing of the faulty and the correct outputs. A notable example is Differential Fault Analysis (DFA) [15].

Li *et al.* [88] introduce a hybrid attack called FSA. FSA is an actively triggered passive side-channel attack. The attacker injects faults, not to obtain faulty ciphertexts, but only to detect when the target starts malfunctioning. To do so, an attacker gradually increases the fault intensity, instead of using a fixed one like in DFA. Regardless of the fault injection mechanism, *e.g.* clock-glitching or supply-voltage manipulation, FSA attackers aim to reduce the clock period so they can detect when does the circuit start malfunctioning. Effectively, FSA side-channel is the propagation-delay for each data input. Therefore, it requires a more sophisticated fault injection setup compared to the classical fault attacks such as DFA. However, since FSA does not require the faulty ciphertexts it can bypass fault-detection countermeasures that issue an alarm and prevent output of the faulty ciphertexts. Several authors [107, 106, 101, 130] have used FSA to circumvent the traditional FA countermeasures. This is not surprising as the FA attackers inject faults in order to perform cryptanalysis on the faulty outputs. However, the FSA attackers inject faults to reveal a physical side-channel, *i.e.* the data-dependent propagation delay. Therefore, SCA countermeasures may provide better protection against FSA.

4.2 Related Work

Li *et al.* [88] also argue for the use of SCA countermeasures against FSA. In particular they suggest that masking schemes could provide resistance against FSA based on the randomization of the intermediate variables. Moradi *et al.* [107, 106], as well as Mischke *et al.* [101] debunk this assumption by successfully breaking several masked AES implementations. Consequently, a few dedicated countermeasures against FSA emerged.

Ghalaty *et al.* [55] propose a gate-level approach based on the propagation-delay balancing. Li *et al.* [87] suggest another countermeasures that can be applied at the RTL stage. Namely, they gate every register input, allowing the outputs of the combinatorial logic to reach the register-input pins only after a fixed time, thus blinding all the data-dependent propagation delays with the arrival time of the gate-enable. Endo *et al.* [44, 45] give a method to tune the gate-enable timing after chip manufacturing, to ensure protection against FSA. Both of these countermeasures have negligible circuit-complexity and performance overhead compared to masked implementations. However, neither of these countermeasures provide any resistance against SCA. As prospecting adversaries tend to search for the weakest link, it stands to reason to provide

protection against combined attacks. To this end Schneider *et al.* [133] and De Cnudde and Nikova [31] propose combined protection strategies based on Boolean masking schemes.

Lastly, Moradi *et al.* [107, 106], seemingly show that glitch-resistant threshold implementations provide no resistance against FSA. In particular, they attack a core named `AES_TI`. However, this masked implementation fulfills only one of the three constituting properties of the threshold implementations, namely correctness. Therefore, no security properties of the threshold implementations extend to this implementation.

4.3 Contributions

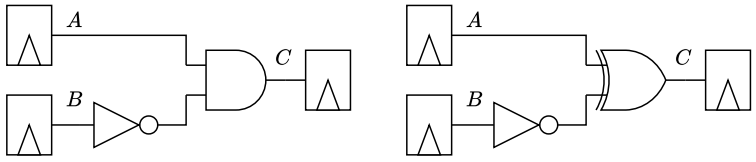
The resistance of glitch-resistant masking schemes, such as threshold implementations, against FSA introduced by Li *et al.* [88] remains to be determined. To this end, we first layout theoretical considerations and physical assumptions about the underlying hardware to answer this question. We focus on threshold implementations in particular. As they are likely to be deployed as a SCA countermeasure, they have the potential to increase the level of FSA resistance without further costs.

Inline with our considerations and assumptions, we create an experimental setup based on the detailed CCS timing simulation in the backend design stages. We argue why such a simulated setup is the best-case scenario for the attackers. Using this setup, we attack two representative cryptographic S-Boxes, namely PRESENT and KECCAK, in their protected and unprotected form. Unrealistically high precision of the simulated FSA traces allows unprotected implementations to be broken with ease. However, our experiments show that the protected versions of both S-Boxes resist the FSA.

The remainder of this chapter is organized as follows. In Section 4.4 we describe the fault sensitivity analysis as introduced by Li *et al.* [88]. In Section 4.5 we argue how glitch-resistant masking schemes, such as threshold implementations, can be used to mitigate FSA. Next, we present our experimental setup, the reasoning behind it and demonstrate the results attacks on two state-of-the-art S-Boxes in Section 4.6. We conclude our findings in Section 4.7. Lastly, we briefly address the attack put forward by Delvaux [38].

4.4 Fault Sensitivity Analysis

Assume that an attacker injects a fault with Fault Intensity (FI). Fault Sensitivity (FS) is the intensity of the injected fault FI, at which the device starts outputting incorrect results. Reversely, circuit operates correctly as long as FI < FS. FSA as introduced by Li *et al.* [88] correlates the input data values with their fault sensitivity. Faults can be injected using, *e.g.* by lowering the supply voltage, by introducing glitches in the clock signal and so on. In any case, attackers aim to reduce the time allocated for signal propagation. As propagation delays in CMOS circuits depend on the input data, FS is data dependent too. This dependency constitutes the actively triggered side-channel. Figure 4.2 illustrates this dependency.



(a) AND gate propagation. (b) XOR gate propagation.

Figure 4.2: Variable data propagation delay.

For the AND gates, Figure 4.2a, if $A = 0$ output $C = 0$ is set immediately, regardless of the B value. However, if $A = 1$, signal B has to propagate through the inverter first. Therefore, propagation delay t_d can be described using Equation (4.1). Similarly, in case of OR gates, active value $A = 1$ would yield the invariant behavior of the input B. In case of XOR gates, depicted in Figure 4.2b, propagation delay is data independent and always equal to $t_d = t_d^{NOT} + t_d^{XOR}$.

$$t_d = \begin{cases} t_d^{AND} \text{ , if } A = 0, \\ t_d^{NOT} + t_d^{AND} \text{ , otherwise.} \end{cases} \tag{4.1}$$

4.4.1 Attack phases

FSA is divided into two phases. The first is the profiling phase, as delineated in Algorithm 4. It entails collection of fault sensitivities for a set of input plaintexts.

Algorithm 4 Profiling phase of the FSA.

Input: Encryption algorithm $E_k(pt, fi)$, applied to plaintext pt, perturbed using FI fi.
Input: Fault injection resolution δ.
Input: Vector $\mathcal{I}$ of N plaintexts.
Output: Vector $\mathcal{O}$ of N ciphertext ct, fault sensitivity fs tuples.

```
for i = 1 to N do
    O[i] ← (E_k(I[i], 0), 0)
    fi ← δ
    while O[i][0] = E_k(I[i], fi) do
        fi ← fi + δ
    end while
    O[i][1] = fi
end for
```

A vector of plaintexts $\mathcal{I}$ is chosen uniformly at random. For each plaintext $\mathcal{I}[i]$, the corresponding output tuple $\mathcal{O}[i]$ is initialized with the correct ciphertext $E_k(\mathcal{I}[i], 0)$, and a null FS value. Said plaintext $\mathcal{I}[i]$ is repeatedly encrypted, reseting the device before every encryption, while perturbing the design using incrementally larger fault intensity. Here, δ is the resolution at which the attacker can control the perturbation, *e.g.* the precision of the clock glitching apparatus. FI at which the encryption yields an incorrect ciphertext is stored in the output tuple as the FS for the target plaintext.

The second is the key recovery phase, as delineated in Algorithm 5. It entails correlation between the collected fault sensitivities with the predictive model.

Algorithm 5 Key recovery phase of the FSA.

Input: Modeling function $\hat{fs}(ct, \hat{k})$, applied to the ciphertext ct and the key guess $\hat{k}$.
Input: Vector $\mathcal{O}$ of N ciphertext ct, fault sensitivity fs tuples.
Output: A t-bit (sub)key $k \in \{0,1\}^t$.

```
for k̂ = 0 to 2^t − 1 do
    for i = 1 to N do
        F̂S[i] ← f̂s(O[i][0], k̂)
        FS[i] ← O[i][1]
    end for
end for
Corr[k̂] ←| ρ(F̂S, FS) |
k ← k̂ |_{Corr[k̂] ≡ max(Corr)}
```

The collected fault sensitivities fs are stored along the ct data in a set $\mathcal{O}$. Fault sensitivities stored at $\mathcal{O}[i][1]$ are correlated with the predicted values FS based on a model $\hat{fs}$. Modeling functions are typically based on HW and HD models of the targeted intermediate. Similarly like in the CPA [27], for each output tuple $\mathcal{O}[i]$ Pearson's correlation coefficient ρ is computed between the predicted FS and the measured FS $\mathcal{O}[1]$. The highest correlation then leads to the correct key [88].

Success of a FSA attack is in practice is dictated by the resolution of the fault injection δ the attacker can apply, and whether or not the selected model, *i.e.* function $D_{\hat{k}}(ct, \hat{fs})$, can be extracted. Endo *et al.* [46, 47] demonstrate the efficacy of clock-glithers. As our setup is based on ASIC simulation we use a "clock-glitcher" that outperforms any one that can be found in practice. Li *et al.* [88] rely on the HW model. Alternatively, Mischke *et al.* [101] use a zero-value attack model.

4.5 Glitch-Resistant Masking Schemes as a FSA Countermeasure

In this section we delineate the theoretical considerations based on which we claim the resistance of masking schemes that fullfil the *non-completeness* property—such as TI, DOM and Consolidated Masking Schemes (CMS)—against FSA, as introduced by Li *et al.* [88]. We base our delineation on the following assumptions:

1. we discuss protection against FSA as introduced by Li *et al.* [88], therefore the attacker does not exploit the faulty ciphertexts in any way, other than determining their correctness and the profiling phase is performed exactly as described in Algorithm 4.
2. the attacker measures the data-dependent propagation delay t_d by injecting faults;
3. each share leaks independently and masking schemes are implemented properly.

The second assumption follows directly from the way the profiling phase is conducted. After each FI increase, the attacker notes whether the computation was correct and the corresponding FI. Let the attacker note the response after injection i as tuples (Δ_i, FI_i), defined as per Equation ((4.2)).

$$(\Delta_i, FI_i) = \begin{cases} (1, i \times \delta) \ \text{, if the computation finishes correctly,} \\ (0, i \times \delta) \ \text{, otherwise.} \end{cases} \tag{4.2}$$

The measurements are repeated until the attacker receives $(\Delta_m, FI_m) = (0, m \times \delta)$. At that point attacker collects $[(\Delta_1 = 1, FI_1), (\Delta_2 = 1, FI_2), \ldots, (\Delta_m = 0, FI_m)]$ and moves on to the next data input. Therefore, the attacker determines that the critical propagation delay, *i.e.* data dependent propagation delay t_d, as per Equation ((4.3)).

$$T_{CLK} - m \times \delta < t_d \leqslant T_{CLK} - (m-1) \times \delta \ . \tag{4.3}$$

In other words, the attacker measures the t_d, with the measurement resolution δ. Practical success of attacks is sufficient to argue that sufficiently small δ is attainable in practice. Conversely, fault intensity FI_m corresponds to the fault sensitivity. If FI_m is high relative to the T_{CLK}, then the fault sensitivity is low.

The third assumption is equivalent to the independent leakage assumption. In other words, all shares of threshold implementations are implemented in parallel and the FS of each share is independent of the FS of other shares.

4.5.1 Propagation Delay of Non-Complete Shares

We demonstrate how non-complete shares resist FSA using an example of a three-share AND gate. Given the inputs $x = x_1 \oplus x_2 \oplus x_3$ and $y = y_1 \oplus y_2 \oplus y_3$ output shares z_1, z_2 and z_3 are computed as per Equation (4.4). As we make no additional assumptions other then the non-completeness, this reasoning can be generalized.

$$\begin{aligned} z_1 &= f_1(x_2, x_3, y_2, y_3) = x_2y_2 + x_2y_3 + x_3y_2 \ , \\ z_2 &= f_2(x_1, x_3, y_1, y_3) = x_3y_3 + x_3y_1 + x_1y_3 \ , \\ z_3 &= f_3(x_1, x_2, y_1, y_2) = x_1y_1 + x_1y_2 + x_2y_1 \ . \end{aligned} \tag{4.4}$$

All shares are exist in parallel. For each data input, different fault intensity FI_m causes the share with the longest propagation delay t_d to stop operating correctly. Conversely, that share has the highest fault sensitivity. Although

circuits are algebraically identical, consisting of three two-input AND gates, followed by a three input XOR gate, each path will have a different propagation delay due to clock skew on input registers, RC parasitics of the routing wires, variations in the manufacturing process, and the inherent asymmetries of CMOS gates described in the previous chapter. Additionally, sharing of more complex functions can yield asymmetrical shares, one of which may have a distinctly higher logic depth. Therefore, without the loss of generality, we assume this occurs in share f_1 and that the fault sensitivities of the remaining two shares are as per Equation ((4.5)), where d is a piece of input data.

$$FS^{f_1}(d) \geqslant FS^{f_2}(d) \geqslant FS^{f_3}(d) \iff FI_m^{f_1}(d) \leqslant FI_m^{f_2}(d) \leqslant FI_m^{f_3}(d) \ . \quad (4.5)$$

Let us analyze the data-dependency of the transitions in the share f_1. Figure 4.3 depicts the structural netlist of each share. Here $t_d^i(d)$ denotes the propagation delay of the i-th net for the input value d. Similarly, t_d^A represents the data dependent propagation delay at the output of the AND gate labeled A. Let the $t_d^i = t_d^0 + i \times \epsilon$, where ϵ is used to capture the above mentioned physical artifacts that introduce variable delay. As we can label the nets arbitrarily, we do not lose generality with such ranking.

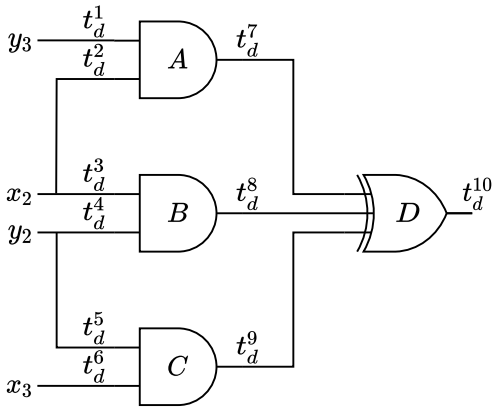


Figure 4.3: One share of the AND gate presented in Equation (4.4)

The corresponding propagation delays are as per Equation ((4.6)).

$$t_d^A(d) = \begin{cases} t_d^1 + t_d^{AND} & \text{, if } y_3 = 0 \ , \\ t_d^2 + t_d^{AND} & \text{, otherwise} \ . \end{cases}$$

$$t_d^B(d) = \begin{cases} t_d^3 + t_d^{AND} & \text{, if } x_2 = 0 \ , \\ t_d^4 + t_d^{AND} & \text{, otherwise} \ . \end{cases} \tag{4.6}$$

$$t_d^C(d) = \begin{cases} t_d^5 + t_d^{AND} & \text{, if } y_2 = 0 \ , \\ t_d^6 + t_d^{AND} & \text{, otherwise} \ . \end{cases}$$

Similarly, the propagation delay t_d for input data d at the output is given in Equation (4.7).

$$\begin{aligned} t_d(d) \equiv max(& \\ & t_d^A(y_3) + t_d^{AND}(y_3, x_2) + t_d^7 + t_d^{XOR} + t_d^{XOR}(x_2, x_3, y_2, y_3) + t_d^{10} \ , \\ & t_d^B(x_2) + t_d^{AND}(y_2, x_2) + t_d^8 + t_d^{XOR} + t_d^{XOR}(x_2, x_3, y_2, y_3) + t_d^{10} \ , \\ & t_d^C(y_2) + t_d^{AND}(y_2, x_3) + t_d^9 + t_d^{XOR} + t_d^{XOR}(x_2, x_3, y_2, y_3) + t_d^{10}) \ . \end{aligned} \tag{4.7}$$

Here, the data-dependent propagation delays of CMOS gates are annotated as output net delays. Seeing how these variations can be considered an order of magnitude smaller compared to the average gate propagation time together with the propagation across routing wires, we can get to the model used by Li *et al.* [88]. Propagation delay t_d following this simplification is shown in Equation (4.8). However, the non-completeness property ensures that no information can be inferred about the unshared variables x and y as x_1 and y_1 can not be leaked this way. Lastly, note that while we make this theoretical considerations, the simulations take all these variations into account.

$$\begin{aligned} t_d(d) \equiv max(& t_d^A(y_3) + t_d^7 + t_d^{XOR} + t_d^{10} \ , \\ & t_d^B(x_2) + t_d^8 + t_d^{XOR} + t_d^{10} \ , \\ & t_d^C(y_2) + t_d^9 + t_d^{XOR} + t_d^{10}) \ . \end{aligned} \tag{4.8}$$

Equation (4.5) does allow multiple shares to have the same fault sensitivity. In theory this allows the attacker to observe multiple shares, hence breaking the non-completeness for a particular input data d. Therefore, the provable security of threshold implementations does not apply for protection against FSA. In practice though, the likelihood of this happening for a significant number of inputs, if any, is low due to the manyfold of physical sources of variation including the added noise coming from fresh masks. Therefore, we believe that threshold implementations, and other masking schemes that feature non-completeness, provide a high degree of resistance against FSA.

Additionally, the shared AND gate from Figure 4.3 fails to satisfy the output uniformity property of threshold implementations. However, as demonstrated above, this lack of uniformity does not affect the resistance against FSA.

Lastly, depending on the time when the fault is injected relative to the clock edge, *i.e.* $T_{CLK} - i \times \delta$, hold times of output registers can be violated causing metastable behavior. Consequently, the output register bit affected by the error could be flipped back to the correct value, or another output register bit could be affected. Similarly, as electronic circuits inevitably suffer from noise that follows the Gaussian distribution [120], multi-σ outliers can disrupt this ordering. This is observed and exploited by the novel attack of Delvaux [38]. However, in the case of the original FSA attacked introduced by Li *et al.* [88] we do not see how these events could impact the security. If anything, in case one such occurrence alters the ordering from Equation (4.5) it would give the attacker an inaccurate measurement of the t_d, and the attacker who does not exploit the ciphertext could still not determine which in which share does the fault occur.

4.6 Experiments

We experimentally validate our theoretical considerations by simulating the FSA attack introduced by Li *et al.* [88]. We use backend simulations to attain the highest level of physical detail based on the CCS models. Thus we obtain perfectly aligned and noiseless observations. Moreover, simulations allow us the FI increment $\delta = 1\,\text{ps}$, *i.e.* propagation-delay measurement resolution, without any jitter. Such a setup exceeds technical capabilities of modern day humans, allowing us to observe even the smallest variations—like the ones caused by the pin-dependent asymmetries of standard-cells. Lastly, our simulations include only the target circuits, hence no surrounding circuitry can interfere with the measurements. We believe that such measurements are unattainable in practice.

ASIC experiments. We synthesize, place and route designs using a 45 nm open-source standard-cell library from NanGate [78]. It relies on state-of-the-art CCS models. CCS timing models capture even the smallest data-dependent asymmetries of standard-cells, as discussed in Chapter 3. Using the worst-case process corner, we maximally enhance these differences. We use Synopsys Design Compiler for logic synthesis; Cadence Innovus for placement, routing and RC extraction; and MentorGraphics QuestaSim for logic simulation. Design flow automation and data processing is done using the CASCADE [153] framework.

FPGA experiments. We use Spartan-6 XCLX75-2CG484 as the target platform and the simulator integrated in the Xilinx ISE 14.7 to perform the simulations. Although the exact methodology of delay calculation for the post-layout FPGA design is not available to us, as the manufacturers trade-secret, we believe it is designed to provide designers with the estimates of the comparable accuracy as the ASIC models. Moreover, as FPGAs are already fabricated devices with a finite number of configurations, and their entire design is "under one roof" of the manufacturer, we see no reason why these models would be inferior to ASIC ones. All of the FPGA experiments are the work of our co-authors. As this dissertation focuses on ASIC design, we remove them from further considerations. We note that all results are consistent across both platforms and kindly refer the interested reader to the original publication [156].

4.6.1 Profiling Phase

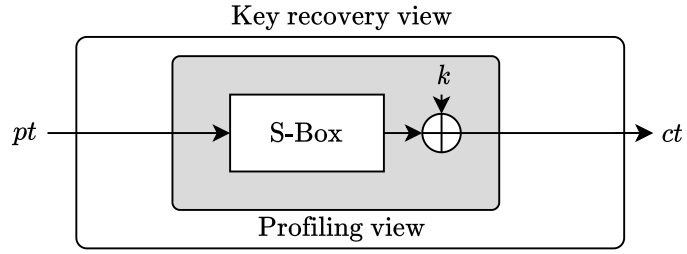
Li *et al.* [88] always reset the input to an all-zero bit vector. Therefore, they have to profile 2^n different input values for an n-bit circuit. We leverage the speed and precision of simulations, as well as the size of our test circuits to consider all input $2^{2 \cdot n}$ input transitions. Thus we obtain 2^n different profiles, one for each reset possible value. This allows us to select the profile with the highest correlation, hence further maximizing the adversaries advantage.

4.6.2 Key Recovery Phase

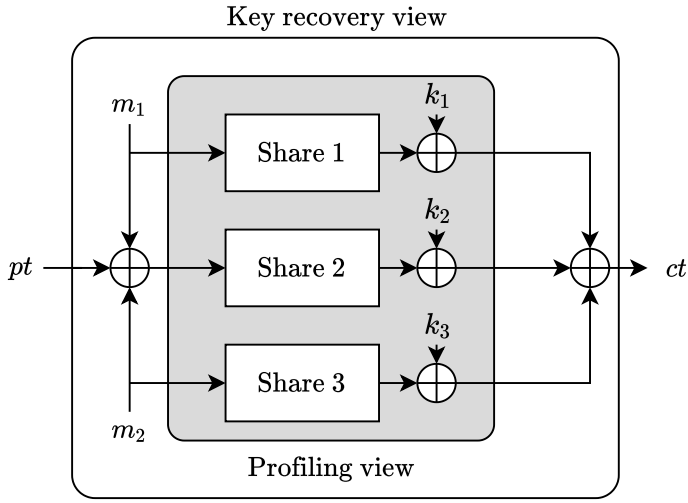
We target a dummy cipher round consisting of an S-Box and the key addition. We attack the circuit at the inputs using the HW-based prediction function $\hat{fs}$ described in Equation (4.9). Here, ct stands for the correct ciphertext value and $\hat{k}$ for a key guess.

$$\hat{fs} = \mathrm{HW}(\text{S-Box}^{-1}(ct \oplus \hat{k})) \; . \tag{4.9}$$

Figure 4.4 depicts the principal architecture of the dummy cipher round. Architecture for the unprotected S-Box is shown in Figure 4.4a, and of the three-shared masked one in Figure 4.4b. We distinguish the "profiling view" (grey) and the "key recovery view" (white). For the unprotected implementation they are the same. For the protected implementation, we allow the attacker to gain a more detailed profile based on the unshared inputs. Profiling view is unattainable to non-invasive adversaries.



(a) Unprotected S-Box.



(b) Protected S-Box.

Figure 4.4: Target circuit.

Therefore, we give the attackers another unrealistically strong advantage by allowing them to create profile with the strongest correlation. Moreover, the profiling view removes the "jitter" in the propagation delay introduced by the generation of random masks. Lastly, we allow the attacker full control over the input values, including the selection of a reset value. In practice, such access

would be restricted by the device interface. Therefore, the target circuit can only rely on its non-completeness and uniformity to resist FSA.

In summary, our experimental setup gives multiple unrealistically strong advantages to the FSA attackers. The advantages of the simulated approach, combined with the intimate view of the unshared values and the control over data inputs form a best-case scenario for the attackers. We now move to show how glitch-resistant masking schemes hold against the FSA under these assumptions.

4.6.3 Present S-Box

Our first target is the 4-bit S-Box of the lightweight ISO blockcipher standard PRESENT [24]. It is a $\{0,1\}^4 \to \{0,1\}^4$ non-linear mapping. In particular we use the masked design of Poschmann *et al.* [117]. This design is decomposed into two quadratic S-Boxes F and G forming a pipelined implementation. As previously discussed, FSA introduced by Li *et al.* [88] targets the longest propagation delay. For simplicity, we target just one of the two stages to gather the profiles, choosing the F stage for our experiments. Note that as each of the decomposed stages constitutes a threshold implementation we could observe only one of them without the loss of generality. PRESENT S-Box has four input bits, hence $2^{2\cdot4}$ possible input transitions. Three shares of the threshold implementations yield a total of $2^{2\cdot12}$ transitions.

Profiling phase and the optimal reset value.

Typically FSA attackers use the all-zero input vector as the reset value between two evaluations. We show that this is not always the best approach. Firstly, we plot the correlation values between the HW of the data input and the FS for different reset values in Figure 4.5. For the unprotected ASIC implementation reset value `0xC` results in a clear corelation peak of $\rho = 0.75$. Hence, using this reset value in the to collect traces gives the best results in the key recovery phase. For the protected implementation the shared reset value `0x789` yields the maximal correlation peak of $\rho = 0.41$. Note that as attackers can not control the shared value in practice. Allowing the optimal reset values is another advantage we gift to the attackers.

Furthermore, we note that using the HD model instead of the HW decreases the correlation. On the contrary, HD model is preferred in the context of classical SCA on hardware implementations. We delineate this difference on the example of a two-input AND gate. For the classical side-channels, such

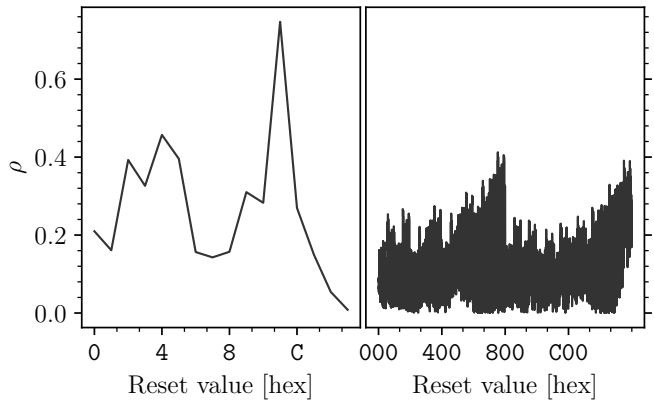


Figure 4.5: PRESENT S-Box, correlations for different reset values, unprotected (left), protected (right).

as the instantaneous power consumption, input transitions `0b00` → `0b11` and `0b11` → `0b00` both result in an output toggle. Therefore, on average their side-channel emanation varies little. However, the second transition has a lower propagation delay on average, determined by the first zero-value arriving. Therefore, HW-based predictions outperform the HD-based ones in the FSA setting.

Additionally, Figure 4.6 illustrates the advantage of the optimal reset value, compared to considering all possible input transitions. Counterintuitively, a smaller number of traces gives a consistently better correlation.

Key recovery phase.

Figure 4.7 illustrates the key recovery attacks on the unprotected and protected implementations using the optimal profiles. Despite all the advantages, protected implementation shows barely any correlation. The results hold for all the key values. PRESENT S-Box protected using a full-fledged threshold scheme exhibits virtually no correlation, testifying to the resistance against FSA.

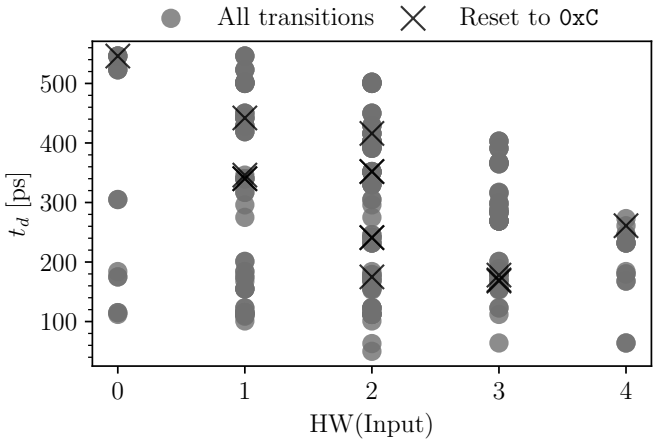


Figure 4.6: Unprotected PRESENT S-Box, data-dependent propagation delay t_d in the function of the input HW.

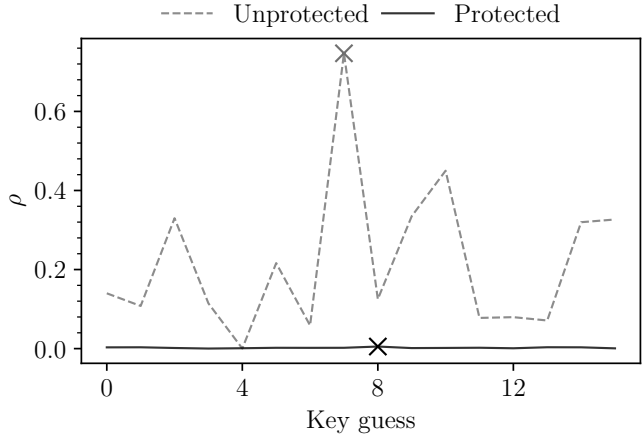


Figure 4.7: PRESENT S-Box, FSA recovery for the key value 7; $\times$ indicates $\hat{k}$.

4.6.4 Keccak S-Box

We now move to evaluate the 5-bit nonlinear KECCAK χ permutation, *i.e.* the KECCAK S-Box. We use the threshold implementation approach by Daemen [33], also known as changing of the guards. It is a three-share implementation that consumes four random bits per evaluation. Random bits are needed to achieve uniformity. By setting the random bits to an all-zero value, we can evaluate the

resistance of glitch-resistant masking schemes reduced only to non-completeness and correctness. Compared to the previous experiment, we do not exhaust all of the $2^{2 \cdot 3 \cdot 5}$ input transitions. Instead we randomly choose $2^{2 \cdot 12}$ input transitions to match the previous experiment in volume. We repeat the experiments using different pools of randomness and obtain consistent results.

Profiling phase and the optimal reset value.

Similarly to the PRESENT S-Box, we plot the correlation values between the HW of the data input and the FS for different reset values in Figure 4.8. For the unprotected ASIC implementation reset value `0x9` results in the highest corelation of $\rho = 0.63$. Hence, using this reset value in the to collect traces gives the best results in the key recovery phase. However, this circuit has two more reset values of comparable merit. For the protected implementation the shared reset value `0x755f` yields the maximal correlation peak of $\rho = 0.49$.

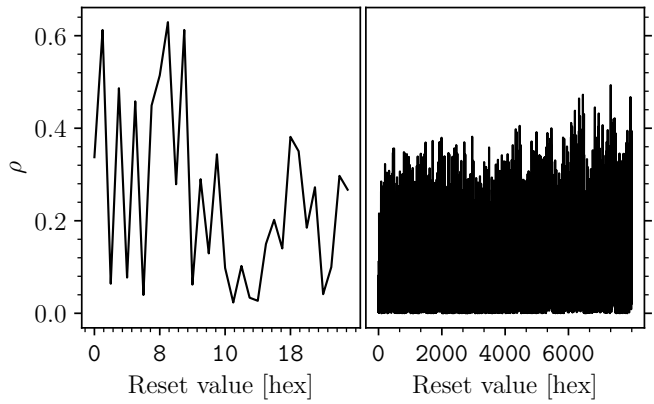


Figure 4.8: KECCAK S-Box, correlations for different reset values, unprotected (left), protected (right).

Figure 4.9 confirms the previously-discussed advantage of the optimal reset value, compared to considering all possible input transitions.

Key recovery phase.

Figure 4.10 shows the result of the key recovery phase. Despite the lack of uniformity the advantageous attacker is as oblivious to the key value, as if a

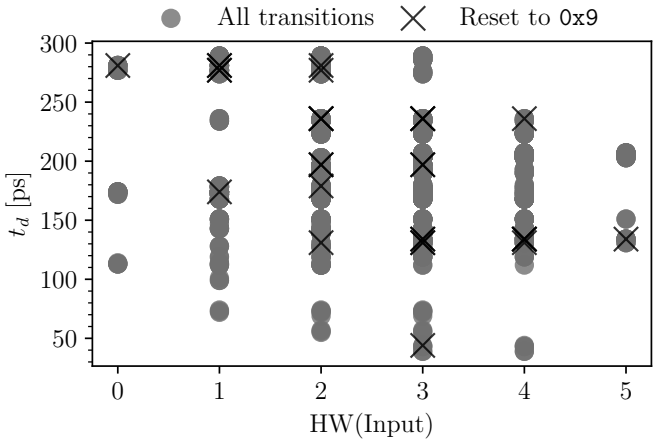


Figure 4.9: Unprotected PRESENT S-Box, data-dependent propagation delay t_d in the function of the input HW.

full-fledged threshold implementation were employed.

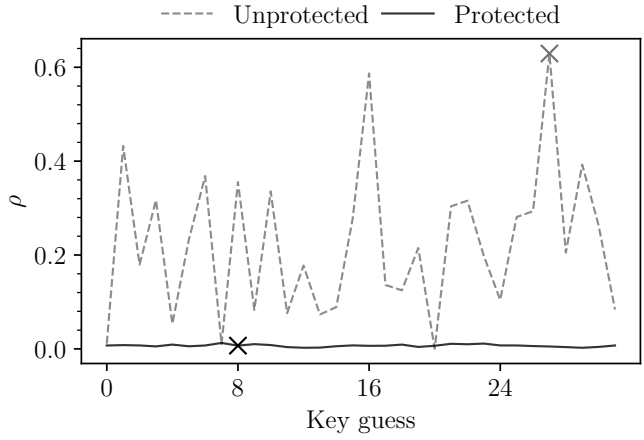


Figure 4.10: KECCAK S-Box, FSA recovery for the key value 27; $\times$ indicates $\hat{k}$.

4.7 Conclusions

In this chapter we investigate the resistance of glitch-resistant masking schemes against fault sensitivity analysis, actively triggered passive side-channel attack, introduced by Li *et al.* [88]. We present our theoretical reasoning and experimental setup based on ASIC hardware simulation. All simulations are based on detailed post-layout delay calculation that takes into account many intricate asymmetries of CMOS hardware. Under a set of assumptions about the underlying hardware we experimentally show how glitch-resistant schemes that satisfy the non-completeness property resist the aforementioned attack.

To fortify our findings, we perform experiments under conditions that strongly favor the attacker. Amongst other advantages given to the attackers, we allow them to choose the optimal profile, based on the reset value. This consideration in itself has not been previously addressed in the literature. However this is understandable as from the perspective of attacking larger designs it is infeasible to exhaust all input transitions in search of the optimal reset value.

We focus on two instances of threshold implementations in our experiments. However, we believe that any glitch-resistant masking schemes that fulfills the non-completeness provides the same level of resistance against FSA introduced by Li *et al.* [88]. Lastly, our analysis shines light onto why Moradi *et al.* [107, 106] succeeded in breaking the masked AES implementations, as they were actually lacking the non-completeness property and therefore were insecure in the presence of glitches.

4.8 Follow-up Work

In a recent follow-up of this work, Delvaux [38] introduces a more powerful form of FSA. The original FSA is based on measuring data-dependent propagation delays, by means of injecting faults of increased intensity. We discuss this while explaining Equation (4.3). Figure 4.11 depicts how FI relates to the FS. Namely, for each transition FS is denoted using the horizontal dashed line. It is the "critical FI", as the circuits stops operating for FI $>$ FS. Under such profiling phase our theoretical considerations and experimental results hold.

However, Delvaux mounts the attack as shown in Figure 4.12. Namely, the author applies the same FI value across different transitions. This simple change allows multiple shares to be faulted, thus breaking the non-completeness. Furthermore, Delvaux uses the fixed FI value, depicted using the horizontal dashed line, as a distinguisher and relies on ciphertext exploitation akin to the

DFA. As glitch-resistant masking schemes are not designed to withstand fault attacks, it stands to reason that such an attack would work. However, we do not consider this to be the same attack as the FSA introduced by Li *et al.* [88]; as shown in the two figures.

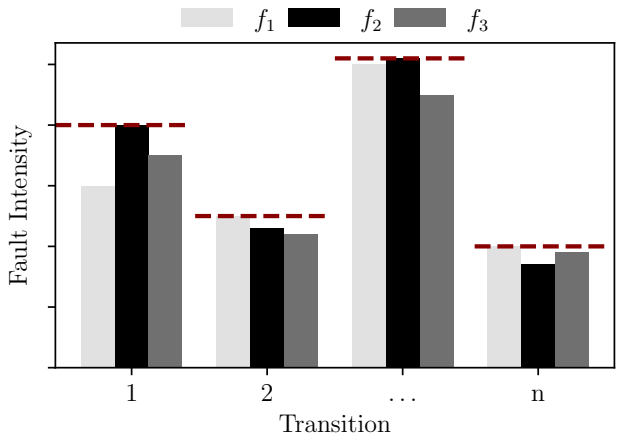


Figure 4.11: Measuring propagation delays of different shares via FI.

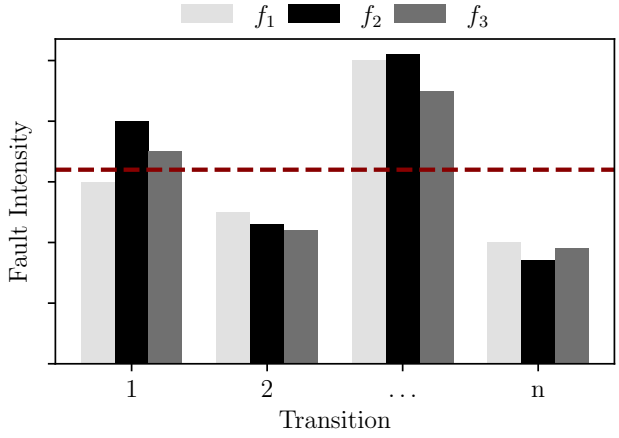


Figure 4.12: Fixed FI as a distinguisher by Delvaux [38].

Chapter 5

Investigating the Impact of Layout Parasitics on Masked Circuits

> Inspiration is for amateurs—the rest of us just show up and get to work.
>
> Charles Thomas Close

Content Source

This chapter is largely based on material published in:

Šijačić, D., Balasch, J., and Verbauwhede, I. Sweeping for leakage in masked circuit layouts. In 2020 Design, Automation Test in Europe Conference Exhibition (DATE) (2020), pp. 915–920.

Contribution: Principal author.

We denote the steps of the standard-cell ASIC design flow affected by the work in this chapter using green arrows in Figure 5.1. Methods in this chapter do not rely on the gate-level simulation, characteristic for this flow. Instead, we improve the gate-level modeling for SCA purposes. We denote the remaining parts of the flow using grey color.

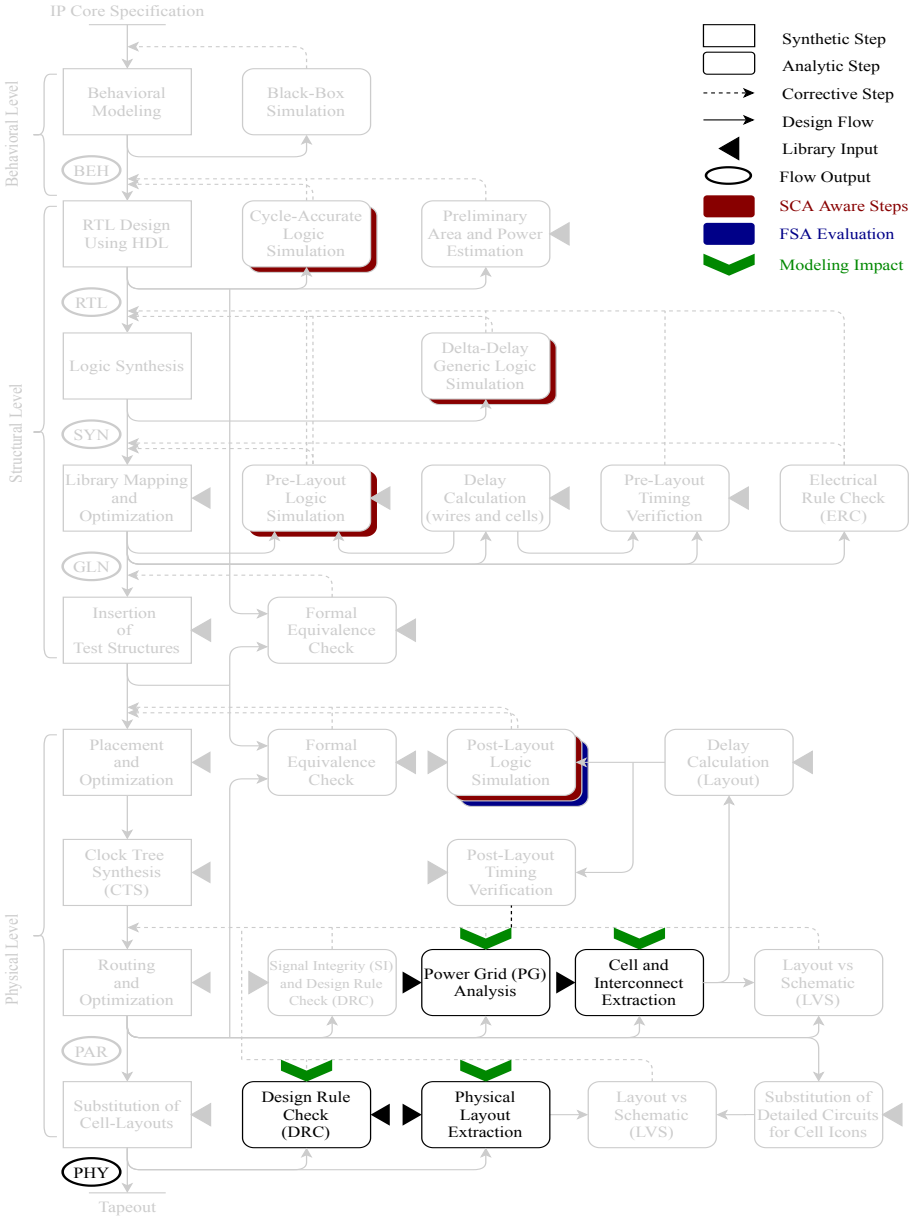


Figure 5.1: Steps of the standard-cell ASIC design flow affected by SCA-dedicated layout consideration.

In this chapter we move from the standard-cell logic gate simulations to the underlying level of analog Simulation Program with Integrated Circuit Emphasis (SPICE) simulations. We do so to uncover the effects of physical phenomena that may pass unnoticed as a consequence of performance-driven modeling for digital circuits, yet that impact SCA security. We aim to gain detailed insights into these effects, caused by the parasitic elements that naturally occur in layouts, and to improve their modeling such that they can be captured in the standard-cell ASIC design flow. Figure 5.1 emphasizes directly influenced backend stages. However, as the extraction of parasitic elements plays crucial role in delay calculation, as well as in power distribution network and signal integrity assessments, this work may indirectly influence the entire placement and routing stage of the standard-cell design flow. While this is a preliminary investigation, the ultimate goal of such analysis would be the distillation of design constraints and placement and routing rules dedicated to SCA security.

5.1 Motivation

Provable security claimed by masking schemes is underlined by different assumptions regarding hardware and logic behavior. Focusing on Boolean masking schemes we introduce in Section 2.3.2, sharing an n-bit variable x into N shares entails randomly generating x_1, x_2, $\ldots x_{N-1}$ and computing x_N such that Equation (5.1) holds. Thus obtained variables x_i are used as inputs to the N randomized shares, implemented as concurrent blocks in hardware.

$$x = x_1 \oplus x_2 \oplus \ldots \oplus x_N \ . \tag{5.1}$$

While the exact sets of assumptions vary among masking schemes, the *independent leakage assumption* is inevitable. It states that information leakage is a linear combination of leakages of the individual shares. For a hardware implementation of an arbitrary shared computation, the independent leakage assumption can be represented with the circuit depicted in Figure 5.2. All shares are connected in parallel to an ideal voltage source, capable of providing infinite supply current to each share.

As masking schemes do not make assumptions on the implementation of individual shares, they can be realized using standard-cell ASIC libraries or on FPGAs. We focus on ASIC implementations. On the front-end of the design cycle, using gate-level modeling, the abstraction shown on Figure 5.2 holds. In the back-end stages, said *logic cells* are placed on the same substrate, along with different *physical cells* such as filler cells, a clock distribution network and a Power Distribution Network (PDN). Logic cells are then interconnected

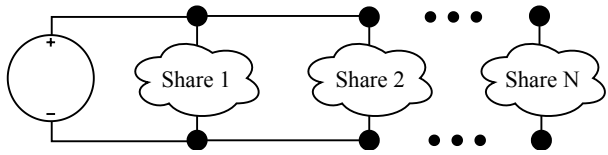


Figure 5.2: Circuit model for the independent leakage assumption.

using multiple metal layers in a process called routing. Consequently, a myriad of parasitic elements and non-linear effects emerge that are not considered in models used for masking. We aim to uncover whether any of said phenomena can violate the independent leakage assumption. To this end, we consider analog simulations on a wholesome circuit model that incorporates the effects due to:

a) *PDN*, an analog circuit distributing a low-frequency signal across a wide area. It employs supply buffers to achieve sufficient current capacity demanded by the digital logic core. Large parasitic resistances and inductances of long PDN wires can cause *ground-bounce* (increase of the low power rail from the reference) and *supply sag* (drop in the high power rail below the nominal value) [69]. As chip-ground is often realized as a large metal plate, each node of the PDN additionally forms a significant capacitance towards ground.

b) *logic core*, the circuit's functionality composed by numerous tightly placed digital cells. Relatively short wires and library designers' efforts to avoid oscillations leave no significant parasitic inductances. As each share is performing self-contained computations, there are no wires—hence no parasitic resistances—between them. Parasitic capacitances, however, can electrically couple the shares via crosstalk. Thus, capacitances between routing wires carrying data-dependent, high-frequency signals from different shares remain threatening to the independent leakage assumption.

5.2 Related Work

The usual suspect for SCA leakage due to back-end phenomena are parasitic capacitances. Recent work by De Cnudde *et al.* [30] for the first time practically demonstrates information leakage on a masking scheme caused by placement and routing. The authors implement a masking scheme on an FPGA platform and compare two placements: unconstrained (shares placed closely together)

versus constrained (all shares are placed far apart). Using Test Vector Leakage Assessment (TVLA) [59, 32] they show that unconstrained placement leads to lower SCA security. Nevertheless the authors are unable to pinpoint the source of observed leakage. Instead they suspect crosstalk and the resistive (IR) supply voltage drop as causes of the decreased SCA security. As they target an FPGA implementation, where the device layout is a closely guarded secret of the manufacturer, the actual causes of increased leakage cannot be further investigated. Other works that address sources of coupling include [41, 29, 151] Dyrkolbotn *et al.* [41] rely on capacitive crosstalk to observe double the leakage of the Hamming Distance (HD) model on an 8-bit bus. Chen *et al.* [29] discuss the effects of glitches and coupling through inter-wire capacitances. Zussa *et al.* [151] show that capacitive coupling between logically independent blocks can be the source of information leakage.

Analog SPICE transistor models yield the most accurate representation of ASIC designs. Unfortunately, the number of transistors in digital circuits is prohibitively large for analog simulations to scale efficiently. Designers can instead resort to gate-level simulators, that use piece-wise linear models extracted from complex transistor-level simulations. Such models are enabled by the high robustness of the CMOS logic style: large parasitic components, may cause significantly different analog waveforms without altering the digital behavior. Consequently, highly efficient gate-level models are obtained at the price of overlooking the analog deviations. Such mismatch in modeling and toolchains can however lead to serious misrepresentation of SCA security in simulation. Tiri and Verbauwhede *et al.* [143] show how different RC extraction methods can lead to greatly different security evaluations using SPICE. Lastly, Šijačić *et al.* [155] show there is little distinction between the wire-load models of the pre-layout netlist versus the extracted RC parasitics from the layout, using state-of-the-art digital simulators.

Nevertheless, no concrete results quantifying the impact of layout effects on SCA security can be found. Crosstalk through parasitic capacitances has been identified to deteriorate security in several cases, but this remain a qualitative observation. In terms of other design parameters, such as energy consumption, it has been quantified to inevitably increases the energy consumption and couples the supply current of logically independent components, as demonstrated by Moll *et al.* [102]. On the other hand, baring the work of De Cnudde *et al.* [30] considerations of the PDN as the source of information leakage remains unaddressed to the best of our knowledge.

5.3 Contributions

We propose a SPICE model for the co-simulation of the analog PDN with the digital logic core. It accounts for the finite current capacity power supply buffers, parasitic resistors, inductors and capacitors that inevitably occur in the long lines of the PDN, along with the parasitic capacitors of the logic core. We use this model to investigate the gap between digital and analog modeling for SCA security by analyzing representative masked gates devised to provide first-order security. By means of experiments, we determine the impact of all parasitic elements that may exist in a masked circuit, while allowing it to operate correctly. Thus, we provide a deeper insight into the sources of leakage in masked designs.

The rest of this chapter is organized as follows. Section 5.4 describes the rational behind our experimental setup. Section 5.6 contains the results of experimental evaluations. In Section 5.7 we discuss these findings and their applicability. Lastly, we conclude this work in Section 5.8.

5.4 Methodology

Parasitic elements are inherent to any circuit. As byproducts of the placement and routing, their values can not be controlled directly. Instead designers—with the aid of EDA tools—try to minimize their values such that their impact on performance allows meeting the design constraints. Models and extraction tools for digital design are tailored to this purpose, creating a gap between the analog and digital behavior. We aim to investigate this gap to detect how may parasitic elements—while being acceptable for the circuit performance—impact the SCA security of masked designs and to what extent.

To this end, we use the analog SPICE simulator from the Synopsys FineSim v2018.09 suite and the 45 nm open-source standard-cell library from Nangate [78] in junction with predictive transistor models [149]. We use transient simulation with a 10 ps simulation step, equivalent to a 100 GS/s sampling rate. To ensure equidistant sampling at the simulator output, we set the `strobeperiod=10p` argument of the `.tran` simulation.

Our setup allows to obtain single precision, noiseless and perfectly aligned simulations, which may represent an overly pessimistic SCA evaluation scenario. Our aim is however not to determine how feasible such leakages can be exploited or even measured in practical laboratory settings, but rather to diagnose which parasitic elements have the potential to compromise the SCA security. Lastly,

we focus on the potential sources of leakage stemming from the design, rather than the influences of measurement setups [72, 86].

We aim to conduct a preliminary investigation of possible sources of leakage, without questioning how probable they may be. We annotate each parasitic element individually in order to observe its independent impact on the SCA security. For each parasitic element, our methodology works as follows:

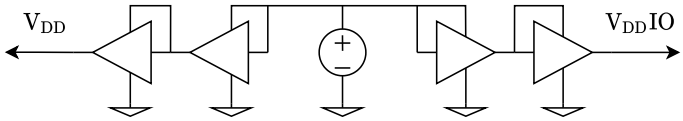
1. Chose the sweep ranges such that they cover scenarios varying from negligible small to values that cause the circuit to malfunction.
2. Perform series of transient analyses by sweeping the value of each parasitic element.
3. Probe voltage waveforms of input and output data nodes to continually verify the correctness of computations.
4. Probe all supply current waveforms.
5. Perform SCA security evaluation on supply current waveforms.

Here, recorded supply current waveforms are used to form side-channel traces. Each transient simulation is driven using randomly generated input data vectors, adhering to the TVLA evaluation methodology. We use the TVLA as the core metric to evaluate the information leakage in the function of each parasitic element and its values, relative to the parasitic-free case.

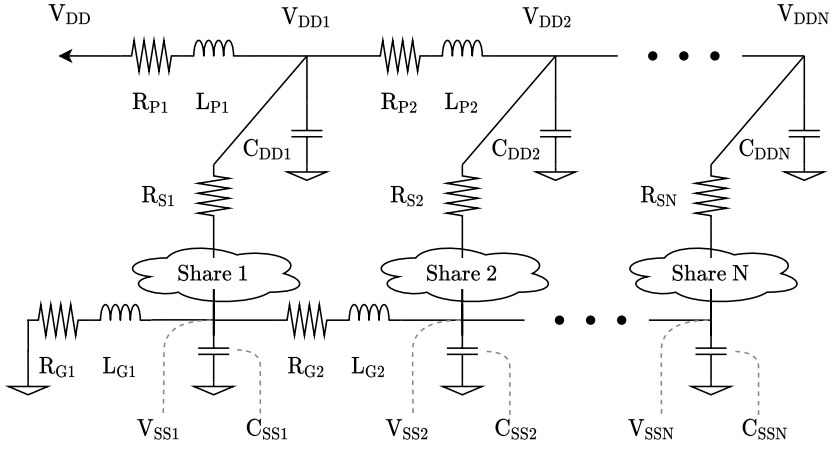
5.5 SPICE Model

Our SPICE model for co-simulation of the PDN with the logic core is depicted in Figure 5.3. We buffer the ideal voltage source using standard-cell buffers of finite current capacity, as shown in Figure 5.3a. Buffers generate two different power supplies: one for the core logic and one for buffering of the ideal inputs coming from the SPICE vector file (`*.vec`). Figure 5.3c shows how input and output signals are driven and loaded.

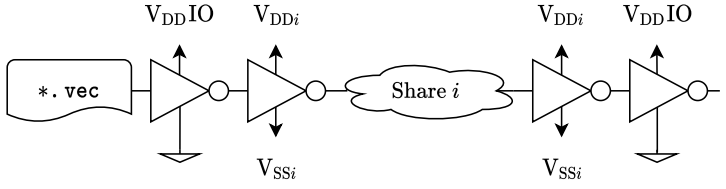
We thus isolate the target circuitry to ensure that signals adhere to the models of the standard-cell library, preventing the influence of any ideal waveforms that may offset the results obtained for smaller target circuits. In other words, we create a “sterile” environment for observing the supply currents of shares. Furthermore, we account for the high-frequency response caused by supply currents drawn by the CMOS gates in logic shares. Unlike the constant



(a) PDN load buffers.



(b) PDN R, L, C components.



(c) Data input isolation.

Figure 5.3: SPICE model for PDN co-simulation.

impedance of parasitic resistors $Z_R = R$, both the impedance of parasitic inductances $Z_L = j\omega L$ and capacitances $Z_C = \frac{1}{j\omega C}$ depend on the angular frequency of the inciting current. Thus the voltage fluctuations over Z_L and Z_C cause non-linear ground-bounce and supply-sag. Lastly, in contrast to a security evaluation where the waveform of merit is the supply current that can be measured, we chose the supply current i_{CORE}, as per Equation (5.2), as the side-channel for diagnostic purposes. Thus we isolate the core logic from ideal waveforms and create a sterile environment for observing supply currents of shares.

$$i_{CORE} = \sum_{j=1}^{N} i_{V_{DDj},V_{SSj}} \quad . \tag{5.2}$$

5.5.1 Target Circuits

Any Boolean function over binary fields $GF(2^n)$ can be implemented in algebraic normal form using only two-input XOR and AND gates, XOR2 and AND2 respectively. Thus secure instances of these two gates are fundamental to protect any Boolean masking scheme. Boolean masking schemes use XOR as the sharing operator, making the linear layer masking trivial. Given an n-bit XOR2 gate $z_i = a_i \oplus b_i$, sharing it into N shares requires $z_{i,j} = a_{i,j} \oplus b_{i,j}$; where $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, N\}$. Conversely, the unshared output bit is obtained as $z_i = \bigoplus_{j=1}^{N} z_{i,j}$. Figure 5.4 shows the generic schematic of an n-bit XOR2 gate with N = 2. It provides first-order security, that is, it prevents SCA that exploit leakage in the first-order statistical moment.

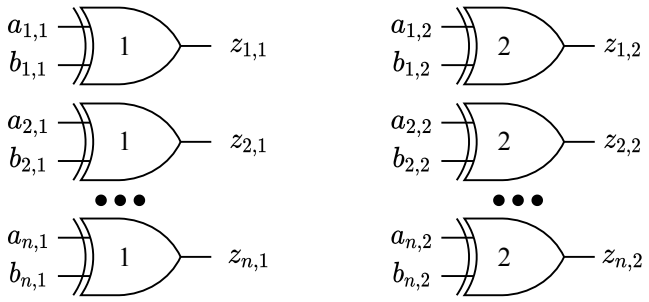


Figure 5.4: Shared n-bit XOR2 circuit with N = 2, implemented using `XOR2_X1` gate. Each gate is marked with its share number.

In contrast, masking non-linear operations is a demanding task. Figure 5.5 shows the schematic of an AND2 gate with N = 3, proposed in [19]. It also provides first-order security, but requires an additional share and an additional random input bit r, compared to the protected version of XOR2.

Our study focuses on these simple circuits, as they allow to address virtually any Boolean masking schemes, looking from the algebraic perspective. Additionally, their small sizes are favorable for SPICE simulation.

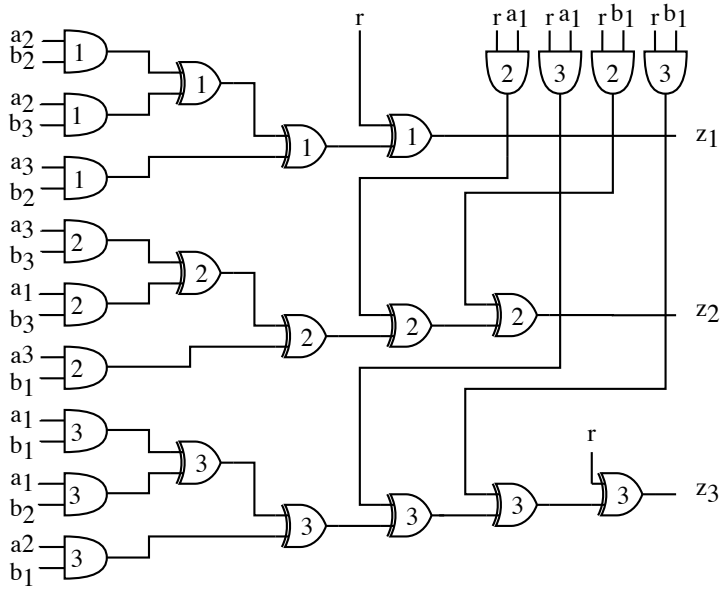


Figure 5.5: Shared 1-bit AND2 circuit with N = 3, implemented using `XOR2_X1` and `AND2_X1` gates. Each gate is marked with its share number.

5.5.2 Security Metric

We use TVLA [32] as the core security metric. TVLA determines whether statistical moments of two sets of data, of N_1 and N_2 elements, are distinguishable by using Welch's t-test. Upon partitioning traces based on unshared data, the security order is defined as the highest statistical moment in which the Welch's t-statistic does not exceed a confidence interval of $|t| \leqslant 4.5$. For the first-order security, a score t^i is computed for each sample i in the supply current waveform as per Equation (5.3), where μ^i and $(\sigma^i)^2$ are sample mean and variance normalized to the number of samples in each set N_1, N_2.

$$t^i = \frac{\mu_1^i - \mu_2^i}{\sqrt{(\sigma_1^i)^2/N_1 + (\sigma_2^i)^2/N_2}} \ . \tag{5.3}$$

The resulting t-trace is a temporal waveform that contains all t^i samples corresponding to a fixed number of traces (constant N_1 and N_2). In our experiments in Section 5.6 we often plot the trend of the $max(|t|)$ score for an increasing number of traces. A constant $max(|t|)$ trend with $|t| \leqslant 4.5$ indicates a certainty that no leakage exists. On the contrary, a rising $max(|t|)$ trend indicates that collecting more traces may lead to a vulnerability. Alternatively,

by fixing the number of traces, we can additionally observe the $max(|t|)$ trend function of the swept parasitic element value. Doing so allows us to identify the parasitics that may compromise the SCA security. Due to the small circuit sizes and the noiseless nature of simulation, the number of traces we simulate in certain cases is relatively small. Nevertheless, this does not undermine the findings. Should we increase the number of traces rising $max(|t|)$ trends would only lead to higher t-scores, further strengthening our observations.

5.6 Experimental Results

In this section we provide experimental results of the parametrized sweeps of parasitic elements. We target the first-order secure instances of a two-share XOR2 and a three-share AND2 gates. In all cases we use TVLA with partitioning based on the unshared output.

5.6.1 Impacts of PDN

To examine the impact of PDN we use an 8-bit instance of the two-share XOR2 gate depicted in Figure 5.4. We instantiate eight `XOR2_X1` cells in parallel to instigate a more significant supply current. As the circuit computation is completely linear and inputs of each share are independent, all observed leakage may stem only from the PDN. We simulate 2^{15} traces for each experiment.

Firstly, we investigate how the finite current capacity of the power supply impacts the SCA security, independently of the layout parasitics. By setting $\mathrm{R_{Pi}} = \mathrm{R_{Gi}} = 0\,\mathrm{Ohm}$, $\mathrm{L_{Pi}} = \mathrm{L_{Gi}} = 0\,\mathrm{H}$ and $\mathrm{C_{DDi}} = \mathrm{C_{SSi}} = 0\,\mathrm{F}$ and omitting the buffers, we reduce our model to the idealization depicted in Figure 5.2. We then introduce buffers of different drive strength and plot the average supply sag for each case in Figure 5.6. There is no significant difference between the two supply nodes. Average supply sag is 10%, 15%, and over 20% of the nominal value for `BUF_X32`, `BUF_X8`, and `BUF_X1`, respectively. In the latter case, the excessive supply sag causes the circuit to malfunction. Therefore, this case is not relevant as such a device would never enter the market. However, supply sag in the range of 10–15% is likely to occur.

Figure 5.7 shows the $max(|t|)$ score for an increasing number of traces. As the ideal voltage source has infinite current capacity, it maintains the nominal supply voltage throughout the circuit operation. Consequently, the corresponding $max(|t|)$ score has a stable horizontal trend around the $|t| = 4.5$ mark.

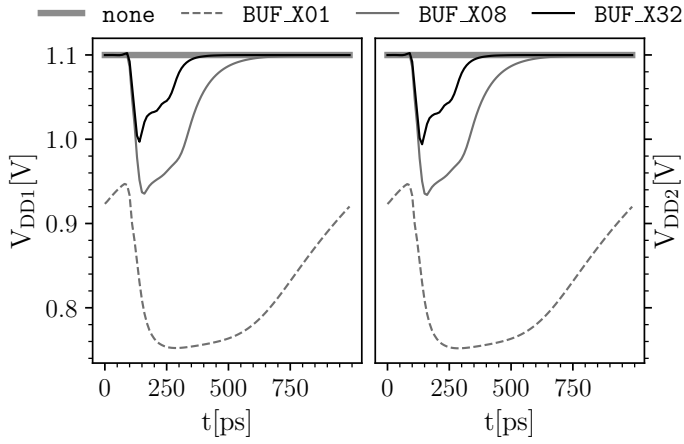


Figure 5.6: Two-share 8-bit XOR2, average supply sag.

However, using buffers with lower current capacity, results in higher supply sag. As a result the $max(|t|)$ scores start to rise. For, `BUF_X32` and `BUF_X8` $max(|t|)$ trends stabilize below the $|t| = 4.5$ mark. However, this increase indicates that the current capacity, *i.e.* driving strength, of the power supply influences side-channel security of masked designs. In the rest of the experiments we use `BUF_32` to model the finite current capacity of the power supply. We call this the referent, parasitic-free, case.

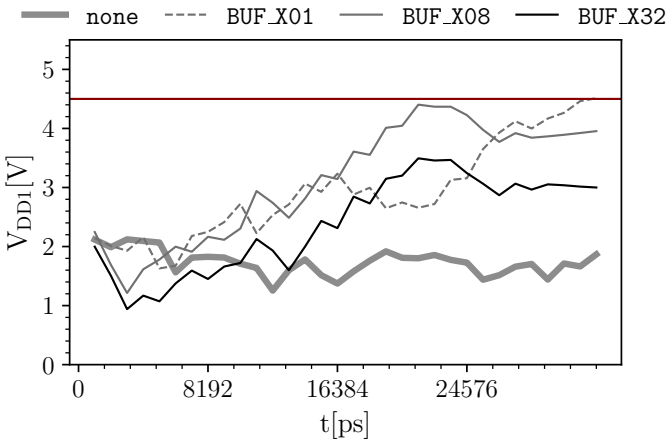


Figure 5.7: Two-share 8-bit XOR2, impact of supply buffers to $max(|t|)$ score.

Next, we move to investigate the impact of parasitic elements in the PDN: resistors, capacitors and inductors. The results are shown in Figures 5.8, 5.9 and 5.10. We overlap results obtained for the parasitic elements that exhibit similar behavior, as well as $max(|t|)$ trend curves for different sweep values.

Figure 5.8 shows the impact of IR drop over parasitic resistors. We sweep resistor values between 1 Ohm and 10 kOhm using a logarithmic sweep. On the one hand, R_{P1} and R_{L1} affect the supply nodes of both shares. Increasing them to 3.5 kOhm and 6.3 kOhm, respectively, leads to a slight increase in the $max(|t|)$ score, though remaining below the $|t| = 4.5$ threshold. Further increases of their value break the circuit functionality. On the other hand, R_{P2} and R_{L2} decrease the $max(|t|)$ score when increased up to 4.0 kOhm and 3.9 kOhm, respectively. As they affect only the supply node of the second share, they have a limiting effect, as serial resistors, on the supply currents drawn by that share. Therefore, they attenuate the signal of the chosen side-channel and, consequently, the amount of leakage.

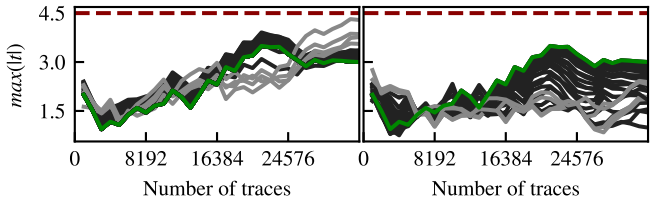


Figure 5.8: Impact of resistors R_{P1} and R_{G1} (left), R_{P2} and R_{G2} (right). The referent case is green, correct traces are black, malfunctioning traces are grey.

Figure 5.9 shows the impact of the PDN capacitors. We sweep capacitance values between 1 fF and 1 nF using a logarithmic sweep. In addition to the capacitors towards the ground plate, we sweep capacitors between supply nodes of the same polarity (left), and capacitors between supply nodes of the opposing polarity (right).

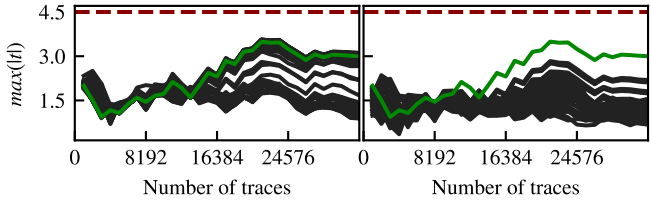


Figure 5.9: Impact of capacitors C_{DD1},C_{SS1},C_{DD2},C_{SS2}, $C_{V_{DD1},V_{DD2}}$, and $C_{V_{SS1},V_{SS2}}$ (left), $C_{V_{DD1},V_{SS1}}$,$C_{V_{DD1},V_{SS2}}$, $C_{V_{DD2},V_{SS1}}$, and $C_{V_{DD2},V_{SS2}}$ (right). The referent case is green, correct traces are black. There are no malfunctions.

The latter are referred to as decoupling capacitors in the literature. Serving as charge caches for the logic core, they lower the required bandwidth of the PDN. In other words, they filter information carrying high-frequency components of the supply current. Hence, we observe a decrease in the $max(|t|)$ score.

Figure 5.10 shows the impact of the PDN inductors. We sweep inductance values between 1 pH and 1 μH using a logarithmic sweep. L_{P1} and L_{G1} lead to $max(|t|) > 4.5$ when they exceed 1.2 nH and 0.8 nH, respectively. They produce peak $max(|t|)$ scores of between 20 and 23 when increased up to 18 nH. L_{P2} and L_{G2} have significantly lower impact, as the affect only the supply node of the second share. Figure 5.11 shows the magnitude of the $max(|t|)$ score function of parasitic inductors.

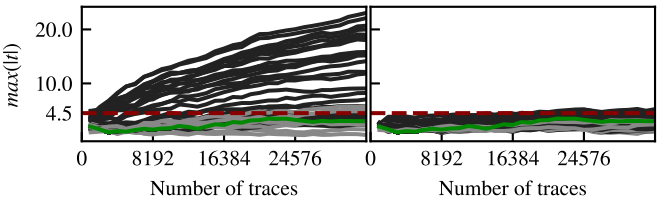


Figure 5.10: Impact of inductors L_{P1} and L_{G1} (left), L_{P2} and L_{G2} (right). The referent case is green, correct traces are black, malfunctioning traces are grey.

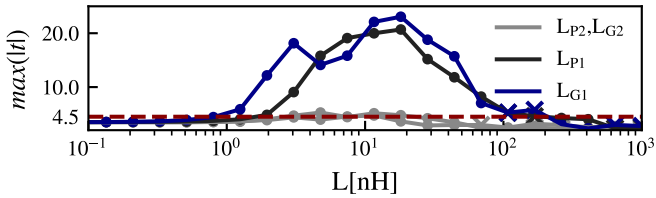


Figure 5.11: Magnitude of the $max(|t|)$ for 2^{15} traces; • denotes correct and × failed computation of the two-share 8-bit XOR2.

5.6.2 Effects of Coupling Capacitances

The last part of our experiments examines the impact of coupling capacitances. For a circuit with n nodes, a total of $\binom{n}{2}$ capacitors can be annotated between them. The computational effort of exhausting all $\binom{48+4}{2}$ capacitors of the 8-bit XOR2 is non-permitting. Thus in order to keep the computational effort reasonable, we perform experiments on the 1-bit shared XOR2 shown in Figure 5.4. We then move on to the AND2 gate shown in Figure 5.5 with $\binom{32+6}{2}$ possible capacitors.

Shared XOR2

Given the four input bits, there exist only $2^{4\times2} - 2^4 = 240$ non-trivial input transitions to simulate. By excluding the supply to supply capacitors discussed in the previous section, we split the remaining $\binom{10}{2} - 6 = 39$ capacitors as follows:

- *share-rail* (SR) between data nodes of one share and the supply node of the same share;
- *cross-rail* (CR) between data nodes of one share and the supply node of another share;
- *share-data* (SD) between data nodes within one share;
- *cross-data* (CD) between data nodes of different shares.

We sweep each capacitance using a logarithmic sweep between 10 aF and 100 fF. The impact of all four categories of capacitors is shown in Figure 5.12: SR, CR and SD (left) and CD (right). In accordance with theory, only CD capacitances, C_{z_1,z_2} in particular, result in a significant increase the SCA leakage. The rest of capacitances cause barely any deviation from the referent $max(|t|)$ trend.

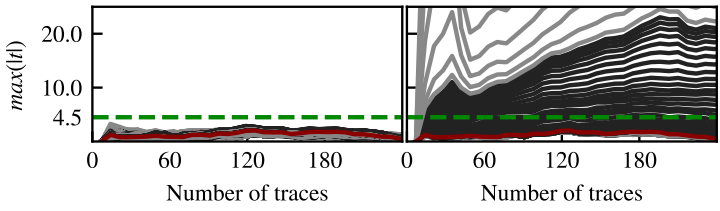


Figure 5.12: Impact of SR, CR, SD (left) and CD (right) on 1-bit XOR2, N = 2; overlapped sweep traces.

Figure 5.13 shows the magnitude of the $max(|t|)$ function of CD capacitance. The presence of leakage due to C_{z_1,z_2} starts at 0.3 fF, peaking to $max(|t|) = 23.1$ at $C_{z_1,z_2} = 19.3$ fF. For larger values, the circuit starts to malfunction.

Qualitatively, this result is expected, as we partition based on the unshared value of the circuit output. Similar behaviors are obtained for capacitors coupling input pins, *e.g.* C_{a_1,a_2} should an input pin, *e.g.* a, be the partitioning target.

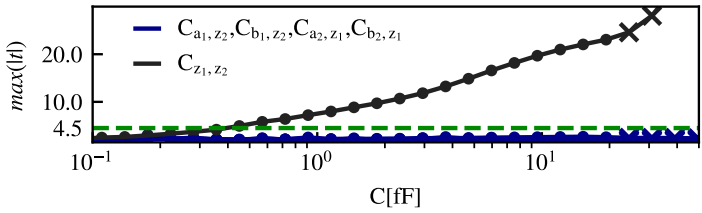


Figure 5.13: Magnitude of the $max(|t|)$ for 240 traces; • denotes correct and × failed computation of 1-bit XOR2.

Therefore only a subset of parasitic capacitors harms the side-channel security. Next we investigate how the remaining CD and SD capacitors impact the side-channel security in combination with a large C_{z_1,z_2} capacitor. We fix the value $C_{z_1,z_2} = 4.9\,\text{nF}$, and repeat the experiment when this is the only non-zero CD capacitance. Keeping $C_{z_1,z_2} = 4.9\,\text{nF}$ fixed, we run 100 randomized experiments. Each time, we sample the normal distribution $\mathcal{N}(\sigma^2 = 0.2\,\text{fF}, \mu = 1\,\text{fF})$ to annotate the values for all remaining CD and SD capacitors.

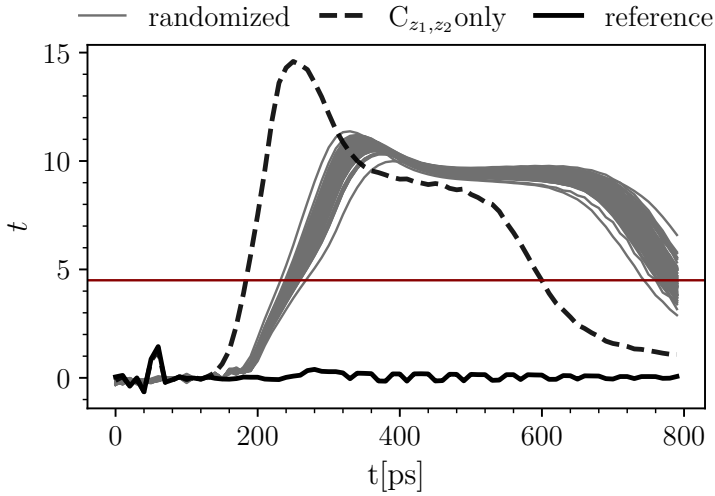


Figure 5.14: The t-trace for the sole $C_{z_1,z_2} = 4.9\,\text{fF}$, compared to added 100 randomly chosen values of the remaining CD and SD capacitors.

Other than slowing the computation down, added capacitors unanimously reduce the t-scores. This is better depicted using the $max(|t|)$ trend in Figure 5.15. Increasing parameters μ and σ^2 of the random capacitance distribution causes

a wider spread of the randomized t-traces. Nevertheless, the $max(|t|)$ scores are consistently lower compared to the sole CD capacitance $C_{z_1,z_2} = 4.9\,\text{fF}$.

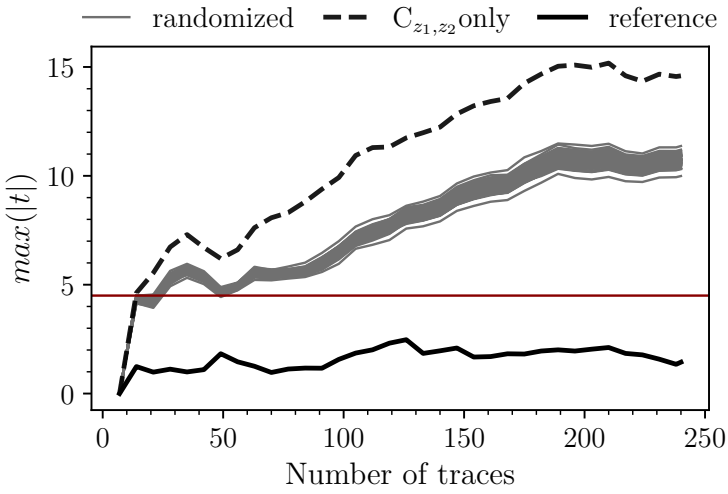


Figure 5.15: Magnitude of the $max(|t|)$ score for 240 traces for the sole $C_{z_1,z_2} = 4.9\,\text{fF}$, compared to added 100 randomly chosen values of the remaining CD and SD capacitors.

TI AND2

Given the 7 input bits there exist only $2^{7\times2} - 2^7$ non-trivial input transitions to simulate. As we annotate numerous 326 CD capacitors for the non-linear TI AND2, N = 3 gate, we lower the amount of experiments to 2^{13} traces. We sweep each capacitance using logarithmic sweep between 0.1 fF and 128 fF using a logarithmic sweep. Out of 326 CD capacitors, we identify 191 which cause $max(|t|) > 4.5$ while allowing the circuit to compute correctly. Based on the minimal capacitance value for which the capacitor causes $max(|t|) > 4.5$, henceforth called *critical value*, we further separate these capacitors in three groups:

- 18 high risk (H) capacitors, with critical value $C_H \leqslant 1\,\text{fF}$.
- 88 medium risk (M) capacitors, with critical value $1\,\text{fF} \leqslant C_M \leqslant 4\,\text{fF}$
- 85 low risk (L) capacitors, with critical value $C_L > 4\,\text{fF}$.

We chose the delimiting values of 1 fF and 4 fF for demonstrative purposes for the given circuit. Nevertheless, they are meaningful values for they are the same order of magnitude as the pin capacitances of the standard-cells we use. Figure 5.16 shows the magnitude of the $max(|t|)$ in the function of the H, M, L groups of CD capacitances. It is important to notice that we deem the capacitors with lower critical value to bear more risk, despite leading to lower $max(|t|)$ scores before causing the circuit to malfunction. Most of these 191 CD capacitors cause the circuit to malfunction when they reach between 17.5 fF and 23.7 fF in value or more. We discuss this ranking in more detail in Sec. 5.7.

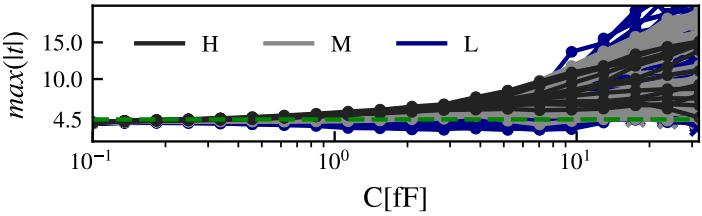


Figure 5.16: Magnitude of the $max(|t|)$ for 2^{13} traces; • denotes correct and × failed computation of TI AND2.

5.7 Discussion

In this section we discuss our findings in detail. We argue these phenomena could demonstrate themselves in a practical scenario. Lastly, we propose further research steps.

5.7.1 Impacts of the PDN

To the best of our knowledge, this is the first time a detailed consideration is given to the PDN modeling for SCA security. Firstly, we show that the finite current capacity of the power supply may increase the levels of leakage even when everything else in the circuit is ideal, Figure 5.7. Contrary to the suspicions of [30], our experiments do not show the resistive IR drop as one of the leading sources of SCA leakage. As shown in Figure 5.8, PDN resistors may increase the $max(|t|)$ score before the circuit malfunctions. In our experiments this increase is small. As serial impedances in the PDN circuit, R_{P1} causes the sag on supply nodes of both shares. Therefore, the $max(|t|)$ score increase is small, such resistors should not be neglected during security analysis. Furthermore, our experiments further show the decoupling capacitances, often used in PDN

modeling, lead to the decrease of the $max(|t|)$ score, as shown in Figure 5.9. Reactive impedances $Z_C = \frac{1}{j\omega C}$ form highly conductive paths for the data dependent high-frequency components of the supply current without upsetting the power supply voltages. Our experiments show however that parasitic PDN inductors are the more likely culprit, causing a steep rise in the $max(|t|)$ score, as shown in Figure 5.10 Reactive impedances $Z_L = j\omega L$ cause voltage drop predominated by the data dependent high-frequency components of the supply current drawn by the target circuit. More importantly, the supply sag caused by the serial inductor is more dependant on the data-carrying high-frequency signal. The extent to which such a data-dependent upset of the power supply nodes can break the independent leakage assumption is shown in Figure 5.11.

Ground-bounce and supply-sag are prominent issues at the packaging level, over long bonding wires, as they may cause the upset of supply nodes that makes the circuit malfunction. While some physical cells, such as power gating circuitry, are shown to cause significant upsets [75], their effects are certainly smaller within the logic core. Still, our experiments show that parasitic inductors may break the independence assumption at values $L \leqslant 1\,\text{nH}$ two orders of magnitude before causing the circuit to malfunction at $L \geqslant 100\,\text{nH}$. This margin is design specific and likely not to be this dramatic. Yet as long as the performance constraints are met, parasitic inductors with significant impact on the SCA security may remain in the design. Moreover, parasitic inductances on the order of nH, or worse yet tens of nH, may seem unrealistically large to an experienced design practitioner. As our target designs are minute, easily four orders of magnitude smaller than a practical design, orders of magnitude higher values are needed to instigate a significant voltage fluctuation in the supply nodes.

Lastly, decoupling capacitors are used to battle the effects of the ground-bounce and supply sag. This effect is shown in Figure 5.9. Nevertheless, decoupling capacitors are not free, they require area and design effort. Larger area means increased the PDN parasitics, as the rails grow longer. Hence, they are likely to be sized up to the extent that satisfies the performance margin, not the SCA one.

5.7.2 Impacts of Coupling Capacitances

Even though theoretical leakage models completely overlook the influence of the PDN, SD and CD coupling capacitances are widely accepted as the possible source of leakage. Our experiments support these suspicions and quantify them further. We confirm that SR and SD capacitances lead to no significant leakage regardless of their size. Interestingly, CR capacitances yield no significant leakage either, although they span from one share to another. Therefore, CD

capacitances remain as the only potential sources of leakage. This is clearly shown in the fundamental example of the 1-bit XOR2, N = 2 gate in Figure 5.13. Only the capacitance C_{z_1,z_2} directly coupling the shares of the partitioning target causes $max(|t|) > 4.5$, indicating significant leakage.

Once all of the CD and SD capacitors are considered jointly the resulting $max(|t|)$ score is lowered from the independent scenario. To show this we choose the moderately high value of $C_{z_1,z_2} = 4.9\,\text{nF}$, while the other capacitors are on average about five times smaller. As the target circuit is small, it can be expected to be sufficiently localized for capacitances equivalent to the annotated ones to exist. Thus, we attribute the decrease in t-scores to the charge distribution among these uneven capacitors, causing an upset in the supply current waveform drawn. Moreover, all CD and SD capacitances, including the C_{z_1,z_2} would belong to the same Gaussian distribution if they were extracted instead of annotated. Hence we believe the relative decrease compared to the one demonstrated in Figure 5.14 would be even more prominent.

In the case of TI AND2, N = 3 however determining which capacitances may cause information leakage is not so trivial. We qualify 191 out of all 326 CD capacitances as the potential source of leakage. We rank them to emphasize the following. Higher $max(|t|)$ score at the right-hand end of the Figure 5.16 are a consequence of larger capacitances in the range between 17.5 fF and 23.7 fF or more. While their impact is indeed higher, we expect fewer of such large parasitic elements to occur in a functioning design. On the other hand, capacitors on the order of magnitude of 1 fF are highly likely to occur given the input capacitances of cells in the target library (on the order of fF), especially assuming unconstrained placement and routing. Hence, we asses the potential risk from such capacitors to be higher. Lastly, note that the joint impact of annotated capacitors, demonstrated in Figure 5.14, applies to this circuit as well.

5.8 Conclusions

We have proposed a model for co-simulation of the analog PDN along with the digital logic core dedicated to SCA. Using said model we conduct an exploratory study to quantify the impact of different layout parasitics to SCA security. We craft experiments in a way that allows us to diagnose all possible sources of leakage from within the logic core. Selecting small, fundamental, circuits allows us to perform all experiments using SPICE. This allows us to categorize parasitic elements and quantify their impact on SCA security. Furthermore, we are the first to study the impact of PDN to the security of the core logic. As all

of these issues arise in the back-end stages, they are left out of the mathematical models although they may account for the significant information leakage. We provide the first detailed insights into the matter.

Computational complexity of SPICE simulators prevents this model to be applied on larger designs. It is worth investigating whether other existing EDA tools could be used for such analysis. For example, supply current waveform obtained using data driven digital simulators such as Synopsys PrimeTime (PT) with PX addon could possibly be back-annotated into the analog PDN model. In this case, manual annotation and export of parasitic components can be used to study effects of parasitic components in a generic manner, without having to go through the laborious process of multifaceted placement and routing.

Other than the investigative nature of this method, we believe it can be adopted by the designers, assuming the computational aspects are overcome. Given a gate-level netlist of the target design, practitioners could perform course grain exploratory sweeps to determine which parasitic elements may be critical to their designs. Thus they could focus on their extraction and analysis, potentially tailoring Electronic Design Automation (EDA) tool constraints to bound said parasitic elements.

Chapter 6

Conclusions and Insights for Future Research

> If more information was the answer, then we'd all be billionaires with perfect abs.
>
> Derek Sivers

This dissertation tackles different aspects of design-time SCA security evaluations. We aim to uncover the physical vulnerabilities prior to chip manufacturing. To do so, we cascade down the standard-cell ASIC hardware design flow, addressing design stages as depicted in Figure 6.1.

6.1 Conclusions

Firstly, in Chapter 3 we go far and wide to study the interoperability of the state-of-the-art EDA tools and models with the state-of-the-art SCA evaluation methods and metrics. We introduce several SCA-aware evaluation steps as denoted in Figure 6.1 using the red shade. We strengthen the practicality of our findings by implementing a flexible framework that aids the automation and supports the integration of design-time evaluations into the standard-cell ASIC hardware design flow. We list the following findings.

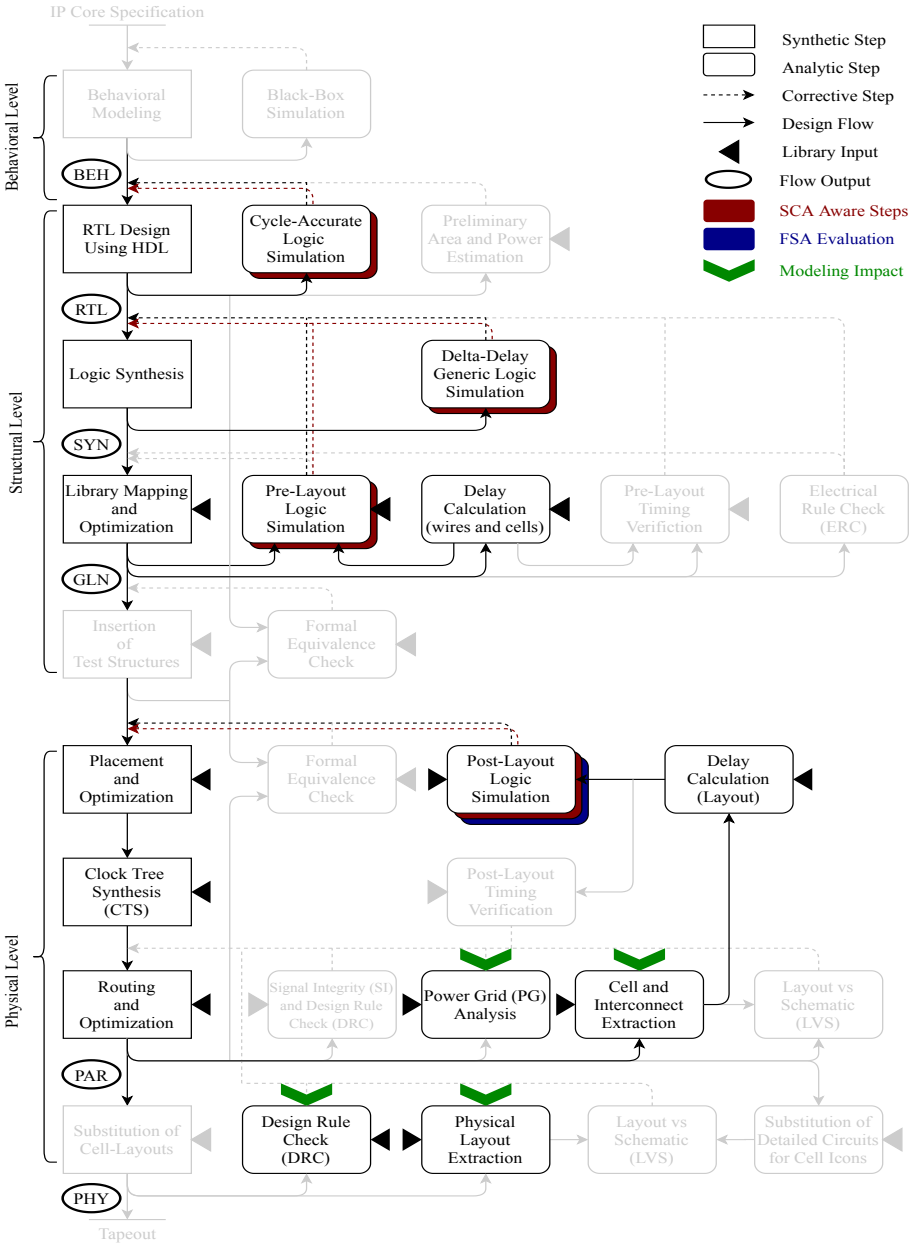


Figure 6.1: The impact of this thesis on the standard-cell ASIC design flow.

- Flaws in cryptographic implementations based on standard-cell libraries can be detected as soon as the gate-level netlist is synthesized.
- This can be done efficiently based on the logic simulation and simple power models as the Marching Sticks Model (MSM), and timing models as simple as the Δ-delay model. As simulation performance varies slightly depending on the underlying timing model, we recommend performing evaluations based on Composite Current Source (CCS) timing where possible, *i.e.* when the target library is known.
- Using increasingly detailed models detects leakage with a smaller number of traces. CCS power simulation, *e.g.* using PT, does introduce a significant overhead. However, it is predicated by the logic simulation that constitutes the bulk of the MSM power computation. Therefore, it is always advantageous to perform the MSM-based evaluation before the PT traces are ready.
- Although they are performed in the same manner, we differentiate glitch-resistant Boolean masking schemes evaluations from standard-cell-based secure logic styles evaluations. The former benefit from high-precision observations and time resolution, as we aim to maximize the occurrence of glitches. For the latter, high simulation precision, absence of noise and jitter, as well as the level of detail provided by the CCS models, expose the security-critical asymmetries beyond what can be measured.
- The 1 ps characterization precision of the CCS is needed for modeling on-chip behavior. However, we believe that the 1 TS/s measurement sampling is unattainable using today's CMOS technology. As moving to the 10 ps simulation precision does change the evaluation of Boolean masking schemes and gives a more realistic evaluation of the standard-cell-based secure logic styles, we recommend this precision for the most rigorous simulations. Its sampling period, *i.e.* 100 GS/s sampling rate, corresponds to the 50 GHz measurements bandwidth that can be achieved using high-end equipment only. Moreover, as these observations are free from any distortions and low-pass filtering of the physical components such as the PDN and decoupling capacitors, we believe this simulation precision need not be exceeded.
- As synthesis tools, both logic and physical, can breach properties of countermeasures (*e.g.* the non-completeness property of the threshold implementations) it is important that this approach be applied at each design stage. Moreover, improper handling of the design constraints can cause false positive leakage assessment.

- Parsers, analyzers, automation scripts and data formats developed within the Computer-Aided Side-Channel Analysis Design Environment (CASCADE) framework serve to demonstrate practicalities of the approach and the feasibility in terms of performance in realtime. These results are enabled by the underlying fast gate-level CCS models.

Secondly, in Chapter 4 we show how the high level of physical detail provided by the CCS models can be used to scrutinize theoretical considerations regarding masking schemes. In particular we examine resistance of glitch-resistant Boolean masking schemes to Fault Sensitivity Analysis (FSA). The high-level of physical detail captured by the CCS models is a good tradeoff between the abstract assumptions masking schemes make about hardware and the analog behavior of the physical world. Naturally, CCS models do not encompass all physical phenomena. However, they characterize the temporal behavior of physical cells at the 1 ps precision. As FSA leverages measuring data-dependent propagation delays, we rely on this precision to construct the best-case scenario for the attackers. Namely, we give the attackers the 1 ps precision noiseless traces, that are otherwise unattainable in practice. In this scenario, we apply FSA to two glitch-resistant Boolean masking schemes, to show that they resist FSA under a set of assumptions about the underlying hardware. These experiments relate in the backend stages of the standard-cell ASIC design flow as per Figure 6.1, denoted using the blue shade.

Thirdly, in Chapter 5 we step away from the gate-level modeling to investigate the impact of layout parasitics to the SCA security of glitch-resistant Boolean masking schemes. We design an experimental setup that allows us to examine the impact of individual parasitic elements. Contribution of this work are twofold.

- To the best of our knowledge we are the first to consider the RLC parasitic elements of the PDN in a SCA security evaluation. Accordingly we create a co-simulation model for the analog PDN along with the digital logic core that implements the shares.
- We use the parametric sweeps of individually annotated parasitic values to examine how the impact the SCA security. Using this approach we confirm that inter-share capacitances coupling data nodes may cause the deterioration of the SCA security. Additionally, we demonstrate the leakage-mitigation that comes from decoupling capacitances. More importantly we point out to parasitic resistors and inductors as potential sources of SCA information leakage.

These findings may indirectly impact steps of the standard-cell ASIC hardware design flow as noted using green arrows in Figure 6.1. They do so by potentially changing how layout parasitics are extracted and included in the analysis of SCA-resistant designs.

6.2 Future Research Directions

Following the work for this dissertation we find it important to make the distinction depicted in Figure 6.2. The left-hand portion depicts the "modeling for performance" of ASIC chips, common in the EDA world. We believe that the "modeling for SCA security" needs to address what attackers can measure. In the case of the non-invasive adversaries monitoring power consumption it is the current waveform of the digital logic core distorted by the analog PDN, decoupling capacitors and so on.

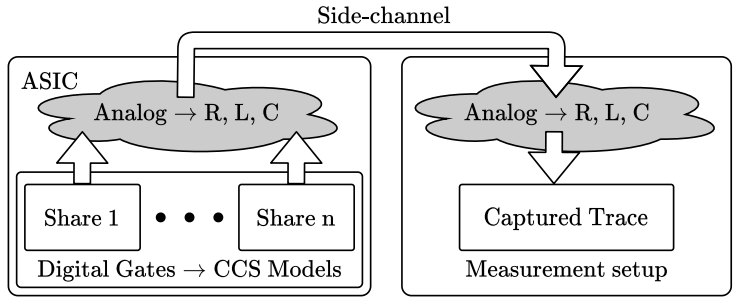


Figure 6.2: Modeling for performance (left) and SCA security (left and right).

The digital logic core in Figure 6.2 is depicted as a series of independent shares, but the insights hold for any other standard-cell-based countermeasure. The sophisticated CCS models developed by the EDA industry are meant to capture the behavior of digital gates at the highest precision possible. They are meant to represent the behavior of the logic core with the highest accuracy, such that every last picosecond of the timing slack can be attained for the optimal performance under target constraints. Thus modeling for performance is clearly viable for the typical EDA applications. However, the high-frequency data-dependent current waveforms of the digital logic core are significantly distorted by the low-pass filters in the analog layers above them. Thereby, from Chapter 3 we believe that the excessive simulation precision can be safely degraded for the purpose of design-time SCA evaluations. Furthermore, from Chapter 5 layout extraction can be done selectively as many parasitic elements do not affect the SCA security, *e.g.* intra-share coupling capacitors. The same

chapter leads us to believe that that inter-share coupling capacitors are less likely to cause a noticeable information leakage. Instead, we believe the "macro" coupling through the PDN that mixes the currents of all shares is the more likely culprit.

We believe that models similar to the PDN co-simulation model we introduce are principal for unequivocal discovery of the leakage sources. In Chapter 5 we simulate the entire test circuit using SPICE. Running similar experiments on a large-scale design, *e.g.* an entire masked AES implementation, can take several months. This can be done for a more detailed investigation. However, performing such a simulation for a design-time SCA evaluation would be a serious time-to-market increase. Instead, the power consumption waveforms of the digital core can be obtained using CCS simulations, *e.g.* using PT as if they were independent. These waveforms can be back-annotated as piece-wise linear current sources in the SPICE simulation, one for each of the shares. Thus the complexity of the SPICE circuit can be drastically reduced. This solution allows the off-the-shelf use of library models. However, the library characterization process, shown in Algorithm 1, can be extended to capture the current waveforms for different values of the power supply. Varying power supply values are already used for characterizing sets of operating conditions, therefore we see no technical difficulties for such an extended characterization. Given such models, it would be straightforward to implement an instantaneous power consumption simulator dedicated to SCA security. Unfortunately, this approach is predicated by technology manufacturers adopting this type of characterization.

Recently, Levi *et al.* [86] have demonstrated how a single external resistor can be used to degrade SCA security of a masked implementation, based on measurements. As this resistor corresponds to the R_{P1} and L_{P1} in our simulated setup shown in Figure 5.3b, we are further encouraged to believe that the sources of the inter-share coupling stem from the analog R, L, C parasitics "on top" of the digital logic core, rather than the inter-share capacitors coupling data nodes. In support of this insight we present Figure 6.3. In this motivational example, for the independent leakage assumption—the physical assumption that underlies all masking schemes—to hold, Equation (6.1) must be true.

$$i_{ext}(t)|_{\text{Share1+Share2}} = i_{ext}(t)|_{\text{Share1}} + i_{ext}(t)|_{\text{Share2}} \,. \tag{6.1}$$

In other words, the addition of the share supply currents is a linear operation, hence whether both shares are present at the same time $i_{ext}(t)|_{\text{Share1+Share2}}$, or only one of them exists in the circuit $i_{ext}(t)|_{\text{Share}i}$ the resulting current measured by the attacker is invariant. However, this is not the case, even in the simple model from Figure 6.3. The measurable current $i_{ext}(t)$ is as per Equation (6.2).

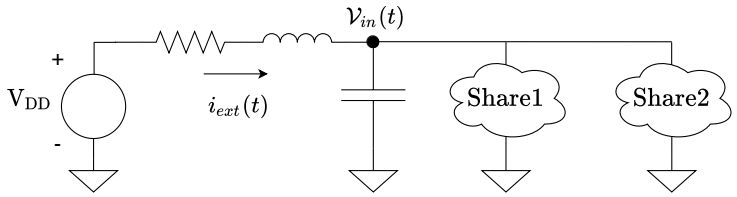


Figure 6.3: A simple model for SCA security of a shared design.

$$i_{ext}(t) = C\frac{d\mathcal{V}_{in}(t)}{dt} + i_{\text{Share1}}(data, \mathcal{V}_{in}(t), t) + i_{\text{Share2}}(data, \mathcal{V}_{in}(t), t)\,. \qquad (6.2)$$

Assuming that each share is implemented using the standard-cell CMOS, the drain current of each transistor depends on the supply voltage. Moreover, supply voltage change impacts the gain of the CMOS inverters, Section 5.3 in [120]. Therefore, as soon as one of the shares starts drawing a supply current it affects the supply voltage of both shares, creating a non-linear dependency between the two currents. The independent leakage assumption can never be fully attained in hardware. The research question that remain are how small does this coupling have to be in order for the designs to remain secure and how to model and evaluate it at design-time efficiently.

The counterpart depicted in the right-hand portion of Figure 6.2 corresponds to the modeling of the attacker's setup. The influence of measurement setups in simulation [71] and in practice [86] is addressed in the literature, albeit sparsely. We believe that more research should be done on that side too.

To increase the confidence in findings on both sides of the modeling for SCA security, we believe that dedicated tapeouts are needed. For a fixed Gate-Level Netlist (GLN) of a logic core, *e.g.* a masked AES implementation, several PDNs and on-chip decoupling capacitor banks can be implemented. Tapeouts should also include different placement and routing constraints. For instance, placement and routing can be such that the capacitive crosstalk among different shares is maximized. It would be interesting to demonstrate how much SCA measurements in this worst-case differ from an unconstrained placement and routing scenario.

The modeling for SCA security, depicted in Figure 6.2, combined with the current contributions of this dissertation and dedicated tapouts, are the key-stones of the reliable design-time evaluation for SCA attack resistant cryptographic implementations.

Bibliography

[1] FIPS PUB 140-3, Security requirements for cryptographic modules, 2019. U.S. Department of Commerce/National Institute of Standards and Technology.

[2] Abraham, D. G., Dolan, G. M., Double, G. P., and Stevens, J. V. Transaction security system. *IBM Syst. J. 30*, 2 (Mar. 1991), 206–229.

[3] Amin, C. S., Dartu, F., and Ismail, Y. I. Weibull-based analytical waveform model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24*, 8 (Aug 2005), 1156–1168.

[4] Arribas, V., Bilgin, B., Petrides, G., Nikova, S., and Rijmen, V. Rhythmic keccak: Sca security and low latency in hw. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2018*, 1 (Feb. 2018), 269–290.

[5] Arribas, V., Nikova, S., and Rijmen, V. Vermi: Verification tool for masked implementations. In *25th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2018, Bordeaux, France, December 9-12, 2018* (2018), IEEE, pp. 381–384.

[6] Arribas, V., Wegener, F., Moradi, A., and Nikova, S. Cryptographic fault diagnosis using verfi. *IACR Cryptol. ePrint Arch. 2019* (2019), 1312.

[7] Atlantic, T. Why you're probably getting a microchip implant someday, Sep. 2018. Accessed April, 2020.

[8] Balasch, J., Gierlichs, B., Verdult, R., Batina, L., and Verbauwhede, I. Power analysis of Atmel CryptoMemory - recovering keys from secure EEPROMs. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San*

Francisco, CA, USA, February 27 - March 2, 2012. Proceedings (2012), O. Dunkelman, Ed., vol. 7178 of *LNCS*, Springer, pp. 19–34.

[9] Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.-A., Grégoire, B., Strub, P.-Y., and Zucchini, R. Strong non-interference and type-directed higher-order masking. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, USA, 2016), CCS '16, Association for Computing Machinery, p. 116–129.

[10] Bellare, M., Canetti, R., and Krawczyk, H. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings* (1996), N. Koblitz, Ed., vol. 1109 of *Lecture Notes in Computer Science*, Springer, pp. 1–15.

[11] Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. Keccak. In *Advances in Cryptology – EUROCRYPT 2013* (Berlin, Heidelberg, 2013), T. Johansson and P. Q. Nguyen, Eds., Springer Berlin Heidelberg, pp. 313–314.

[12] Bertoni, G., and Martinoli, M. A methodology for the characterisation of leakages in combinatorial logic. In *Security, Privacy, and Applied Cryptography Engineering - SPACE 2016* (2016), C. Carlet, M. A. Hasan, and V. Saraswat, Eds., vol. 10076 of *LNCS*, Springer, pp. 363–382.

[13] Bhasin, S., Danger, J., Graba, T., Mathieu, Y., Fujimoto, D., and Nagata, M. Physical security evaluation at an early design-phase: A side-channel aware simulation methodology. In *Engineering Simulations for Cyber-Physical Systems - ES4CPS 2014* (2014), C. Berger and I. Schaefer, Eds., ACM, p. 13.

[14] Bhatnagar, H. *Advanced ASIC Chip Synthesis: Using Synopsys' Design Compiler and PrimeTime*. Kluwer Academic Publishers, USA, 1999.

[15] Biham, E., and Shamir, A. Differential fault analysis of secret key cryptosystems. In *CRYPTO* (1997), vol. 1294 of *Lecture Notes in Computer Science*, Springer, pp. 513–525.

[16] Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., and Rijmen, V. Higher-order threshold implementations. In *ASIACRYPT (2)* (2014), vol. 8874 of *Lecture Notes in Computer Science*, Springer, pp. 326–343.

[17] Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., and Rijmen, V. A more efficient AES threshold implementation. In *Progress in Cryptology*

– *AFRICACRYPT 2014* (Cham, 2014), D. Pointcheval and D. Vergnaud, Eds., Springer International Publishing, pp. 267–284.

[18] Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., and Rijmen, V. Trade-offs for threshold implementations illustrated on AES. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34*, 7 (2015), 1188–1200.

[19] Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., and Stütz, G. Threshold Implementations of All 3 x 3 and 4 x 4 S-Boxes. In *Cryptographic Hardware and Embedded Systems - CHES 2012* (2012), E. Prouff and P. Schaumont, Eds., vol. 7428 of *LNCS*, Springer, pp. 76–91.

[20] Bloem, R., Gross, H., Iusupov, R., Könighofer, B., Mangard, S., and Winter, J. Formal verification of masked hardware implementations in the presence of glitches. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II* (2018), J. B. Nielsen and V. Rijmen, Eds., vol. 10821 of *Lecture Notes in Computer Science*, Springer, pp. 321–353.

[21] Bloomberg. Internet of things market to reach us $ 1111.3 billion by 2026, at a ferocious cagr of 24.7%, Jul. 2019. Accessed October, 2019.

[22] Board, L. T. A. Liberty. Accessed May, 2020.

[23] Bogdanov, A., Khovratovich, D., and Rechberger, C. Biclique cryptanalysis of the full AES. In *Advances in Cryptology – ASIACRYPT 2011* (Berlin, Heidelberg, 2011), D. H. Lee and X. Wang, Eds., Springer Berlin Heidelberg, pp. 344–371.

[24] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., and Vikkelsoe, C. Present: An ultra-lightweight block cipher. In *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems* (Berlin, Heidelberg, 2007), CHES '07, Springer-Verlag, p. 450–466.

[25] Boneh, D., DeMillo, R. A., and Lipton, R. J. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT* (1997), vol. 1233 of *Lecture Notes in Computer Science*, Springer, pp. 37–51.

[26] Božilov, D., Knežević, M., and Nikov, V. Optimized threshold implementations: Securing cryptographic accelerators for low-energy and low-latency applications. Cryptology ePrint Archive, Report 2018/922, 2018.

[27] Brier, E., Clavier, C., and Olivier, F. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004* (2004), M. Joye and J. Quisquater, Eds., vol. 3156 of *LNCS*, Springer, pp. 16–29.

[28] Chari, S., Jutla, C. S., Rao, J. R., and Rohatgi, P. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology - CRYPTO '99* (1999), M. J. Wiener, Ed., vol. 1666 of *LNCS*, Springer, pp. 398–412.

[29] Chen, Z., Haider, S., and Schaumont, P. Side-channel leakage in masked circuits caused by higher-order circuit effects. In *Advances in Information Security and Assurance* (Berlin, Heidelberg, 2009), J. H. Park, H.-H. Chen, M. Atiquzzaman, C. Lee, T.-h. Kim, and S.-S. Yeo, Eds., Springer Berlin Heidelberg, pp. 327–336.

[30] Cnudde, T. D., Bilgin, B., Gierlichs, B., Nikov, V., Nikova, S., and Rijmen, V. Does coupling affect the security of masked implementations? In *Constructive Side-Channel Analysis and Secure Design -COSADE 2017* (2017), S. Guilley, Ed., vol. 10348 of *LNCS*, Springer, pp. 1–18.

[31] Cnudde, T. D., and Nikova, S. More efficient private circuits ii through threshold implementations. In *FDTC* (2016), IEEE Computer Society, pp. –.

[32] Cooper, J., DeMulder, E., Goodwill, G., Jaffe, J., Kenworthy, G., and Rohatgi, P. Test Vector Leakage Assessment (TVLA) Methodology in Practice. International Cryptographic Module Conference, 2013.

[33] Daemen, J. Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing. In Fischer and Homma [50], pp. 137–153.

[34] Daemen, J., and Rijmen, V. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[35] De Cnudde, T., Reparaz, O., Bilgin, B., Nikova, S., Nikov, V., and Rijmen, V. Masking AES with d+1 shares in hardware. In *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security* (New York, NY, USA, USA, 2016), TIS '16, Association for Computing Machinery, p. 43.

[36] De Meyer, L., Bilgin, B., and Reparaz, O. Consolidating security notions in hardware masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2019*, 3 (May 2019), 119–147.

[37] De Santis, F., Schauer, A., and Sigl, G. Chacha20-poly1305 authenticated encryption for high-speed embedded iot applications. In *Proceedings of the Conference on Design, Automation & Test in Europe* (Leuven, BEL, 2017), DATE '17, European Design and Automation Association, p. 692–697.

[38] Delvaux, J. Threshold implementations are not provably secure against fault sensitivity analysis. Cryptology ePrint Archive, Report 2020/400, 2020.

[39] Dobbertin, H., Bosselaers, A., and Preneel, B. RIPEMD-160: A strengthened version of RIPEMD. In *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings* (1996), D. Gollmann, Ed., vol. 1039 of *Lecture Notes in Computer Science*, Springer, pp. 71–82.

[40] Dobraunig, C., Eichlseder, M., Mendel, F., and Schläffer, M. Cryptanalysis of ascon. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings* (2015), K. Nyberg, Ed., vol. 9048 of *Lecture Notes in Computer Science*, Springer, pp. 371–387.

[41] Dyrkolbotn, G. O., Wold, K., and Snekkenes, E. Security implications of crosstalk in switching CMOS gates. In *Information Security* (Berlin, Heidelberg, 2011), M. Burmester, G. Tsudik, S. Magliveras, and I. Ilić, Eds., Springer Berlin Heidelberg, pp. 269–275.

[42] Dziembowski, S., and Pietrzak, K. Leakage-resilient cryptography. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science* (USA, 2008), FOCS '08, IEEE Computer Society, p. 293–302.

[43] Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., and Shalmani, M. T. M. On the power of power analysis in the real world: A complete break of the keeloqcode hopping scheme. In *Advances in Cryptology - CRYPTO 2008* (2008), D. Wagner, Ed., vol. 5157 of *LNCS*, Springer, pp. 203–220.

[44] Endo, S., Li, Y., Homma, N., Sakiyama, K., Ohta, K., and Aoki, T. An efficient countermeasure against fault sensitivity analysis using configurable delay blocks. In *FDTC* (2012), IEEE Computer Society, pp. 95–102.

[45] Endo, S., Li, Y., Homma, N., Sakiyama, K., Ohta, K., Fujimoto, D., Nagata, M., Katashita, T., Danger, J., and Aoki, T. A silicon-level countermeasure against fault sensitivity analysis and its evaluation. *IEEE Trans. VLSI Syst. 23*, 8 (2015), 1429–1438.

[46] Endo, S., Sugawara, T., Homma, N., Aoki, T., and Satoh, A. An on-chip glitchy-clock generator for testing fault injection attacks. *J. Cryptographic Engineering 1*, 4 (2011), 265–270.

[47] Endo, S., Sugawara, T., Homma, N., Aoki, T., and Satoh, A. A configurable on-chip glitchy-clock generator for fault injection experiments. *IEICE Transactions 95-A*, 1 (2012), 263–266.

[48] Faust, S., Grosso, V., Pozo, S. M. D., Paglialonga, C., and Standaert, F. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018*, 3 (2018), 89–120.

[49] Fischer, H. *A History of the Central Limit Theorem*. Springer-Verlag, New York, NY, USA, 2011.

[50] Fischer, W., and Homma, N., Eds. *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings* (2017), vol. 10529 of *Lecture Notes in Computer Science*, Springer.

[51] Fujimoto, D., Nagata, M., Katashita, T., Sasaki, A. T., Hori, Y., and Satoh, A. A fast power current analysis methodology using capacitor charging model for side channel attack evaluation. In *Hardware-Oriented Security and Trust - HOST 2011* (2011), IEEE, pp. 87–92.

[52] Gandolfi, K., Mourtel, C., and Olivier, F. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems - CHES 2001* (2001), Ç. K. Koç, D. Naccache, and C. Paar, Eds., vol. 2162 of *LNCS*, Springer, pp. 251–261.

[53] Genkin, D., Shamir, A., and Tromer, E. Rsa key extraction via low-bandwidth acoustic cryptanalysis. In *Advances in Cryptology – CRYPTO 2014* (Berlin, Heidelberg, 2014), J. A. Garay and R. Gennaro, Eds., Springer Berlin Heidelberg, pp. 444–461.

[54] Gershenfeld, N. *When Things Start to Think*. Henry Holt and Co., Inc., New York, NY, USA, USA, 1999.

[55] Ghalaty, N. F., Aysu, A., and Schaumont, P. Analyzing and eliminating the causes of fault sensitivity analysis. In *DATE* (2014), European Design and Automation Association, pp. 1–6.

[56] Ghoshal, A., and Cnudde, T. D. Several masked implementations of the boyar-peralta AES s-box. In *Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings* (2017), A. Patra and N. P. Smart, Eds., vol. 10698 of *Lecture Notes in Computer Science*, Springer, pp. 384–402.

[57] Gierlichs, B., Batina, L., Tuyls, P., and Preneel, B. Mutual information analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2008* (2008), E. Oswald and P. Rohatgi, Eds., vol. 5154 of *LNCS*, Springer, pp. 426–442.

[58] Giry, D. Cryptographic key lenght recommendation, Jul. 2020.

[59] Goodwill, G., Jun, B., Jaffe, J., and Rohatgi, P. A Testing Methodology for Side-Channel Resistance Validation. NIST non-invasive attack testing workshop, 2011.

[60] Goubin, L., and Patarin, J. DES and differential power analysis (the "duplication" method). In *Cryptographic Hardware and Embedded Systems - CHES'99* (1999), Ç. K. Koç and C. Paar, Eds., vol. 1717 of *LNCS*, Springer, pp. 158–172.

[61] Gross, H., Mangard, S., and Korak, T. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security* (New York, NY, USA, USA, 2016), TIS '16, Association for Computing Machinery, p. 3.

[62] Harris, D., Ho, R., Wei, G.-Y., and Horowitz, M. The fanout-of-4 inverter delay metric.

[63] Hassoune, I., Mace, F., Flandre, D., and Legat, J.-D. Dynamic differential self-timed logic families for robust and low-power security ICs. *INTEGRATION, the VLSI Journal 40* (3 2007), 355–364.

[64] He, M. T., Park, J., Nahiyan, A., Vassilev, A., Jin, Y., and Tehranipoor, M. RTL-PSC: automated power side-channel leakage assessment at register-transfer level. In *37th IEEE VLSI Test Symposium, VTS 2019, Monterey, CA, USA, April 23-25, 2019* (2019), IEEE, pp. 1–6.

[65] Hennessy, J. L., and Patterson, D. A. A new golden age for computer architecture. *Commun. ACM 62*, 2 (Jan. 2019), 48–60.

[66] Huss, S. A., Stöttinger, M., and Zohner, M. AMASIVE: an adaptable and modular autonomous side-channel vulnerability evaluation

framework. In *Number Theory and Cryptography - Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday* (2013), M. Fischlin and S. Katzenbeisser, Eds., vol. 8260 of *Lecture Notes in Computer Science*, Springer, pp. 151–165.

[67] Inc., S. Ccs timing technical white paper copyright notice and proprietary information.

[68] Ishai, Y., Sahai, A., and Wagner, D. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003* (Berlin, Heidelberg, 2003), D. Boneh, Ed., Springer Berlin Heidelberg, pp. 463–481.

[69] Jakushokas, R., Popovich, M., Mezhiba, A. V., Kse, S., and Friedman, E. G. *Power Distribution Networks with On-Chip Decoupling Capacitors*, 2nd ed. Springer Publishing Company, Incorporated, 2010.

[70] Kaeslin, H. *Top-Down Digital VLSI Design: From Architectures to Gate-Level Circuits and FPGAs*, 1st ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[71] Kamel, D., Renauld, M., Flandre, D., and Standaert, F. Understanding the limitations and improving the relevance of SPICE simulations in side-channel security evaluations. *J. Cryptographic Engineering 4*, 3 (2014), 187–195.

[72] Kamel, D., Renauld, M., Flandre, D., and Standaert, F.-X. Understanding the limitations and improving the relevance of spice simulations in side-channel security evaluations. *Journal of Cryptographic Engineering 4*, 3 (Sep 2014), 187–195.

[73] Kerckhoffs, A. La cryptographie militaire, première partie. In *Journal des sciences militaires, vol. IX* (Jan. 1883), pp. 5—38.

[74] Kerckhoffs, A. La cryptographie militaire, seconde partie. In *Journal des sciences militaires, vol. IX* (Feb. 1883), pp. 161—191.

[75] Kim, S., Kosonocky, S. V., and Knebel, D. R. Understanding and minimizing ground bounce during mode transition of power gating structures. In *ISLPED '03.* (Aug 2003), pp. 22–25.

[76] Kirschbaum, M., and Popp, T. Evaluation of power estimation methods based on logic simulations. In *Austrochip 2007* (2007), K.-C. Posch and J. Wolkerstorfer, Eds., Verlag der Technischen Universität Graz, p. 45–51.

[77] Knichel, D., Sasdrich, P., and Moradi, A. SILVER - statistical independence and leakage verification. *IACR Cryptol. ePrint Arch. 2020* (2020), 634.

[78] Knudsen, J. Nangate 45nm open cell library. In *12th Si2/OpenAccess+ Conference* (2008).

[79] Koblitz, N. Elliptic curve cryptosystems. *Mathematics of Computation 48*, 177 (1987), 203–209.

[80] Kocher, P., Horn, J., Fogh, A., , Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., and Yarom, Y. Spectre attacks: Exploiting speculative execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)* (2019).

[81] Kocher, P. C. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology - CRYPTO '96* (1996), N. Koblitz, Ed., vol. 1109 of *LNCS*, Springer, pp. 104–113.

[82] Kocher, P. C., Jaffe, J., and Jun, B. Differential power analysis. In *Advances in Cryptology - CRYPTO '99* (1999), M. J. Wiener, Ed., vol. 1666 of *LNCS*, Springer, pp. 388–397.

[83] Krovetz, T., and Rogaway, P. The OCB Authenticated-Encryption Algorithm. RFC 7253, May 2014.

[84] Leiserson, A. J., Marson, M. E., and Wachs, M. A. Gate-level masking under a path-based leakage metric. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings* (2014), L. Batina and M. Robshaw, Eds., vol. 8731 of *Lecture Notes in Computer Science*, Springer, pp. 580–597.

[85] Leiserson, C. E., Rose, F. M., and Saxe, J. B. Optimizing synchronous circuitry by retiming (preliminary version). In *Third Caltech Conference on Very Large Scale Integration* (Berlin, Heidelberg, 1983), R. Bryant, Ed., Springer Berlin Heidelberg, pp. 87–116.

[86] Levi, I., Bellizia, D., and Standaert, F. Reducing a masked implementation's effective security order with setup manipulations and an explanation based on externally-amplified couplings. *IACR Trans. Cryptogr. Hardw. Embed. Syst. 2019*, 2 (2019), 293–317.

[87] Li, Y., Ohta, K., and Sakiyama, K. Toward effective countermeasures against an improved fault sensitivity analysis. *IEICE Transactions 95-A*, 1 (2012), 234–241.

[88] Li, Y., Sakiyama, K., Gomisawa, S., Fukunaga, T., Takahashi, J., and Ohta, K. Fault sensitivity analysis. In *CHES* (2010), vol. 6225 of *Lecture Notes in Computer Science*, Springer, pp. 320–334.

[89] Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., Horn, J., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., and Hamburg, M. Meltdown: Reading kernel memory from user space. In *27th USENIX Security Symposium (USENIX Security 18)* (2018).

[90] Macé, F., Standaert, F., and Quisquater, J. Information theoretic evaluation of side-channel resistant logic styles. In *Cryptographic Hardware and Embedded Systems - CHES 2007* (2007), P. Paillier and I. Verbauwhede, Eds., vol. 4727 of *LNCS*, Springer, pp. 427–442.

[91] Mangard, S., Oswald, E., and Popp, T. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag, Berlin, Heidelberg, 2007.

[92] Mangard, S., Popp, T., and Gammel, B. M. Side-channel leakage of masked CMOS gates. In *Topics in Cryptology – CT-RSA 2005* (Berlin, Heidelberg, 2005), A. Menezes, Ed., Springer Berlin Heidelberg, pp. 351–365.

[93] Marin, E., Singelée, D., Yang, B., Verbauwhede, I., and Preneel, B. On the feasibility of cryptography for a wireless insulin pump system. In *Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy, CODASPY 2016, New Orleans, LA, USA, March 9-11, 2016* (2016), E. Bertino, R. Sandhu, and A. Pretschner, Eds., ACM, pp. 113–120.

[94] McCann, D., Whitnall, C., and Oswald, E. ELMO: emulating leaks for the ARM cortex-m0 without access to a side channel lab. *IACR Cryptol. ePrint Arch. 2016* (2016), 517.

[95] McGrew, D. A., and Viega, J. The security and performance of the galois/counter mode (gcm) of operation. In *Progress in Cryptology - INDOCRYPT 2004* (Berlin, Heidelberg, 2005), A. Canteaut and K. Viswanathan, Eds., Springer Berlin Heidelberg, pp. 343–355.

[96] Menezes, A., and Vanstone, S. A. Elliptic curve cryptosystems and their implementations. *J. Cryptology 6*, 4 (1993), 209–224.

[97] Menezes, A. J., Vanstone, S. A., and Oorschot, P. C. V. *Handbook of Applied Cryptography*, 1st ed. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[98] Miller, C., and Valasek, C. Adventures in automotive networks and control units. In *Defcon 21, USA* (2013).

[99] Miller, C., and Valasek, C. Remote exploitation of an unaltered passenger vehicle. In *Black Hat, USA* (2015).

[100] Miller, V. S. Use of elliptic curves in cryptography. In *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings* (1985), H. C. Williams, Ed., vol. 218 of *Lecture Notes in Computer Science*, Springer, pp. 417–426.

[101] Mischke, O., Moradi, A., and Güneysu, T. Fault sensitivity analysis meets zero-value attack. In *FDTC* (2014), IEEE Computer Society, pp. 59–67.

[102] Moll, F., Roca, M., and Isern, E. Analysis of dissipation energy of switching digital CMOS gates with coupled outputs. *Microelectronics Journal 34*, 9 (2003), 833 – 842.

[103] Moos, T. Static power SCA of sub-100 nm CMOS asics and the insecurity of masking schemes in low-noise environments. *IACR Trans. Cryptogr. Hardw. Embed. Syst. 2019*, 3 (2019), 202–232.

[104] Moos, T., Moradi, A., and Richter, B. Static power side-channel analysis - an investigation of measurement factors. *IEEE Trans. Very Large Scale Integr. Syst. 28*, 2 (2020), 376–389.

[105] Moradi, A. Side-channel leakage through static power - should we care about in practice? In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings* (2014), L. Batina and M. Robshaw, Eds., vol. 8731 of *Lecture Notes in Computer Science*, Springer, pp. 562–579.

[106] Moradi, A., Mischke, O., and Paar, C. One attack to rule them all: Collision timing attack versus 42 AES ASIC cores. *IEEE Trans. Computers 62*, 9 (2013), 1786–1798.

[107] Moradi, A., Mischke, O., Paar, C., Li, Y., Ohta, K., and Sakiyama, K. On the power of fault sensitivity analysis and collision side-channel attacks in a combined setting. In *CHES* (2011), vol. 6917 of *Lecture Notes in Computer Science*, Springer, pp. 292–311.

[108] Moradi, A., Richter, B., Schneider, T., and Standaert, F. Leakage detection with the x2-test. *IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018*, 1 (2018), 209–237.

[109] Moradi, A., Salmasizadeh, M., Shalmani, M. T. M., and Eisenbarth, T. Vulnerability modeling of cryptographic hardware to power analysis attacks. *Integration, the VLSI Journal 42*, 4 (2009), 468 – 478.

[110] Motassadeq, T. E. Ccs vs nldm comparison based on a complete automated correlation flow between primetime and hspice. In *2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)* (2011), pp. 1–5.

[111] National Institute of Standards and Technology. *FIPS PUB 180-4: Secure Hash Standard*. Federal Inf. Process. Stds. (NIST FIPS), Aug. 2015. Supersedes FIPS PUB 180 2012 March 6.

[112] Nikova, S., Rechberger, C., and Rijmen, V. Threshold implementations against side-channel attacks and glitches. In *ICICS* (2006), vol. 4307 of *Lecture Notes in Computer Science*, Springer, pp. 529–545.

[113] Nikova, S., Rijmen, V., and Schläffer, M. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology 24*, 2 (2011), 292–321.

[114] Noorman, J., Bulck, J. V., Mühlberg, J. T., Piessens, F., Maene, P., Preneel, B., Verbauwhede, I., Götzfried, J., Müller, T., and Freiling, F. C. Sancus 2.0: A low-cost security architecture for iot devices. *ACM Trans. Priv. Secur. 20*, 3 (2017), 7:1–7:33.

[115] Oswald, D., and Paar, C. Breaking mifare desfire MF3ICD40: power analysis and templates in the real world. In *Cryptographic Hardware and Embedded Systems - CHES 2011* (2011), B. Preneel and T. Takagi, Eds., vol. 6917 of *LNCS*, Springer, pp. 207–222.

[116] Popp, T., Kirschbaum, M., Zefferer, T., and Mangard, S. Evaluation of the masked logic style mdpl on a prototype chip. In *Cryptographic Hardware and Embedded Systems - CHES 2007* (Berlin, Heidelberg, 2007), P. Paillier and I. Verbauwhede, Eds., Springer Berlin Heidelberg, pp. 81–94.

[117] Poschmann, A., Moradi, A., Khoo, K., Lim, C., Wang, H., and Ling, S. Side-channel resistant crypto for less than 2, 300 GE. *J. Cryptology 24*, 2 (2011), 322–345.

[118] Pozo, S. M. D., Standaert, F., Kamel, D., and Moradi, A. Side-channel attacks from static power: when should we care? In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition,*

DATE 2015, Grenoble, France, March 9-13, 2015 (2015), W. Nebel and D. Atienza, Eds., ACM, pp. 145–150.

[119] Quisquater, J., and Samyde, D. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *E-smart* (2001), vol. 2140 of *Lecture Notes in Computer Science*, Springer, pp. 200–210.

[120] Rabaey, J. M., Chandrakasan, A., and Nikolić, B. *Digital Integrated Circuits*, 3rd ed. Prentice Hall Press, USA, 2008.

[121] Regazzoni, F., Cevrero, A., Standaert, F., Badel, S., Kluter, T., Brisk, P., Leblebici, Y., and Ienne, P. A design flow and evaluation framework for DPA-resistant instruction set extensions. In *Cryptographic Hardware and Embedded Systems - CHES 2009* (2009), C. Clavier and K. Gaj, Eds., vol. 5747 of *LNCS*, Springer, pp. 205–219.

[122] Reparaz, O. Detecting flawed masking schemes with leakage detection tests. In *Fast Software Encryption - FSE 2016* (2016), T. Peyrin, Ed., vol. 9783 of *LNCS*, Springer, pp. 204–222.

[123] Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., and Verbauwhede, I. Consolidating masking schemes. In *Advances in Cryptology - CRYPTO 2015* (2015), R. Gennaro and M. Robshaw, Eds., vol. 9215 of *LNCS*, Springer, pp. 764–783.

[124] Reparaz, O., Gierlichs, B., and Verbauwhede, I. Fast leakage assessment. In *Cryptographic Hardware and Embedded Systems - CHES 2017* (2017), W. Fischer and N. Homma, Eds., vol. 10529 of *LNCS*, Springer, pp. 387–399.

[125] Rivain, M., and Prouff, E. Provably secure higher-order masking of AES. In *Cryptographic Hardware and Embedded Systems, CHES 2010* (Berlin, Heidelberg, 2010), S. Mangard and F.-X. Standaert, Eds., Springer Berlin Heidelberg, pp. 413–427.

[126] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 21*, 2 (Feb. 1978), 120–126.

[127] Rogaway, P. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002* (2002), V. Atluri, Ed., ACM, pp. 98–107.

[128] Ronen, E., Shamir, A., Weingarten, A., and O'Flynn, C. IoT goes nuclear: Creating a ZigBee chain reaction. In *2017 IEEE Symposium on Security and Privacy (SP)* (2017), pp. 195–212.

[129] Sasdrich, P., Bilgin, B., Hutter, M., and Marson, M. E. Low-latency hardware masking with application to AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020*, 2 (2020), 300–326.

[130] Schellenberg, F., Finkeldey, M., Gerhardt, N., Hofmann, M., Moradi, A., and Paar, C. Large laser spots and fault sensitivity analysis. In *HOST* (2016), IEEE Computer Society, pp. 203–208.

[131] Schlösser, A., Nedospasov, D., Krämer, J., Orlic, S., and Seifert, J.-P. Simple photonic emission analysis of AES. In *Cryptographic Hardware and Embedded Systems – CHES 2012* (Berlin, Heidelberg, 2012), E. Prouff and P. Schaumont, Eds., Springer Berlin Heidelberg, pp. 41–57.

[132] Schneider, T., and Moradi, A. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In *Cryptographic Hardware and Embedded Systems - CHES 2015* (2015), T. Güneysu and H. Handschuh, Eds., vol. 9293 of *LNCS*, Springer, pp. 495–513.

[133] Schneider, T., Moradi, A., and Güneysu, T. Parti - towards combined hardware countermeasures against side-channel and fault-injection attacks. In *CRYPTO (2)* (2016), vol. 9815 of *Lecture Notes in Computer Science*, Springer, pp. 302–332.

[134] Shamir, A. Protecting smart cards from passive power analysis with detached power supplies. In *Cryptographic Hardware and Embedded Systems — CHES 2000* (Berlin, Heidelberg, 2000), Ç. K. Koç and C. Paar, Eds., Springer Berlin Heidelberg, pp. 71–77.

[135] Shor, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994), pp. 124–134.

[136] Standaert, F. How (not) to use Welch's t-test in side-channel security evaluations. In *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers* (2018), B. Bilgin and J. Fischer, Eds., vol. 11389 of *Lecture Notes in Computer Science*, Springer, pp. 65–79.

[137] Standaert, F., Malkin, T., and Yung, M. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology - EUROCRYPT 2009* (2009), A. Joux, Ed., vol. 5479 of *LNCS*, Springer, pp. 443–461.

[138] Statista. Statista dossier about the Internet of Things (IOT). Accessed October, 2019.

[139] Tao, B., and Wu, H. Improving the biclique cryptanalysis of AES. In *Information Security and Privacy* (Cham, 2015), E. Foo and D. Stebila, Eds., Springer International Publishing, pp. 39–56.

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

[141] Tiri, K., Akmal, M., and Verbauwhede, I. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Proceedings of the 28th European Solid-State Circuits Conference* (2002), pp. 403–406.

[142] Tiri, K., and Verbauwhede, I. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *Design, Automation and Test in Europe - DATE 2004)* (2004), IEEE Computer Society, pp. 246–251.

[143] Tiri, K., and Verbauwhede, I. Simulation models for side-channel information leaks. In *Design Automation Conference - DAC 2005* (2005), W. H. J. Jr., G. Martin, and A. B. Kahng, Eds., ACM, pp. 228–233.

[144] Tiri, K., and Verbauwhede, I. A VLSI design flow for secure side-channel attack resistant ICs. In *Design, Automation and Test in Europe* (March 2005), pp. 58–63 Vol. 3.

[145] Tiri, K., and Verbauwhede, I. A digital design flow for secure integrated circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems 25*, 7 (2006), 1197–1208.

[146] Trichina, E. Combinational Logic Design for AES SubByte Transformation on Masked Data. Cryptology ePrint Archive, Report 2003/236, 2003.

[147] Wanlass, F., and Sah, C. Nanowatt logic using field-effect metal-oxide semiconductor triodes. In *1963 IEEE International Solid-State Circuits Conference. Digest of Technical Papers* (1963), vol. VI, pp. 32–33.

[148] Wegener, F., and Moradi, A. A first-order sca resistant AES without fresh randomness. Cryptology ePrint Archive, Report 2018/172, 2018.

[149] Wei Zhao, and Yu Cao. New generation of predictive technology model for sub-45nm design exploration. In *7th International Symposium on Quality Electronic Design (ISQED'06)* (March 2006), pp. 6 pp.–590.

[150] Wu, H., and Preneel, B. Aegis: A fast authenticated encryption algorithm. Cryptology ePrint Archive, Report 2013/695, 2013.

[151] Zussa, L., Exurville, I., Dutertre, J.-M., Rigaud, J.-B., Robisson, B., Tria, A., and Clédière, J. Evidence of an information leakage between logically independent blocks. In *Proceedings of the Second Workshop on Cryptography and Security in Computing Systems* (2015), CS2 '15, pp. 25:25–25:30.

List of publications

[152] T. Ashur, R. Posteuca, D. Šijačić, and S. D'haeseleer, "Generalized Matsui algorithm 1 with application for the full DES," in *Security and Cryptography for Networks*, C. Galdi and V. Kolesnikov, Eds. Cham: Springer International Publishing, 2020, pp. 448–467.

[153] D. Šijačić, J. Balasch, B. Yang, S. Ghosh, and I. Verbauwhede, "Towards efficient and automated side-channel evaluations at design time," *Journal of Cryptographic Engineering*, p. 15, 2020.

[154] D. Šijačić, J. Balasch, and I. Verbauwhede, "Sweeping for leakage in masked circuit layouts," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 915–920.

[155] D. Šijačić, J. Balasch, B. Yang, S. Ghosh, and I. Verbauwhede, "Towards efficient and automated side channel evaluations at design time," in *PROOFS 2018, 7th International Workshop on Security Proofs for Embedded Systems, colocated with CHES 2018, Amsterdam, The Netherlands, September 13, 2018*, ser. Kalpa Publications in Computing, L. Batina, U. Kühne, and N. Mentens, Eds., vol. 7. EasyChair, 2018, pp. 16–31.

[156] V. Arribas, T. De Cnudde, and D. Šijačić, "Glitch-resistant masking schemes as countermeasure against fault sensitivity analysis," in *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2018, pp. 27–34.

[157] D. Šijačić, A. B. Kidmose, B. Yang, S. Banik, B. Bilgin, A. Bogdanov, and I. Verbauwhede, "Hold your breath, PRIMATEs are lightweight," in *Selected Areas in Cryptography – SAC 2016*, R. Avanzi and H. Heys, Eds. Cham: Springer International Publishing, 2017, pp. 197–216.

[158] S. Picek, D. Sisejkovic, D. Jakobovic, L. Batina, B. Yang, D. Šijačić, and N. Mentens, "Extreme pipelining towards the best area-performance

trade-off in hardware," in *Progress in Cryptology - AFRICACRYPT 2016 - 8th International Conference on Cryptology in Africa, Fes, Morocco, April 13-15, 2016, Proceedings*, ser. Lecture Notes in Computer Science, D. Pointcheval, A. Nitaj, and T. Rachidi, Eds., vol. 9646. Springer, 2016, pp. 147–166.

Appendix A

Data Formats

In this short chapter we present the specification of the two data formats designed in this thesis. Both formats are designed to support efficient and automated SCA evaluations. This is a form of vertical integration, as they can embed design and evaluation decisions. They are optimized for performance. However, if a larger flexibility is needed we recommend using the Hierarchical Data Format (HDF5).

We represent the data formats using a C-like pseudo code. Variable length vectors, denoted using `data_type elements<n>;` are encoded as `uint64 n; data_type *elements;`.

Power Frame File (PFF) described in Listing A.1 is the custom binary used for storing parsed traces.

Listing A.1: Power Frame File encoding

```
struct PFF {
    frame_type                  type;
    uint32                      frame_period;
    uint64                      max_events_in_frame;
    TaggedField                 extras<n>;
    Frame                       frames<n_frames>;
}
```

Tagged fields, described in Listing A.2 can be used to embed the communication between the tools. For example, a tagged field can be used to instruct an analyzer to reconstruct the frame data as a MSM or as a rectangular waveform produced by PT.

Listing A.2: Tagged field encoding.

```
struct TaggedField { // Extended functionality.
    uint16                      tag;
    uint08                      encoded<n_bytes>
}
```

Each PFF can store frames of a single type, denoted by type `type` field. Frame structure is described in Listing A.3, along with the type encodings in Listing A.4.

Listing A.3: Frame encoding.

```
struct Frame {
    DataVector                  data<n_data>;
    switch (type) {
        uint32                  time_samples<n_events>;
        case UI32FL32: // Default for simulations.
            float32             values<n_events>;
            ;
        case UI32FL64:
            float64             values<n_events>;
            ;
        case UI32UI08:
            uint08              values<n_events>;
            ;
        case UI32UI16:
            uint16              values<n_events>;
            ;
    }
}
```

Listing A.4: Frame type encodings.

```
typedef enumerate { // X-axis type, Y-axis type
    UI32FL32                    = 0x00,
    UI32FL64                    = 0x01,
    UI32UI08                    = 0x08,
    UI32UI16                    = 0x09
} frame_type;
```

Data vectors are variable length byte vectors, described in Listing A.5.

Listing A.5: Data vector encoding.

```
struct DataVector {
    uint08                      bytes<n_bytes>;
}
```

Analyzers input PFF files and output Analyzed Frame Data (AFD) files, described in Listing A.6. Similarly to the PFF files, a number of tagged fields can be used to alter the behavior, *e.g.* output formatting or additional processing such as adding gaussian noise. Context of any analyzer can be encoded and stored in case more traces need to be added to the computation. As PFF files store traces from all inputs in the same time-value format, desired waveform after the reconstruction has to be communicated to the analyzers—either through a command line argument or via a tagged field in the PFF file. While we mostly use the AFD files for storing and visualizing results of the analysis, such an encoding schemes is intended to minimize the amount of human error.

Listing A.6: Analyzed Frame Data encoding.

```
struct AFD {
    TaggedField             extras<n>;
    AnalyzerContext         context; // Analyzer-dependent.
    waveform_type           waveform; // Reconstruction.
    float64                 x_resolution;
    float64                 y_resolution;
}
```

Encodings for the waveform reconstruction functions are given in Listing A.7

Listing A.7: Frame type encodings.

```
typedef enumerate { // X-axis type, Y-axis type
    MSM                     = 0x00, // Marching sticks
    REC                     = 0x01, // Rectangular waveform
    PWL                     = 0x08, // Piece-wise linear
    ANA                     = 0xff  // Analog waveform
} waveform_type;
```

Lastly, we give an example of an analyzer context, in particular the third-order TVLA in Listing A.8.

Listing A.8: Third-order TVLA analyzer context.

```
struct CtxTvla1 {
    uint64                  n_samples;
    uint64                  n_epochs;
    EpochTvla1              epochs[n_epochs];
}
```

Multiple epochs can be stored for acquiring the evolution traces. Format of an epoch for the third-order TVLA is described in Listing A.9.

Listing A.9: Third-order TVLA analyzer epoch.

```
struct EpochTvla1 {
    // Set 0
    uint64                  n_0; // Number of traces
    float64                 mean_0[n_samples];
    float64                 cs2_0[n_samples];
    float64                 cs3_0[n_samples];
    float64                 cs4_0[n_samples];
    float64                 cs5_0[n_samples];
    float64                 cs6_0[n_samples];
    // Set 1
    uint64                  n_1; // Number of traces
    float64                 mean_1[n_samples];
    float64                 cs2_1[n_samples];
    float64                 cs3_1[n_samples];
    float64                 cs4_1[n_samples];
    float64                 cs5_1[n_samples];
    float64                 cs6_1[n_samples];
}
```

Here, each “cs” field is a non-normalized central moment as defined in [132].

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
COSIC
Kasteelpark Arenberg 10, box 2452
B-3001 Leuven