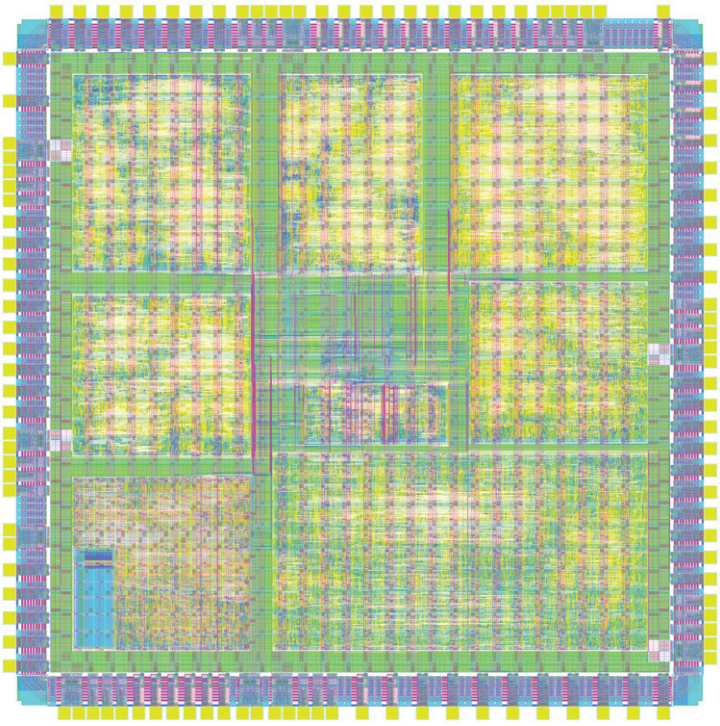
Graz University of Technology
Faculty of Computer Science
Institute for Applied Information Processing and Communications
IAIK

Masking at the Cell Level –
A Hardware Countermeasure against Power Analysis Attacks for Cryptographic Devices



Assessors:
Prof. Dr. Karl Christian Posch
Prof. Dr. Patrick R. Schaumont

A PhD Thesis Presented to the Faculty of Computer Science in Fulfillment of the Requirements for the PhD Degree

March 2010

by **Thomas Popp**

Masking at the Cell Level - A Hardware Countermeasure against Power Analysis Attacks for Cryptographic Devices

by

Thomas Popp

A PhD Thesis
Presented to the Faculty of Computer Science in Partial Fulfillment of the Requirements for the PhD Degree

Assessors

Prof. Dr. Karl Christian Posch (TU Graz, Austria)
Prof. Dr. Patrick R. Schaumont (Virginia Tech, USA)

March 2010

Institute for Applied Information Processing and Communications (IAIK)
Faculty of Computer Science
Graz University of Technology, Austria

Abstract

Implementation attacks and in particular its subtype power analysis (PA) attacks pose a serious threat to IT-devices like cryptographic smart cards or USB tokens. This thesis focuses on countermeasures against PA attacks at the cell level. Special PA-resistant logic styles are employed to avoid any leakage of information via the power consumption signal that could be exploited by an attacker.

Three PA-resistant logic styles are presented in this thesis: mCMOS, MDPL, and iMDPL. They all follow the masking approach to make them resistant against PA attacks. The main incentive for choosing this approach was the assumption that masking can help to avoid any hard to fulfill constraints in the implementation phase of the cryptographic circuits. Various tough constraints burden many other PA-resistant logic styles that have been proposed so far.

Masking at the cell level is explained in general and the three proposed masked logic styles are introduced in detail. Subsequently, the evaluation results and insights obtained for them in particular with respect to their PA resistance are presented and discussed. The evaluations are based on theoretical considerations, (power) simulation results, and measurement data acquired from prototype chips containing circuits implemented in the masked logic styles. It is shown that the main effects that influence the PA resistance of the masked logic styles in a negative way are signal glitches and an early propagation behavior of the PA-resistant logic cells. Also other, more advanced attacks on the masked logic styles are discussed.

Furthermore, it is presented how PA-resistant logic styles following the masking approach can be used in the semi-automatic design flows that are very common to implement digital circuits. Another issue that is related to the design and implementation of digital circuits is the simulation of their power consumption. In the context of PA attacks, it is discussed how power simulations are suitable to estimate the PA resistance of cryptographic circuits. The main issue is to find the right balance between accuracy to consider all relevant effects and resource requirements (memory, time) to allow enough simulation runs in adequate time.

Acknowledgements

I would like to thank all people who supported me during the last years and with whom I had the pleasure to work with. In particular, I want to thank Martin Feldhofer, Stefan Tillich, and Christian Rechberger who have been very good friends and colleagues during my studies and the work at IAIK. I would not have made it without their support and motivation.

Since research never happens in isolation, I also want to dedicate special thanks to all my coauthors and partners who have contributed in one way or the other to my work. Stefan Mangard has already guided me in my early years of research and since then we have done lots of interesting things together. Besides the scientific work this also included great mountain bike tours. I also want to thank Elisabeth Oswald for her help and support. Together with Elisabeth and Stefan, I had the pleasure to write the first book about differential power analysis. Well, the "pleasure" only really appeared after we finished the book, but in retrospect the effort was definitely worth it. In the last years, Mario Kirschbaum became another important colleague and coauthor, so I also want to thank him.

I am also very thankful to all my colleagues from IAIK and especially the VLSI & Security group. In particular, I want to thank Manfred Aigner, who, as the group leader, tirelessly run interference for us in administrative and project management matters so that we could concentrate on our research work. By the way, my experience has shown that it is usually easier to tire him out with sporting activities like mountain biking or hiking. I also want to thank Karl Christian Posch for being my advisor and also the first assessor for this thesis. I really enjoyed the various non-technical discussions we had, especially the "tougher" ones. Last but not least, I also want to thank all the other former and current members of the VLSI & Security group for their help and collegiality: Alexander Szekely, Christoph Herbst, Jörn-Marc Schmidt, Johannes Wolkerstorfer, Johann Großschädl, Martin Schläffer, Marcel Medwed, Michael Hutter, Sandra Dominikus, and Thomas Plos.

I also want to take the opportunity to thank my second assessor Patrick Schaumont for taking the time to travel to Graz.

Finally and most importantly, I want to thank my parents and my wife Silke. Without their support and understanding, this all would not have been possible.

Thomas Popp
Graz, March 2010

Table of Contents

List of Tables

List of Figures

Acronyms

AC	Alternating Current
ADLBL	Asynchronous Directional Latch-Based Logic
AES	Advanced Encryption Standard
ASIC	Application-Specific Integrated Circuit
BSIM	Berkeley Short-Channel IGFET Model
CBC	Cipher Block Chaining
CML	Current-Mode Logic
CMOS	Complementary Metal-Oxide Semiconductor
DEMA	Differential Electro-Magnetic Analysis
DFF	D-Flip-Flop
DM	Difference of Means
DME	Difference of Mean Energies
DPA	Differential Power Analysis
DPDN	Differential Pull-Down Network
DRP	Dual-Rail Precharge
DRSL	Dual-Rail Random-Switching Logic
DSDR	Dual-Spacer Dual-Rail
DyCML	Dynamic Current-Mode Logic
DWDDL	Double Wave Dynamic Differential Logic
ECB	Electronic Code Book
EDA	Electronic Design Automation
EM	Electro-Magnetic
EMA	Electro-Magnetic Analysis
EPDU	Evaluation-Precharge Detection Unit
FA	Fault Analysis
FF	Flip-Flop
FPGA	Field-Programmable Gate Array
GALS	Globally-Asynchronous Locally-Synchronous
GE	Gate Equivalent
GF	Galois Field
GFL	Gammel-Fischer Logic
HD	Hamming Distance
HDL	Hardware Description Language
HW	Hamming Weight
IAIK	Institute for Applied Information Processing and Communications
IC	Integrated Circuit

IEEE	Institute of Electrical and Electronics Engineers
IGFET	Insulated-Gate Field-Effect Transistor
IMAJ	Inverting Majority
iMDPL	Improved Masked Dual-Rail Precharge Logic
I/O	Input/Output
IRAM	Internal RAM
IT	Information Technology
IWDDL	Isolated Wave Dynamic Differential Logic
MAJ	Majority
mAND	Masked AND
MC	MixColumns
MCML	MOS Current-Mode Logic
mCMOS	Masked CMOS
MDPL	Masked Dual-Rail Precharge Logic
mFF	Masked Flip-Flop
mNAND	Masked NAND
mNOR	Masked NOR
mOR	Masked OR
MOS	Metal-Oxide Semiconductor
mXNOR	Masked XNOR
mXOR	Masked XOR
OA	OpenAccess
PA	Power Analysis
PC	Personal Computer
PDF	Probability Density Function
PDN	Pull-Down Network
PLI	Programming Language Interface
PMRML	Precharged Masked Reed-Muller Logic
PRNG	Pseudo-Random Number Generator
PUN	Pull-Up Network
RAM	Random-Access Memory
RFID	Radio-Frequency Identification
ROM	Read-Only Memory
RS	Recommended Standard
RSA	Rivest-Shamir-Adleman
RSL	Random Switching Logic
RTL	Register-Transfer Level
SABL	Sense-Amplifier Based Logic
SAL	Secure Adiabatic Logic
SB	SubBytes
SCA	Side-Channel Analysis
SEMA	Simple Electro-Magnetic Analysis
SHA	Secure Hash Algorithm
SNR	Signal-to-Noise Ratio
SPA	Simple Power Analysis

SPICE	Simulation Program with Integrated Circuit Emphasis
SR	Single-Rail
TA	Timing Analysis
TDPL	Three-Phase Dual-Rail Precharge Logic
3sDL	3-State Dynamic Logic
TRNG	True-Random Number Generator
TSPL	Three-Phase Single-Rail Precharge Logic
TTG	Transition-Trace Generator
UMC	United Microelectronics Corporation
USB	Universal Serial Bus
VCD	Value Change Dump
vcd2pt	VCD File To Power Trace
VCS	Verilog Compiler Simulator
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large-Scale Integration
WDDL	Wave Dynamic Differential Logic
XNOR	Exclusive NOR
XOR	Exclusive OR
XRAM	External Random-Access Memory
ZV	Zero Value

1 Introduction

One main stimulus for the notable changes in our society during the last century has been without question the inventions and developments in the field of *information technology*. Information technology is basically concerned with all aspects of computer-aided generation, storage, processing, transmission, and retrieval of information. In many cases, this information and the related data is sensitive in one way or the other and must thus be protected. The branch of information technology that deals with this protection is commonly called *information security*.

The basic building blocks to achieve information security are *cryptographic algorithms and protocols*. They describe in detail how sensitive data is processed in order to protect it against potential threats like unrestricted accessibility, modification, and forgery. They also regulate how communication entities exchange the data.

Cryptographic algorithms and protocols are specified in an abstract way. For a long time, only these immaterial descriptions have been analyzed for their resistance against all types of increasingly sophisticated mathematical attacks. In these analyses, the internal activities of the algorithms and protocols have been typically hidden in a black box. Any influences of them on the environment have been neglected and only the regular input and output values have been considered.

However, cryptographic algorithms and protocols are executed by mechanical or electronic devices—the so-called cryptographic devices. The execution on physical devices on the one hand influences the environment and on the other hand can also be influenced by the environment. This fact opened the way for a radically new class of attacks summarized with the term *implementation attacks*. The important difference to the classical mathematical attacks is that

the physical "emissions" of intermediate results of cryptographic algorithms and protocols and also the influenceability of the intermediate results from outside the device are included in the attacks. The emissions related to the processing of intermediate results are usually available continuously during the execution of the algorithms and protocols. In the same way, external interferences can be potentially set at all moments in time during the execution. This is the most important novelty of implementation attacks and is the reason for their strength.

A specific implementation attack that exploits the influence of a cryptographic device on its environment during operation is *power analysis* (PA). In a PA attack, the power consumption of the digital circuit on the device is analyzed to draw conclusions about the processed data like the secret key of an encryption algorithm. Many countermeasures against all types of implementation attacks have been developed. In the case of PA attacks, countermeasures aim at reducing and removing the exploitable information in the power consumption signal of cryptographic circuits. In general, the countermeasures can be applied at all design levels of the circuits like the circuit architecture level or the cell level.

This thesis focuses on countermeasures against PA attacks on the cell level. We have made proposals for PA-resistant logic styles and have analyzed our proposals and others for their efficiency. We have also incorporated our PA-resistant logic styles in existing, semi-automatic chip design flows. In a similar way as logic simulations are used to verify the functionality of digital circuits, all types of power simulations can be employed already during the design and implementation phases of cryptographic circuits to assess their resistance against PA attacks. We have also contributed to this simulation and evaluation methodology in the course of the work for this thesis.

1.1 Contributions of This Thesis

The main topic of research covered by this thesis has been the development and improvement of PA-resistant logic styles. At the beginning of our work, we analyzed various PA-resistant logic styles for their working principles, properties, and efficiency. Based on the various shortcomings we identified for existing PA-resistant logic styles, we formulated two main goals for our own proposals. First, the protection achieved by the PA-resistant logic styles should not rely on tough constraints that have to be observed in the implementation phase, e.g. pairs of wires with identical capacitive loads. Second, the cells of the PA-resistant logic styles should be realizable with commonly available CMOS standard cells in order to significantly reduce the implementation effort.

The first proposal of a PA-resistant logic style, developed together with Stefan Mangard, was "masked CMOS" (mCMOS). We found out that the basic functionality of the mCMOS cells can be neatly implemented based on commonly available CMOS standard cells. Furthermore, we also assumed from the results of a first security analysis that mCMOS is secure without the need of obeying tough constraints during implementation. However, we soon identified a major problem of masked circuits: glitches significantly reduce their PA resis-

tance [MPG05]. This publication has been joint work with Stefan Mangard and Berndt Gammel. Because we made the findings about glitches already during the development of mCMOS, we never published this logic style.

In a next step, Masked Dual-Rail Precharge Logic (MDPL) was developed in cooperation with Stefan Mangard. MDPL cells can still be implemented with normal CMOS cells and glitches are prevented by design [PM05]. We also investigated how MDPL circuits can be switched from secure to insecure mode to reduce its power consumption and increase its speed [PM06]. MDPL and its application in semi-automatic chip design flows have also been included in a book about power analysis attacks written together with Stefan Mangard and Elisabeth Oswald [MOP07]. In a joint publication together with Egon Valentini, Edmund Haselwanter, and Robert Ulmer it was analyzed how semi-automatic chip design flows using PA-resistant logic styles profit from the standardized data representation and access interface known as the OpenAccess framework [VHUP05].

In order to get further insights, a prototype chip using MDPL was developed and implemented. Unfortunately, the evaluation of this chip showed that MDPL suffers significantly from data-dependent timing issues caused by the early-propagation behavior of the MDPL cells [PKZM07]. These findings have been the result of joint work together with Mario Kirschbaum, Thomas Zefferer, and Stefan Mangard. A more detailed analysis of the MDPL prototype chip together with Mario Kirschbaum showed that the early propagation problem does not nullify the PA resistance of MDPL in all cases [PKM09].

As a result of the early-propagation issue, "improved MDPL" (iMDPL) was developed [PKZM07]. It avoids the early propagation behavior of the cells but has very high area requirements and consumes a lot of power. An iMDPL prototype chip has been implemented and evaluated together with Mario Kirschbaum [KP09].

Together with Stefan Mangard and later Mario Kirschbaum, various approaches to simulate the power leakage that is exploitable in PA attacks have been analyzed. We built our own tool that performs such a power estimation and evaluated its efficiency with respect to speed, memory requirements, and accuracy [KP07].

1.2 Structure of This Thesis

In the following, we briefly outline the chapters of this thesis.

Chapter 2 starts with a general introduction to implementation attacks, which are used to target all types of cryptographic devices. Subsequently, a specific type of implementation attacks called power analysis (PA) attacks is discussed. Since the power consumption of digital circuits plays an important role in PA attacks, it is reviewed why and how such circuits actually consume power from the supply. Finally, an introduction is given to common countermeasures against PA attacks. Various proposals of PA-resistant logic styles are presented.

Chapter 3 deals with the simulation of the power consumption of cryptographic devices with the specific purpose to estimate their resistance to PA attacks. First, the general approaches to power simulation are investigated to find out how well they are suited for the estimation. Also common high-level power models are discussed, which can be used to translate activities in digital circuits to (estimated) power consumption values. Due to the findings of the general investigation, a power estimation approach based on logic simulation has been chosen. This approach is described in detail and also tools are presented that work accordingly. An example PA attack is shown that works with simulated power traces delivered by one of the tools.

Chapter 4 presents the first masked logic style we proposed to counteract PA attacks at the cell level. It is called "mCMOS". After an introduction to the general concept of masking at the cell level, mCMOS cells and circuits are presented in detail. Then, the important finding that signal glitches significantly reduce the PA resistance of masked logic styles like mCMOS is discussed from a theoretical point of view and also based on practical simulation results.

Chapter 5 introduces the second masked logic style proposed by us called "MDPL". It avoids glitches by design and should therefore have a significant better PA resistance than mCMOS. After a detailed discussion of the functionality and properties of MDPL cells and circuits, a variant of MDPL is presented that allows increasing speed and reducing power consumption by switching to an insecure operation mode while this is unproblematic. Next, it is discussed how PA-resistant logic styles and especially masked logic styles can be used in semi-custom chip design flows.

Chapter 6 is dedicated to the results of the thorough theoretical and practical evaluations of MDPL. First, it is analyzed how the PA resistance of MDPL is influenced by unbalanced complementary wires. A main design goal of MDPL was to be unaffected by such a situation. Next, the early propagation problem of MDPL is investigated, which has been identified by other research groups after the publication of the logic style. Also the results from our own MDPL prototype chip are shown, which clearly prove the existence of the early propagation problem of MDPL in practice. However, it is also shown with the help of the prototype chip that the early propagation problem does not nullify the PA resistance of MDPL in all cases. Finally, the results of a more advanced attack on MDPL called the "PDF-attack" are shown and further issues of MDPL are discussed.

Chapter 7 presents an improved version of MDPL called "iMDPL" which aims at preventing the early propagation behavior of the cells. After a general introduction to iMDPL, the results of a basic security analysis are presented. Next, practical results gained from the analysis of an iMDPL prototype chip are shown. The evaluation results show that iMDPL indeed reduces the problem of early propagation behavior. It shows a significant improved PA resistance compared to MDPL.

Chapter 8 contains the conclusions about our work on PA-resistant logic styles and the simulation of the power leakage of cryptographic circuits.

2

Power Analysis Attacks and Countermeasures

In this chapter, we give an overview of power analysis attacks and describe how they are used to reveal the secrets of cryptographic devices like smart cards. In the beginning, we introduce *implementation attacks*, which is the general class of attacks where power analysis (PA) belongs to. We have published similar introductions to implementation attacks in the paper [Pop09] and about PA attacks in the article [PMO07]. We have given a comprehensive description of PA attacks and related countermeasures in the book [MOP07], which was published in 2007 by Springer.

The power consumption of cryptographic devices is of key importance in the context of PA attacks. We describe in the following how and why digital circuits in general and cryptographic devices in particular consume power, which can then be measured and exploited by an attacker.

In the last part of this chapter, we introduce different types of countermeasures that were proposed so far to counteract PA attacks. The focus of this part lies on countermeasures at the cell level of digital circuits because this is also the main field of contribution of our work presented in this thesis.

2.1 An Introduction to Implementation Attacks

Implementation attacks pose a serious threat to the security of cryptographic algorithms and protocols. In such attacks, not the abstract, mathematical descriptions of cryptographic methods are attacked but their practical realizations in cryptographic devices. This opens up a wide range of powerful attacks, which

are described in the following. We start with a very brief introduction to cryptography and the related main classes of attacks. Subsequently, we focus our introduction on implementation attacks and its various subtypes.

2.1.1 Cryptography

While cryptography tries to protect sensitive data from disclosure and manipulation, various types of attacks aim at circumventing this protection. Implementation attacks are a relatively new and very powerful kind of attack on cryptographic methods.

Whenever sensitive data is exchanged between different entities, a subset of the following main aspects of information security is of importance. Usually, only the intended recipients should be able to read and understand the transmitted data (*confidentiality*). In many cases, it must be verifiable that the data was not altered on its way from sender to receiver (*integrity*) or who the originator of the data actually is (*authenticity*). In other cases, the sender must not be able to deny being the originator of the transmitted information (*non-repudiation*). And also, there are more and more situations where the privacy of sender and receiver must be ensured (*anonymity*).

Cryptography [MvOV97] provides a comprehensive set of methods to achieve these security goals. We generally distinguish between *cryptographic algorithms* that either use no key, the same key (*symmetric-key* methods), or different keys (*asymmetric-key* methods) for the communicating entities. In case of symmetric-key algorithms, the same or a related *secret key* must be transferred in a confidential manner to the receivers, which is the major drawback of this approach. Asymmetric-key algorithms use a key pair: the *private key* and the *public key*. Only the private key, which stays with the sender, must be kept secret. The public key can be transmitted openly to the other communication entities.

An important property of cryptographic algorithms is that their strength should not rely on the fact that the details of a particular algorithm are kept secret. This is known as *Kerckhoffs' principle* [Ker83a, Ker83b] and ultimately means that it is sufficient to protect only the secret or private key from disclosure.

The security goal of confidentiality is typically achieved with the help of *encryption* algorithms. An encryption algorithm takes as inputs the message to be protected (*plaintext*) and the secret or public key. It produces a *ciphertext* as output, which can only be decrypted by an entity that also knows the secret key or possesses the related private key. A very common symmetric-key encryption algorithm today is the *Advanced Encryption Standard* (AES) [Nat01]. To provide the security goal of data integrity, one-way functions (*hash* functions) are usually an important building block. Such functions "compress" data blocks of arbitrary length to blocks with a fixed length. Common hash algorithms are the *Secure Hash Algorithm 1* (SHA-1) and its successor SHA-2 [Nat08]. For authentication and non-repudiation, *digital signatures* are used. An example for them is the *Rivest-Shamir-Adleman* (RSA) signature algorithm [RSA78].

2.1.2 Attacks on Cryptographic Methods in General

The ultimate goal of attacks against cryptographic algorithms is to reveal secret (and private) keys. *Cryptanalytic attacks* aim at cryptographic algorithms directly (*classical cryptanalysis*) or at their implementations in a device (*implementation attacks*). Other approaches to get information about secret keys such as *keystroke logging* or *social engineering* are not considered as cryptanalytic attacks. Social engineering means that people are manipulated so that they are prepared to give away secret credentials like passwords (*phishing*) or that publicly available information about them is collected and analyzed to get hints on their potential passwords. Especially in times of Facebook & Co it became quite easy to acquire such personal data [Tho08].

In classical cryptanalytic attacks, the structure of cryptographic algorithms and their input and output values are analyzed mathematically for weaknesses. In the context of encryption, such a weakness might for example allow to recover (parts of) the plaintexts if a certain number of ciphertexts is available. In the best case, a cryptanalytic attack allows to deduce the secret key. Many different methods of classical cryptanalysis are known, such as *linear* and *differential cryptanalysis* [BS93]. A reasonable cryptanalytic attack on a particular cryptographic algorithm has a lower complexity (time, memory) than a *brute-force attack* on the same algorithm. In a brute-force attack, all possible values of the secret key are tried out, which is computationally infeasible for the key sizes of modern cryptographic algorithms (128 bits and more). Interestingly, there exists a symmetric-key encryption algorithm called the *one-time pad* which cannot even be broken by a brute-force attack because for a given ciphertext all possible plaintexts are equally likely. The major drawback of a one-time pad is that it requires a truly-random key bit for each plaintext bit and the key bits must never be reused.

2.1.3 Implementation Attacks

A relatively new type of cryptanalytic attacks are implementation attacks. As the name implies, not the abstract descriptions of cryptographic algorithms are attacked in this case but their practical implementations in *cryptographic devices*. Implementation attacks became popular with the rise of mechanical and especially electrical devices (integrated circuits) that perform the cryptographic operations. A cryptographic device consists of the components in a system that either perform cryptographic operations or that handle the data of these operations. Examples include general-purpose hardware like microcontrollers that execute software implementations of cryptographic algorithms and protocols, dedicated cryptographic hardware modules like encryption coprocessors, and (non-volatile) memories that store secret keys. Cryptographic devices come in many different shapes like smart cards or USB tokens. They generally tend to be small, closed platforms which facilitates protection from attacks.

Implementation attacks are very powerful because they also utilize the information "emitted" by the physical implementation, for example the execution

time of a cryptographic algorithm. Furthermore, they can include a wide range of manipulations of implementations like cutting a wire in a circuit on a chip. Classical cryptanalytic attacks are much more restricted in this respect. On the other hand, a practical mathematical weakness in a cryptographic algorithm renders it completely unusable while a successful implementation attack typically only affects a specific cryptographic device. In most cases, no general weakness of the cryptographic algorithm can be derived from such an attack.

Implementation attacks can be generally classified in two ways. First, we distinguish between *passive* and *active* attacks. In passive attacks, only information emitted by a normally operating cryptographic device (e.g. a physical property like its power consumption) is observed. This includes that the device is operated within its specifications (supply voltage, temperature, etc.). In active attacks, the cryptographic device and/or its environment are manipulated. The main aim is to make the device behave abnormally and to exploit this abnormal behavior in an attack.

A second common classification criterion for implementation attacks is the interface that is used to manipulate a cryptographic device or to access the available information. In this context, we distinguish *non-invasive*, *semi-invasive*, and *invasive* attacks. In non-invasive attacks, the cryptographic devices are not permanently altered in any way. Only information from directly accessible interfaces is exploited. In semi-invasive attacks [Sko05], only outer, basically non-functional layers of cryptographic devices are removed (e.g. chip packages). Active circuitry is neither directly contacted nor altered. Finally, invasive attacks include all manipulations and means of access that are technically feasible. This includes directly probing, cutting, and rerouting of wires and changing the logic states of signals in a chip (forcing). It is clear that the different types of implementation attacks differ significantly in terms of their complexity and cost. A general rule of thumb is that passive attacks are easier to conduct than active ones and that invasive attacks are much more complex and costly than semi-invasive and non-invasive attacks.

The main types of implementation attacks are: *side-channel analysis*, *fault analysis*, *probing attacks*, and combinations of them. Also *reverse engineering* of cryptographic algorithms is commonly considered as a type of implementation attack. The common taxonomy to classify attacks on cryptography is summarized graphically in Figure 2.1. Reverse engineering in this context means that the software or hardware implementation of an unknown cryptographic algorithm is analyzed to find out how it actually works. Cryptanalytic attacks are typically only possible if the functionality of the targeted cryptographic algorithms is known. This is especially true for implementation attacks. However, security products often use proprietary cryptographic algorithms whose details are kept secret. There is a steady discussion going on if this violation of Kerckhoffs' principle actually increases the overall strength of a cryptographic system or not. Opponents of this *security-by-obscurity* approach argue that designing mathematically strong cryptographic algorithms is so complex that only an open evaluation by many different experts leads to robust results. Furthermore they

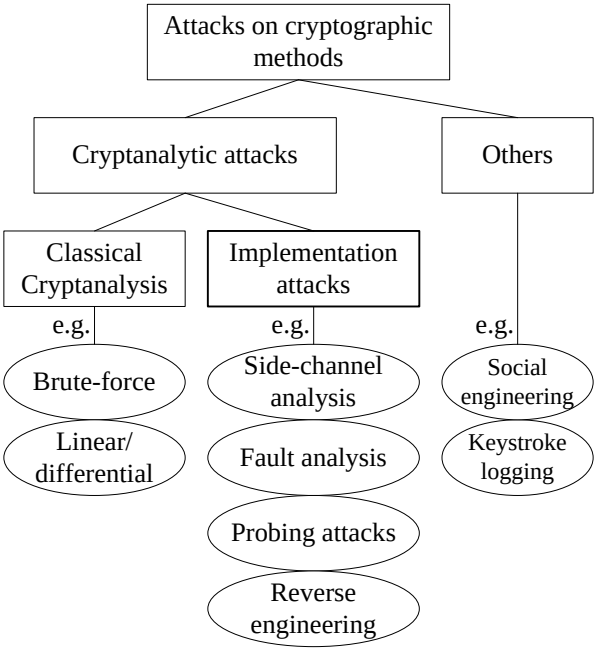


Figure 2.1: The common taxonomy to classify attacks on cryptographic methods.

say that keeping the details of cryptographic algorithms secret in the long run is very hard. A recent example is the successful reverse engineering of the proprietary *Crypto-1* encryption algorithm that is used in some radio-frequency identification (RFID) tags [NESP08]. After the semiconductor and wiring layers of such an RFID tag had been visually analyzed, the digital circuit could be reconstructed and the details of the algorithm were revealed. Subsequent cryptanalytic attacks on the algorithm showed that it is mathematically weak [GvRVS09].

Side-channel analysis (SCA) attacks have received a lot of attention in the last years. They are passive, non-invasive attacks which exploit physical signals (the "side channels") emitted by normally operated cryptographic devices. The most common side channels are execution time, power consumption, and electromagnetic (EM) emanation. The basic principle of SCA attacks is to determine secret keys from their influence on the side-channel signals. How this works and what can be done against it is presented in more detail in Sections 2.2 and 2.4.

Fault analysis (FA) attacks have become increasingly popular in the last years. FA attacks are in general more complex than SCA attacks because they are active attacks. Their aim is to induce faults in a cryptographic circuit so that it behaves abnormally and/or delivers incorrect results, which in turn reveals information about the secret key. FA attacks and possible countermeasures are for example discussed in more detail in [Sch09].

Probing attacks are another, very powerful type of implementation attacks.

These are passive, invasive attacks where outputs of memory cells, wires between logic cells, etc. are electrically contacted to read out their state while the cryptographic circuit operates normally. Very expensive equipment like a chip probing station is necessary for such attacks. Furthermore, probing attacks get more and more complicated because of the steadily shrinking feature sizes of semiconductor technologies into the nanometer range and because of the exponentially increasing complexity of chip circuits. This generally implies more routing layers and thus leads to wires that are no longer accessible without destroying other important circuit structures. Nevertheless, the effort may still be worth it because successfully planted probes allow in the best case to directly read out bits of secret keys.

2.2 Power Analysis Attacks

Besides the obvious output channel of cryptographic devices, namely the one providing the result of the executed cryptographic algorithm, there exist various other, so-called *side channels* which also emit information during operation. Such information is for example how long the algorithm execution takes (*timing analysis, TA*), how much current is drawn from the power supply (*power analysis, PA*), or how much EM radiation is emitted due to electrical activity (*electromagnetic analysis, EMA*). Also some more exotic side-channel signals like sound or temperature have already been investigated.

If a secret key is included in a cryptographic algorithm, the output values of all side channels are somehow related to that key. In case of the result output channel, the influence of the secret key is present and observable only in the final output value. The major difference to side-channel signals is that they are typically also available for the calculation of the *intermediate results* of a cryptographic algorithm. Many intermediate results only depend on small parts of the secret key. This enables a divide-and-conquer approach, which dramatically reduces the attack complexity.

PA and EMA attacks are the most common and well studied types of SCA attacks. If data is processed in digital circuits, signal wires carry different data values and must be charged and discharged. This leads to data and operation-dependent current flows and EM fields. The sources of the power consumption of cryptographic circuits are explained in more detail in Section 2.3.

SCA attacks exploiting the power consumption side-channel were presented for the first time at a conference in 1999 [KJJ99]. One of the first academic publications about EMA attacks was [GMO01]. However, we know today that the US government was already aware in the 1950s that for example EM signals radiated from monitor screens can be used to reconstruct the displayed content from a distance. This led to the development of a set of standards summarized with the term *TEMPEST*, which describes possible countermeasures.

The power consumption and the EM radiation of electronic devices have the same source: The currents flowing in the circuits. Thus, the results from both types of SCA attacks are similar to some extent. A main difference is the

necessary measurement equipment: resistors and current probes inserted in the power supply lines in case of PA; EM probes placed on the chip package in case of EMA. Furthermore, EMA attacks allow targeting only parts of a circuit if a sufficiently small probe can be placed near the interesting circuit component. This might also help to defeat some countermeasures. One the other hand, the noise in EMA measurements is typically higher than in PA measurements (strong carrier signals, etc.).

The basic equipment necessary for PA and EMA attacks is relatively inexpensive. A state-of-the-art personal computer usually provides enough computing power, versatility, and memory to control the side-channel measurements, store the measurement results, and do the analysis. For the measurement of the side-channel signals, instruments like oscilloscopes, EM antennas, and spectrum analyzers are necessary. These are today also available off-the-shelf with sufficient performance and accuracy.

The three basic subtypes of PA and EMA attacks are: *simple power/EM analysis (SPA/SEMA)*, *differential power/EM analysis (DPA/DEMA)*, and *template* PA/EMA attacks. Especially DPA/DEMA attacks are very common. We shortly introduce simple, differential, and template PA attacks in the following. The related EMA attacks basically work in the same way.

2.2.1 Simple PA Attacks

SPA attacks are possible in cases where the power consumption of a circuit strongly depends on the individual bits of the secret key. For example, the value of a bit might determine which of two operations is executed next or whether or not an operation is executed at all. If the operations are clearly visible and distinguishable in the power signal, an attacker can directly deduce the key bits simply by looking at the patterns in a power trace. SPA attacks typically require detailed knowledge about the attacked device and are relatively easy to prevent. The main characteristics of SPA attacks are that only one or few power traces are used and that a trace is analyzed along the time axis.

2.2.2 Differential PA Attacks

Much harder to prevent are DPA attacks. In these attacks, the power information emitted during the calculation of a specific intermediate result is statistically analyzed. The targeted intermediate result must depend on a small part (e.g. 8 bits) of the secret key and on other, varying but known data (e.g. parts of the plaintext or ciphertext). The concrete value of the key part in question specifically influences the emitted power signal when calculating the intermediate result. An attacker tries to estimate the power signal for the different possibilities of the key part with the help of a *power model* and determines which estimation fits best to the actual (measured) power traces. Figure 2.2 graphically depicts the principle of DPA attacks. Very small influences of the secret key on the power signal can be exploited by a differential attack. Such small signals are not directly visible because they are buried in the *switching*

noise caused by simultaneously calculated intermediate results and in all sorts of *electronic noise*. Due to the statistical analysis of the power signal for varying, known data, it is possible to significantly reduce this noise in the signal.

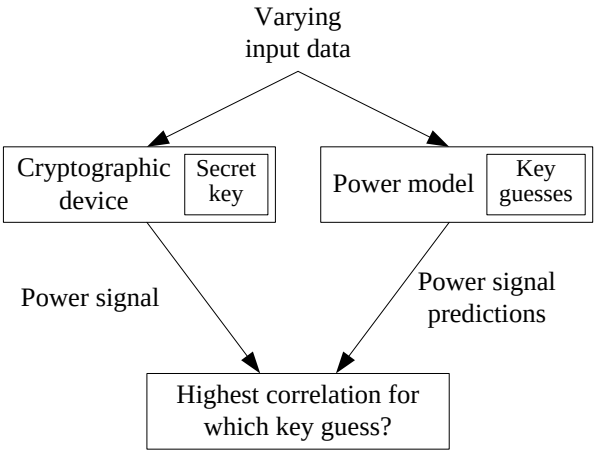


Figure 2.2: The principle of DPA attacks.

DPA attacks are in general performed according to the following five-step strategy:

1. Choose an intermediate result v of the executed algorithm that depends on a small part of the secret key k and on varying but known data d: $v = f(k, d)$.

2. Measure the power consumption of the cryptographic device (i.e. record power traces) for varying data d. This is typically achieved by varying the input data of the algorithm.

3. For the known data values d and all possible guesses of the secret-key part k (i.e. the 256 possible values in case k is 8 bit wide) calculate the hypothetical intermediate values v according to the formula above.

4. Map the hypothetical intermediate values v to hypothetical power consumption values with the help of a suitable power model.

5. Compare the hypothetical power consumption values for each key guess with the measured power values. The key guess for which these values show the highest correlation is under normal circumstances the actual part of the secret key used in the cryptographic device.

Subsequently, the attack is repeated for the next part of the secret key. In most cases, new power measurements (step 2) are not necessary.

DPA attacks are very common because very good generic power models are available for them. We discussed before that such models are required to map

hypothetical intermediate values (depending on the secret key) to hypothetical power consumption values. These generic power models are the Hamming-weight (HW) model, the Hamming-distance (HD) model, and the zero-value (ZV) model. The HW model assumes that the emitted power signal when calculating the intermediate result is correlated to the number of bits that are 1 in the intermediate result, the HD model assumes a correlation to the number of changing bits, and the ZV model assumes that the power signal is significantly different for a special value like 0 compared to all other values. These power models are discussed in more depth in Section 3.1.3.

The main characteristics of DPA attacks are that many power traces for varying processed data are measured and that the analysis happens over the power values of the traces at a fixed point in time. The points in time where the differential attack finally succeeds marks exactly that points when the targeted intermediate result is processed. This means that an attacker does not exactly need to know when this processing actually happens. The power signal is just recorded for a reasonable interval (e.g. from the beginning of an encryption to its end) and the DPA attack is then performed successively at each point in time. This is one main reason why DPA attacks typically require less detailed knowledge about the attacked device than SPA attacks. Obviously, longer traces imply increased attack runtimes (trace transmission, analysis for each point in time) and increased memory capacity for trace storage.

A very common way to quantify the relationship between the hypothetical power consumption values and the measured power consumption values is to calculate (estimate) the linear Pearson correlation coefficient ρ [CKN01, BCO04] between both sets of values. In most cases, the assumption holds that for wrong points in time (where the attacked intermediate result is not processed) and for wrong key guesses this correlations approaches zero with an increasing number of measurements. For the correct time ct and the correct key ck, a correlation peak $\rho_{ck,ct}$ (actually its estimation $r_{ck,ct}$) appears. The height of $\rho_{ck,ct}$ depends on the signal-to-noise ratio (SNR) between the exploitable part of the power consumption in the given attack scenario and the noise (switching noise and electronic noise). In the context of PA attacks, the exploitable part of the power consumption is also often called the *power leakage* of a specific attack scenario. Furthermore, $\rho_{ck,ct}$ depends on the quality of the power model used by the attacker, i.e. how well the hypothetical power values actually describe the exploitable power consumption of a cryptographic device.

From a designer's perspective, we usually use the *minimum number n of needed power traces to get a distinct correlation peak* to quantify the DPA resistance of a cryptographic circuit for a specific attack scenario. Based on the assumption that there only occurs a correlation value different from zero for the correct key and the correct point in time, we consider a correlation peak estimation as distinct if we can distinguish it in a statistical sense from a correlation "peak" of value zero with high probability [Man04a]. For a confidence of 99.99 % and for a correlation peak $\rho_{ck,ct} \leq 0.2$, the minimum number of traces n can be approximated according to Equation 2.1 (ck ... correct key, ct ... point in time

where the attacked intermediate result is processed). $\rho_{ck,ct}$ and n actually show a quadratic relationship. Another, more general approach to quantify the DPA resistance of cryptographic circuits based on success rates has been proposed in [SMY09].

$$n = \frac{28}{\rho^2_{ck,ct}} \tag{2.1}$$

So far, we only discussed first-order DPA attacks where only one intermediate result is analyzed. In *higher-order* DPA attacks, two (second-order attacks) or more intermediate results are considered simultaneously. In other words, higher-order DPA attacks exploit the combined leakage of two or more intermediate results that are processed by the cryptographic device [Mes00]. In a higher-order DPA attack, usually a preprocessing step based on a suitable combination function is necessary to bring the power consumption contributions of the different intermediate results together. Furthermore, the different intermediate results are included in the calculation of the hypothetical intermediate values. In some cases, the preprocessing step is not necessary. Examples are situations where the intermediate results in question are processed *jointly* (i.e. as inputs to the same function), in *parallel* (e.g. in different hardware modules at the same time), or *consecutively* (e.g. stored in the same register one after the other). Higher-order DPA attacks are mainly used to defeat DPA countermeasures like masking (see Section 2.4). Usual targets are in this case intermediate results that are masked with the same mask or a masked value and the corresponding mask.

An example of successfully applying SPA and first-order DPA attacks on a commercial product is the attack on the KEELOQ code hopping protocol, which is widely used in remote keyless entry systems of garage and car doors. The DPA attacks [EKM+08] allowed to retrieve the device keys of the hand-held transmitters, which can then be cloned easily. Much more seriously, also the manufacturer key could be disclosed by the DPA and SPA attacks [KKMP09]. The manufacturer key is unique for a particular vendor of KEELOQ products. Knowing it allows for example to calculate device keys of transmitters and to produce your own transmitters.

2.2.3 Template PA Attacks

Another type of attacks are template PA attacks. In such attacks, the power model that describes how the performed operations and the processed data values influence the power consumption are determined in a *characterization step*. This requires that the attacker has access to a fully controllable cryptographic device, which is similar or even identical to the attacked one. With this characterization device, the attacker builds descriptions of the power signal for different keys and processed data. One such description for a specific key and processed data value is called a template. A template usually describes several points of a power trace that show a relation to the used key and processed data. The templates then help to enhance the subsequent SPA [CRR03] or DPA attacks

[ARR03]. Template-based SPA and DPA attacks are very powerful types of PA attacks but they require the availability of a characterization device.

2.3 CMOS Circuits and Their Power Consumption

Most digital circuits are today implemented in a complementary metal-oxide semiconductor (CMOS) *logic style*. This basically means that *combinational* logic cells usually consist of a pull-up network (PUN) and a pull-down network (PDN), which are controlled by the input signals and are never conducting at the same time. A conducting PUN sets the affected output signals to V_{DD} while a conducting PDN sets it to *GND*. Besides the combinational logic cells, which react directly on input value changes according to their logic function (e.g. AND, XOR, ADD), *sequential* cells like flip-flops (FFs) are important building blocks of digital circuits. Sequential cells work synchronously to a control signal that is usually called the *clock signal* and they are mainly concerned with the storage of signal values. CMOS logic styles are implemented in a CMOS *process technology* from a particular semiconductor foundry. A CMOS process technology allows to efficiently implement the PUNs and PDNs of the logic cells with two complementary types of metal-oxide semiconductor (MOS) transistors in the same substrate.

Due to the widespread use of CMOS logic styles, also cryptographic devices are mainly implemented in CMOS. In the following, we discuss when and how such logic styles consume power because this ultimately is the source of power leakage that can be exploited by attackers.

2.3.1 Total Power Consumption

The total power consumption of a CMOS circuit is the sum of the power consumptions of the logic cells making up the circuit. It essentially depends on the number of logic cells in a circuit, the connections between them, and the fact how the cells are built. These properties are a result of design decisions that are taken at the system level (overall system architecture, used algorithms, hardware/software splitting, etc.), the architecture level (specific implementation of hardware and software components), the cell level (design of the logic cells), and the transistor level (process technology used to implement the MOS transistors of the logic cells).

When operating a CMOS circuit, the circuit is provided with the constant supply voltage V_{DD} and with input signals. The logic cells in the circuit process the input signals and draw current from the power supply. We denote the total instantaneous current of the circuit by $i_{DD}(t)$ and the instantaneous power consumption by $p_{DD}(t)$. Hence, the average power consumption P_{DD} of the circuit over time T can be calculated according to Equation 2.2.

$$P_{DD} = \frac{1}{T}\int_0^T p_{DD}(t)dt = \frac{V_{DD}}{T}\int_0^T i_{DD}(t)dt \tag{2.2}$$

As pointed out before, logic cells are usually implemented using a CMOS logic style. In the following, we use the simplest CMOS cell, the CMOS inverter, to describe when and why CMOS cells dissipate power. The discussion of the inverter is representative for all other cells because all CMOS cells are built based on complementary pull-up and pull-down networks. In case of an inverter, these networks consist of the two transistors $P1$ and $N1$, respectively, as shown in Figure 2.3. In case of more complex gates, more MOS transistors are necessary for these networks.

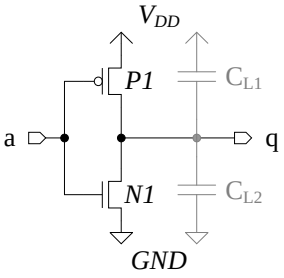


Figure 2.3: Transistor schematic of a CMOS inverter with output capacitances.

The power consumption of a CMOS inverter can essentially be divided into two parts. The first part is the static power consumption P_{stat}. This is the power that is consumed if there is no switching activity in a cell. The second part of the power consumption is the dynamic power consumption P_{dyn}. In addition to the static power, a cell consumes dynamic power if an internal signal or an output signal of a cell switches. The total power consumption of a cell is the sum of P_{stat} and P_{dyn}.

2.3.2 Static Power Consumption

CMOS cells are built in such a way that their PUN and their PDN are never conducting at the same time for constant input signals. In case of the CMOS inverter, $P1$ is conducting and $N1$ is insulating if the input a is set to *GND*. Vice versa, $P1$ is insulating and $N1$ is conducting if the input a is set to V_{DD}. In both cases, there is no direct connection between the V_{DD} line and the *GND* line. Therefore, only a small leakage current is flowing through the MOS transistor that is turned off. Furthermore, this leakage current typically shows a very low data-dependence. If we denote the average leakage current by I_{leak}, the static power consumption P_{stat} can be calculated according to Equation 2.3. The leakage currents of MOS transistors are typically in the range of $10^{-12}\,A$ [RCN03]. However, the leakage currents are increasing significantly for modern process technologies that have structure sizes of less than 100 nanometers. Thus,

also their data dependence becomes more and more important in the context of PA attacks.

$$P_{stat} = I_{leak} \cdot V_{DD} \tag{2.3}$$

2.3.3 Dynamic Power Consumption

Dynamic power consumption occurs if an internal signal or an output signal of a logic cell switches. At a fixed moment of time, an internal or output signal of a CMOS cell can essentially perform one out of four "transitions". In the two cases $0 \rightarrow 0$ and $1 \rightarrow 1$, the cell consumes only static power, while in the other two cases $0 \rightarrow 1$ and $1 \rightarrow 0$ also dynamic power is consumed. The corresponding power consumption values P_{00}, P_{11}, P_{01}, and P_{10} depend on the type, load, and input signal characteristic of the cell and on the used process technology. However, in general it holds that $P_{00} \approx P_{11} \ll P_{01}, P_{10}$. The most important aspect in the context of PA attacks is that the dynamic power consumption is data-dependent.

There are essentially two reasons for the dynamic power consumption P_{dyn} of a CMOS cell. The first one is that the involved load capacitances of the cell need to be charged. The second one is that there usually occurs a short circuit for a short period of time if a signal of a cell is switched. P_{dyn} is the dominant part of the total power consumption for process technologies with structure sizes of $100\,nm$ and more. For smaller process technologies, the static part of the power consumption becomes steadily more significant.

Charging Current

When performing internal and output signal transitions, CMOS cells draw a charging current from the power supply to charge the involved load capacitances. The load capacitances consist of the so-called intrinsic capacitances and the extrinsic capacitances of a CMOS cell. Intrinsic capacitances are internal capacitances that are involved in the switching activity. Extrinsic capacitances include the capacitances of the wires that connect to the subsequent CMOS cells and the input capacitances of these cells. The sizes of the capacitances depend heavily on the properties of the used process technology and the cell and circuit layouts (internal structure and dimension of cells, geometry and environment of output wires, number of subsequent cells, etc.).

To describe the charging power consumption of CMOS cells we use a model that includes two output capacitances C_{L1} and C_{L2} as shown in Figure 2.3. C_{L1} includes all intrinsic and extrinsic capacitances that need to be charged from the power supply when the output signal changes from V_{DD} to *GND*. C_{L2} contains all capacitances that are involved in the opposite transition. In case of the CMOS inverter, C_{L1} is charged from the power supply and C_{L2} is discharged internally via the MOS transistor $N1$ if the output signal q changes from V_{DD} to *GND*. If q changes from *GND* to V_{DD}, C_{L2} is charged from the power supply and C_{L1} is discharged internally via the MOS transistor $P1$. In the general case, C_{L1} and

C_{L2} are not equal due to different effective capacitances to V_{DD} and *GND*. As a consequence, the power consumption values P_{01} and P_{10} are not equal. This fact represents a common root cause of vulnerabilities to PA attacks.

The average charging power P_{chrg} that is consumed by a cell during the time T can be calculated as shown in Equation 2.4. In this equation, $p_{chrg}(t)$ denotes the instantaneous charging power consumed by the cell. If we set T to be the clock cycle time, P_{chrg} can be calculated using the clock frequency f, the so-called activity factor α, the load capacitances, and the supply voltage. The activity factor corresponds in this case to the average number of $0 \to 1 \to 0$ transitions that occur at the output of a cell in each clock cycle. For example, if a cell switches its output value from 0 to 1 and back to 0 once every clock cycle, α is 1.

In Equation 2.4, it can be observed that P_{chrg} is proportional to $\alpha \cdot f$, the sum of the load capacitances C_{L1} and C_{L2}, and the square of the supply voltage V_{DD}. Thus, the most effective way to reduce the dynamic power consumption of CMOS circuits is to reduce the supply voltage. Less effective options are to reduce the clock frequency, the load capacitances, or the switching activity of the cells.

$$P_{chrg} = \frac{1}{T} \int_0^T p_{chrg}(t)dt = \alpha \cdot f \cdot (C_{L1} + C_{L2}) \cdot V_{DD}^2 \tag{2.4}$$

Short-Circuit Current

The second part of the dynamic power consumption of a CMOS cell is caused by the temporary short circuit between the PUN and the PDN that occurs in a CMOS cell during the switching of the output. In case of a switching event of a CMOS inverter, there is a short period of time where both MOS transistors $P1$ and $N1$ are conducting simultaneously. This happens for $0 \to 1$ and $1 \to 0$ transitions.

The average power consumption P_{sc} that is caused by the short-circuit currents in a cell during the time T can be calculated as shown in Equation 2.5. In this equation, $p_{sc}(t)$ is the instantaneous short-circuit power consumed by a cell. If T corresponds to the clock cycle time, I_{peak} denotes the current peak that is caused by the short circuit during the switching event (neglecting a dependency on the transition direction), and t_{sc} is the time for which the short circuit exists, Equation 2.5 can be rewritten as shown. For the calculation of P_{sc}, the waveform of the short-circuit current for one switching event is approximated by a triangle with base length t_{sc} and height I_{peak}. Note that two switching events with a temporary short circuit occur in a complete switching cycle ($0 \to 1 \to 0$) of the inverter output q. The values of t_{sc} and I_{peak} mainly depend on how fast input and output signals switch (absolutely and in relation to each other).

$$P_{sc} = \frac{1}{T} \int_0^T p_{sc}(t)dt = \alpha \cdot f \cdot V_{DD} \cdot I_{peak} \cdot t_{sc} \tag{2.5}$$

2.3.4 Glitches and Early Propagation

An important observation for CMOS circuits is that the different *input signals of multi-input cells usually arrive at different, data-dependent points in time.* There are basically two reasons for this. First, the logic cells and the wires between these cells have non-zero, data-dependent, and in general different propagation delays. It takes a certain time until a cell switches its output upon a specific change of the input, and it also takes a certain time for a specific signal transition to propagate from one cell to the next. Second, multi-input cells often take outputs from cells in different stages of the circuit as input.

Another characteristic behavior of multi-input combinational CMOS cells is that they show what is commonly called an *early-propagation effect.* This means that a combinational cell changes its output as soon as one or some new input values define a new output value. In other words, the evaluation of the cells is not postponed until e.g. all input values have reached their final state of the current clock cycle. It is clear that in the case of binary value encoding and combinational CMOS cells, early propagation is in general not avoidable because a differentiation between valid and invalid signal states is not possible.

The different, data-dependent arrival times of the input signals of combinational cells and the early propagation effect are both *necessary* prerequisites for an important phenomenon to occur in digital circuits. As observable in standard CMOS circuits, where these effects occur, combinational CMOS cells usually switch their output signals several times before they reach the final value of the current clock cycle. Such switchings of combinational cells to temporary output states are called *glitches* or *dynamic hazards* [RCN03].

The number of glitches in CMOS circuits typically increases with the number of stages of the combinational circuit. Glitches propagate like an avalanche through such circuits. If there is a glitch in one of the early stages of the combinational circuit, this single glitch most likely leads to glitches at the outputs of all cells that are connected to this cell etc. In complex CMOS circuits, glitches can become a dominant factor in the total power consumption. Most notably in the context of PA attacks is that the occurrence of glitches is data-dependent.

2.4 Power Analysis Countermeasures

Many countermeasures against SCA attacks in general and PA attacks in particular have been proposed in the last ten years. In case of PA, the aim of any countermeasure is to reduce the exploitable information in the power consumption to a level where attacks are no longer possible or at least to a level that would require too many measurements to be taken by an attacker. The first goal is of course the more attractive one but it is very hard to reach in practice. As a compromise, designers of secure cryptographic systems usually step back from the ideal goal of achieving *unconditional security* in terms of PA attacks and apply the following more practical rule: **The costs for an attacker in terms of required knowledge, equipment, time, and memory must outweigh**

the (expected) profit. Countermeasures are selected and parameterized accordingly in order to fulfill this goal. As a consequence, it is necessary to clearly know the application domain of a cryptographic device already in the design phase.

A general countermeasure especially against differential PA attacks is to use secret keys and similar data as rarely as possible (e.g. derive session keys and work with them) and to update the keys as often as practically possible. The main problems of this approach are the need for regular key updates even in the field and the protection of the key derivation operation itself against PA attacks.

More technically speaking, the goal of PA countermeasures is to break the dependency between the data values and operations occurring in a cryptographic algorithm and the power signal of a cryptographic device that executes this algorithm. Two basic approaches can be followed to achieve this goal. The first one is *hiding*, which aims at breaking the link between the data values processed in the device and the power consumption. The second approach is *masking*, which means that the intermediate results in the algorithm are randomized before they are processed in the device. Hiding as well as masking countermeasures are applied either at the *architecture level* (software/hardware) or at the *cell level* (hardware) of cryptographic devices.

Hiding countermeasures try to change the power consumption characteristics of cryptographic devices in a way that the data and operations processed on a device and the emitted power signals are no longer related. However, this is typically not perfectly possible in practice. There are essentially two approaches to implement hiding. The first one is to *randomize* the power consumption. The second approach is to make the power consumption *equal* for all operations and data values.

Randomization of the power consumption can be achieved by performing the operations of cryptographic algorithms at different moments in time for each execution (randomization in time). As a result, the operations and their corresponding signals in the power traces are no longer aligned, which makes PA attacks much harder [Man04a]. The second randomization technique is to directly increase the noise level in the power signal (randomization in the amplitude dimension). Examples for randomization in time are: random insertion of dummy operations [CCD00], dummy clock cycles, or delays [BLGT05], shuffling of operations [THM07], and randomly changing the clock frequency [YWV+05]. Examples for randomization in the amplitude dimension are: architectures with wider datapaths and noise engines [PGH+04].

A more equal power consumption for different operations and data values can be achieved by carefully selecting executed instructions, program flow (e.g. no conditional jumps depending on the secret key value), and memory accesses. A rather equal power consumption is possible for example by filtering the power signal [Sha00, Plo09] and with special logic styles (see Section 2.4.1). Equalizing the power signal often has the effect that the cryptographic device consumes the maximum amount of power in each clock cycle.

When masking is used as countermeasure against PA attacks [GP99, CJRR99],

each vulnerable intermediate result v of a cryptographic algorithm is concealed by a random value m called the *mask*: $v_m = v * m$. The random mask is generated internally by a device because it must not be known by an attacker. All operations performed in a cryptographic device work only with the masked values v_m. Due to the random mask, these values are independent of the corresponding unmasked values. Therefore, also the power consumption when processing the masked values is independent. The most common masking operations $*$ are the exclusive-or (XOR) function (*Boolean masking)* and arithmetic operations like modular addition or multiplication (*arithmetic masking*). Examples of masking countermeasures are masked look-up tables in software [HOM06], masked functional blocks in hardware [OMPR05, POM+04], masked buses [BGM+03, ETS+05], randomly precharged buses [BGLT04], and masked logic styles (see Section 2.4.4). In the asymmetric-key area, masking is usually called *blinding*.

A significant burden of masking countermeasures and randomization countermeasures in the time and amplitude dimension is their need for random numbers. The pseudo-random number generators (PRNGs) that typically produce these numbers must usually be included in the cryptographic chip so that an attacker cannot access or even alter the random bit stream. Hiding and masked logic styles will be presented in more detailed in the following two sections. Their basic idea is to build cryptographic circuits out of cells that are resistant to PA attacks, i.e. their power consumption is independent of the intermediate results of the executed cryptographic algorithms. If it is furthermore ensured that the cells are not (de)activated in a data-dependent manner (e.g. by using data-dependent clock gating) the circuit will be resistant to PA attacks. In general, PA-resistant logic styles can provide a very high level of protection against attacks and they can usually be applied automatically within well-established chip design flows (see Section 5.2). However, PA-resistant logic styles must be carefully tested in real silicon and they are typically very demanding in terms of implementation constraints and resource requirements.

2.4.1 Hiding at the Cell Level

Counteracting PA attacks at the cell level has been one of the first reactions of the semiconductor industry after the publication of these attacks. In the scientific community, it took much longer until the first proposals for cell-level countermeasures appeared. During the last years, several proposals for logic styles to counteract power analysis attacks have been made. Many of these logic styles are based on the concept of hiding and aim at making the power consumption of the cells constant in each clock cycle for all processed logic values. Constant in each clock cycle means that the instantaneous power consumption of a cell is the same in each clock cycle. A consequence of this behavior is that the logic cells always consume the maximum amount of power in each clock cycle.

Logic styles with a constant power consumption counteract both SPA and DPA attacks. In standard CMOS circuits, essentially four transitions can occur at a node that stores the data value $v(t-1)$ at time $t-1$ and the value $v(t)$

at time t. Table 2.1 lists the possible transitions together with the energy that is dissipated in order to perform the respective transition ($E_{00} \dots E_{11}$). Each of these transitions occurs with a certain probability denoted by $p_{00} \dots p_{11}$.

Table 2.1: Possible transitions of the value v on a node in a CMOS circuit and corresponding energy consumptions.

$v(t-1)$	$v(t)$	Energy	Probability
0	0	E_{00}	p_{00}
0	1	E_{01}	p_{01}
1	0	E_{10}	p_{10}
1	1	E_{11}	p_{11}

In a DPA attack, the recorded power traces of the CMOS circuit are split into two sets e.g. according to the value $v(t)$ at time t. Subsequently, the attacker determines the difference of the means (DM) of the two sets of traces. We refer to the means as $M_{v(t)=0}$ and $M_{v(t)=1}$. Of course, at the time t not only the attacked node performs a transition. Several other nodes also switch their value at this moment in time. However, the energy dissipation that is caused by these other nodes can be modeled as Gaussian noise [MDS02]. Therefore, the expected value of the difference of the means, $\mathcal{E}(DM_{v(t)})$, is calculated as shown in Equation 2.6 for a correctly calculated $v(t)$ (i.e. when the attacker uses the right key guess).

$$\mathcal{E}(DM_{v(t)}) = \mathcal{E}(M_{v(t)=1}) - \mathcal{E}(M_{v(t)=0}) = \frac{p_{11}E_{11} + p_{01}E_{01}}{p_{11} + p_{01}} - \frac{p_{00}E_{00} + p_{10}E_{10}}{p_{00} + p_{10}} \tag{2.6}$$

In case of standard CMOS logic it usually holds true that $E_{00} \approx E_{11} \ll E_{10} \neq E_{01}$ (see Section 2.3.1). In this case, $\mathcal{E}(DM_{v(t)})$ is different from zero, which leads to a successful DPA attack. A common approach to prevent the expected value of the difference of means from being non-zero is thus to use cells with the property that $E_{00} = E_{01} = E_{10} = E_{11}$, i.e. to use cells with a constant power consumption.

The constant power consumption of hiding logic styles is typically achieved by applying the *dual-rail precharge (DRP)* principle and by balancing complementary paths. In the following, we introduce this important working mechanism and we present some examples of common hiding logic styles that are based on the DRP principle. Subsequently, we introduce some alternative approaches to achieve cells and circuits with a constant power consumption.

DRP Logic

A DRP circuit alternates between two operating states called the *evaluation phase* and the *precharge phase*. The sequence of the phases is usually controlled by the clock signal of the circuit, e.g. by the actual value of the clock signal. In DRP circuits, each signal is carried on two wires. During the evaluation phase, a

binary signal value is encoded in a differential manner on the wire pair: the two wires are set to complementary values (thus, the two wires are also commonly called *complementary wires*); which one of the two wires is 1 defines whether a logical 0 or a logical 1 is carried over the wire pair. During the subsequent precharge phase, both wires of a pair are set to the specified precharge value, which is either 0 or 1. This means, wire pairs do not carry complementary signals during the precharge phase.

If we assume that glitches do not occur in a DRP circuit, a DRP cell in this circuit shows the following behavior during a switching sequence of precharge - evaluation - precharge phase. The wire-pair values of the DRP-cell output switch from the precharge value to complementary values and back to the precharge value. Always two transitions occur on one output wire during the sequence, the signal of the other output wire stays at the precharge value. Thus, the DRP cell consumes a constant amount of power if the charging and discharging behavior of both wires of a pair is the same, i.e. if both wires are balanced in an electrical sense. This balancing is usually achieved by ensuring that for each wire pair, the output capacitances of the driving DRP cell, the wire capacitances, and the input capacitances of the subsequent DRP cells are the same. The corresponding resistances and inductances must also be balanced in the same way. Finally, it also has to be ensured that the internal power consumption of the DRP cell is constant with respect to different input and output values. In summary, DRP cells and circuits must fulfill three requirements to have a constant power consumption: no glitches; pairwise balanced complementary output paths of DRP cells; and constant internal power consumption of DRP cells.

Well-studied PA-resistant logic styles based on the DRP concept are Sense-Amplifier Based Logic (SABL) and Wave Dynamic Differential Logic (WDDL). They are presented in more detail in Section 2.4.2 and Section 2.4.3, respectively. There exist various other proposals for PA-resistant DRP logic styles. Bystrov et al. [BSYK03] and Sokolov et al. [SMBY04, SMBY05] presented the so-called *Dual-Spacer Dual-Rail (DSDR)* logic style. In this DRP logic style, both possible precharge values are used in an alternating way in subsequent precharge phases. In doing so, it is ensured that at both complementary outputs of a cell the same signal transition occurs in each switching sequence of precharge - evaluation - precharge phase. As a result, the overall energy consumption of a logic cell during a complete switching sequence is not influenced by any imbalance of the complementary outputs. However, especially for low clock rates, also the variations of the energy consumed in each phase of the switching sequence can be observed in power measurements.

A DRP logic style that works similarly to DSDR logic was proposed by Bucci et al. [BGLT06]. The logic style is called *Three-Phase Dual-Rail Precharge Logic (TDPL)*. During a clock cycle, TDPL cells run through three operation phases. In the first phase, the complementary outputs of a TDPL cell are charged to (1, 1). Next, in the evaluation phase, one output is discharged to 0 according to the input values and the function of the TDPL cell. In the third phase, also the second output is discharged to 0. As discussed for DSDR logic, this leads to a

constant energy consumption in each clock cycle independent of the balancing of the complementary outputs. A single-rail version of TDPL called *Three-Phase Single-Rail Precharge Logic (TSPL)* has been published by Menendez and Mai [MM08]. TSPL should require less area and power than TDPL and does not have the requirement of balanced complementary wires.

Another DRP logic style is *3-State Dynamic Logic (3sDL)* presented by Trifiletti et al. [AMM+05]. This logic style uses $V_{DD}/2$ as precharge value. At the end of the evaluation phase, always one complementary output wire of a cell has been charged to V_{DD} while the other output wire has been discharged to *GND*. In the subsequent precharge phase, the two complementary wires are connected together. If both wires have the same capacitance, the voltage level at both wires settles to $V_{DD}/2$. This approach saves power. Another specialty of 3sDL is that the second wire of each pair is not routed through the circuit. Instead, a dummy capacitor is connected to the respective cell output whose capacitance matches the capacitance of the first wire. A major drawback of this approach is that this matching must be done individually for every wire pair in the circuit.

So far, almost all articles on DRP logic styles have been published by people from academia. One of the very few scientific articles explicitly stating that also industry uses this technique is [FS03].

Asynchronous Logic

Asynchronous (or self-timed) circuits have also been proposed as a cell-level countermeasure against PA attacks. Using such circuits for this purpose has for instance been discussed by Moore et al. [MAC+02] and by Yu et al. [YFP03]. Asynchronous circuits that counteract PA attacks are usually implemented as DRP circuits. Therefore, the power consumption of these asynchronous circuits can be balanced in the same way as discussed above for synchronous DRP circuits. Further considerations for balancing the power consumption in asynchronous circuits and a proposal of PA-resistant asynchronous cells have been made by Kulikowski et al. [KSS+05]. Unfortunately, the PA resistance of asynchronous circuits still relies on the balancing of complementary wires. Additionally, asynchronous circuits are hard to verify and there is still a lack of mature electronic design automation (EDA) tools that support the design of such circuits.

The security evaluation of an asynchronous test chip implementing a 16-bit microcontroller was presented in [FML+03], which showed some weaknesses of the approach. An asynchronous implementation of AES resistant to power and timing attacks was shown by Yu and Brée [YB04]. Another asynchronous AES design that should be resistant to PA attacks was presented by Bouesse et al. [BRG04]. However, in [BRR+04] the authors showed that the asynchronous circuit still leaks due to imbalances in the design. Based on these results, an improved design methodology for the asynchronous AES circuit using hierarchical place and route has been developed subsequently [BRDG05]. An interesting idea called the "path swapping method" to get rid of the balanced-routing constraint of the asynchronous design has been presented afterwards [BSR06].

Gürkaynak et al. [GOK+05] proposed the use of globally-asynchronous locally-synchronous (GALS) designs to increase the DPA security of circuits. A design flow for PA-resistant asynchronous circuits was proposed by Kulikowski et al. [KST06]. Kulikowski et al. also came up with a new cell-level approach called Asynchronous Directional Latch-Based Logic (ADLBL). This logic avoids the need for balanced routing by using a non-standard, directional discharge protocol [KVWT08, KVW+08].

Current-Mode Logic

In current-mode logic (CML) circuits, the output value of a logic cell is defined by currents that are passing through the cell. The sum of these currents is rather constant and more or less independent of the actual output value. This makes CML logic styles interesting for PA-resistant circuits. In [TL05], Toprak and Leblebici proposed the use of *MOS Current-Mode Logic (MCML)* [YY92] as a PA-resistant logic style. The main drawback of MCML is its increased static power consumption. A CML logic style that avoids this drawback is *Dynamic Current-Mode Logic (DyCML)* [AE01]. The use of DyCML in PA-resistant circuits has been proposed by Mace et al. [MSH+04].

Adiabatic Logic

Also adiabatic logic styles [Kae08] like the classical charge recovery logic 2N-2N2P [KDFM95] have been proposed to be used as PA-resistant logic styles. The basic ideas of adiabatic logic are to work with very small, more or less constant currents in a circuit to minimize energy dissipation and to reuse the charge stored in capacitances. A PA-resistant logic style proposal that works according to the adiabatic principle is Secure Adiabatic Logic (SAL) [KME+08, MKSS09].

2.4.2 SABL

Sense Amplifier Based Logic (SABL) is a PA-resistant logic style following the DRP principle and has been introduced by Tiri et al. [TAV02]. SABL cells have a very special structure. That means on the one hand that the cells must be implemented from scratch, which causes a high effort when switching to new or smaller process technologies. On the other hand, it allows having cells with an almost constant internal power consumption. If we assume well balanced complementary wires in the SABL circuit, the PA resistance of the circuit is very high (see [TV03] and Section 6.1.1). In [SA05], Sundström and Alvandpour presented a comparative study of the power consumption variance of static CMOS logic, dynamic CMOS logic, dynamic differential logic, an SABL variant, and DyCML. In this comparison however, SABL performs not as good as usually experienced.

Not only the sequential SABL cells but also the combinational ones are connected to the clock signal to switch their state between precharge and evaluation phase. With the help of the transistor schematic of an SABL NAND cell shown

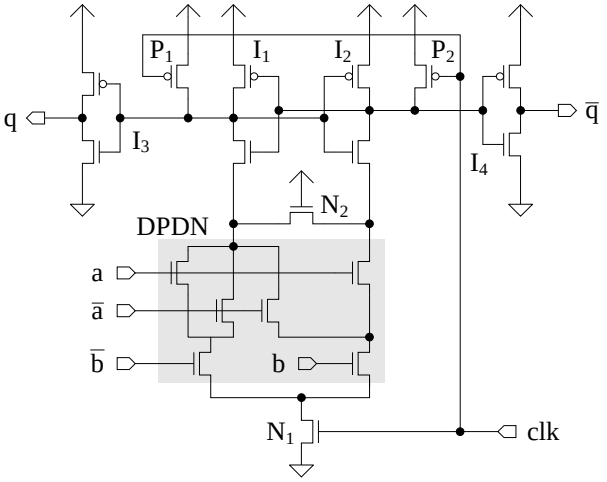


Figure 2.4: Transistor schematic of an SABL NAND cell.

in Figure 2.4, the basic functionality of SABL cells will be discussed (we assume that logic-0 corresponds to *GND* and logic-1 corresponds to V_{DD}). During the precharge phase (clock signal $clk = 0$), the MOS transistors P_1, P_2, and N_1 ensure that all internal nodes of the cell are charged to V_{DD}. Due to the output inverters I_3 and I_4, the outputs q and $\overline{q}$ of the SABL cell are set to the precharge value 0. The same is true for the inputs a, $\overline{a}$, b, and $\overline{b}$ because they originate from preceding SABL cells, which are currently also in the precharge phase.

At the onset of the evaluation phase (clk switches to 1), P_1 and P_2 stop precharging the cell and N_1 activates the differential pull-down network (DPDN), which is designed accordingly to implement a NAND functionality. After the two input signal pairs have been set to complementary values, one branch of the DPDN conducts and switches the cross-coupled inverters I_1, I_2 to the according complementary state. This state is inverted by I_3 and I_4 and finally reaches the complementary outputs of the SABL cell. The transistor N_2 ensures that all internal nodes of the cell have been discharged at the end of the evaluation phase. In doing so, a constant internal power consumption is achieved.

The functionality of an SABL flip-flop is basically the same. The main difference is that it consists of two serially connected stages. When the first stage is in the precharge phase the second one is in the evaluation phase, and vice versa.

2.4.3 WDDL

The DRP logic style Wave Dynamic Differential Logic (WDDL) has been introduced by Tiri and Verbauwhede [TV04b]. The effort to design WDDL cells is minimized because they are based on standard CMOS cells that are commonly available in cell libraries. WDDL circuits can also be implemented in FPGAs

[TV04d]. The rather simple structure of WDDL cells allows small circuits but also sets boundaries to the achievable PA resistance. The main problems of WDDL in this context are a vulnerability to early propagation [SS06] and an unbalanced internal power consumption.

In WDDL circuits, only the sequential cells are connected to the clock signal. The functionality of the combinational WDDL cells must ensure that if all inputs are set to the precharge value also the complementary outputs get precharged. Figure 2.5 shows how this is achieved in a WDDL NAND cell. For complementary input values of a, $\overline{a}$ and b, $\overline{b}$, respectively, the complementary output values of q, $\overline{q}$ are set according to the NAND functionality. In case all inputs are set to the precharge value, the outputs produce the same value. Precharging of the complementary signals is started at the outputs of the WDDL flip-flops. WDDL flip-flops consist of two stages of two CMOS flip-flops each. One stage stores the precharge value while the other stage stores the current complementary values. Due to the two flip-flop stages, WDDL FFs must be clocked twice as fast as the corresponding CMOS circuit to achieve the same data rate.

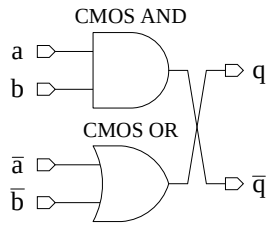


Figure 2.5: Cell schematic of a WDDL NAND cell.

Results of DPA attacks on a test chip using WDDL were presented in [THH+05]. As already mentioned, the PA resistance of WDDL circuits is degraded due to mainly two effects: early propagation and unbalanced internal power consumption. Guilley et al. [GHM+04] proposed a special structure for such DRP cells that avoids these effects. Drawbacks are a reduced cell speed and a significantly increased cell area. Another proposal that aims at mitigating the two weaknesses of WDDL is called Double WDDL (DWDDL) [YS07]. As the name suggests, DWDDL uses two copies of a balanced WDDL circuit, which are operated in a complementary way. DWDDL is primarily intended to be used in FPGAs, but the concept seems to be extendable to application-specific integrated circuits (ASICs) as well. A proposal that tries to reduce the area penalty introduced by DWDDL is called Isolated WDDL (IWDDL) [MMM09].

2.4.4 Masking at the Cell Level

The first special logic styles that have been proposed to counteract PA attacks were all based on the concept of hiding (see Section 2.4.1). Masking was in the beginning mainly used at the architecture level. Only some time later, also several PA-resistant logic styles have been proposed that use masking.

Masking at the cell level basically means that the input and output values of cells are randomized by "combining" them with the so-called masks. Since the resulting masked values are independent of the unmasked values, in theory also the power consumption of the cells is independent of the unmasked values. This makes PA attacks impossible. As shown later in detail, various higher-order circuit effects like different signal delays leads to vulnerabilities of masked circuits in practice.

In the course of the work for this thesis, we have developed three masked logic styles: Masked CMOS (mCMOS), Masked Dual-Rail Precharge Logic (MDPL), and Improved MDPL (iMDPL). Their functionality, properties, and problems are presented and discussed in detail in Chapters 4, 5, and 7. In Section 4.1, the general concept of cell-level masking is presented in detail. Thus, we refrain from repeating the same description here.

As discussed in detail in Chapter 4, the early proposals for masked logic styles like [ISW03, GM04, TKL05] and also mCMOS have the problem that in their security analyses, the negative effects of glitches were not taken into account. In practice, glitches usually lead to a data-dependent power consumption and make PA attacks possible. Several later proposed masked logic styles that take the glitch problem into account include Random-Switching Logic (RSL), MDPL, and Gammel-Fischer Logic (GFL). The GFL style overcomes the glitch problem if each masked input signal of a cell arrives at the same time as the corresponding mask [FG05], which is quite a tough constraint in practice. RSL and its various successors are presented in more detail in the following section. Masked logic styles of this type generally also avoid another problem of masking at the cell level called early propagation (see Section 6.2). Also MDPL suffers from this problem. With the proposal iMDPL we aimed at solving the early propagation problem of MDPL.

2.4.5 RSL

Random-Switching Logic (RSL) is a single-rail masked logic style that has been proposed by Suzuki et al. [SSI04]. As an example of a combinational RSL cell, the transistor schematic of an RSL NAND is depicted in Figure 2.6. The masked input signals a_m, b_m and the masked output signal q_m are related to the corresponding unmasked signals a, b, and q via the random mask bit m in the following way: $a_m = a \oplus m$, $b_m = b \oplus m$, and $q_m = q \oplus m$. en is the enable signal of the RSL NAND cell. During the precharge phase en is 0 ($\overline{en} = 1$) and the output q_m is precharged to 0. At the onset of the evaluation phase en is set to 1 ($\overline{en} = 0$) and the cell is allowed to calculate the result $q_m = \overline{a \cdot b} \oplus m$. Note that the enable signal en must not be set to 1 until the masked input signals a_m and b_m have arrived in order to avoid glitches in the RSL circuit. The enable signal is generated externally by an independent circuit consisting basically of delay lines. This is a rather complex approach that includes non-standard design steps. Furthermore, a new RSL cell library must be built from scratch where all combinational cells have enable inputs.

The main advantage of RSL is that together with the glitch problem also the

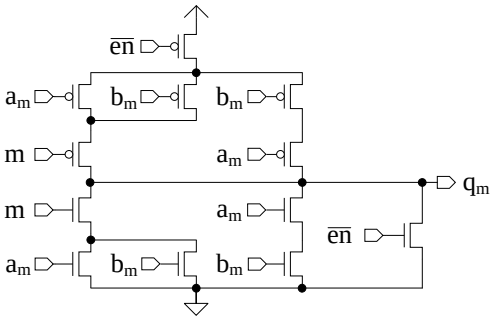


Figure 2.6: Transistor schematic of an RSL NAND cell.

early propagation problem is avoided due to the use of the independent enable signal. However, RSL shows weaknesses to another attack called the "PDF attack" (see Section 6.4). This attack aims at masked logic styles like RSL and also MDPL which use the same mask bit for all signals of a circuit. To overcome this problem, an improved version of RSL circuits has been proposed in [SSI07b]. The proposal includes a re-masking technique to counteract the PDF attack as well as other higher-order DPA attacks. RSL including re-masking has been evaluated on an FPGA.

The drawback that RSL requires an enable signal for each cell which is cumbersome to generate has been addressed in [CZ06]. The proposed logic style is called Dual-Rail Random-Switching Logic (DRSL) and uses an additional circuit in each cell to detect the point in time when it is allowed to evaluate. In doing so, globally generated enable signals are no longer necessary. Unfortunately, our analysis showed that DRSL suffers from early propagation in the precharge phase (see Section 7.1).

Another drawback of the original RSL proposal is that the cells must be implemented from scratch. In [SSSS09], the authors present "pseudo RSL", which addresses this issue. Pseudo RSL still works with globally generated enable signals in order to avoid problems with glitches and early propagation. Furthermore, it is based on commonly available CMOS cells and also uses re-masking to overcome the vulnerability to the PDF attack. Pseudo RSL has been evaluated on a 0.13 μm prototype chip that implements AES [SSSS09, SYO09].

3 Simulation of Power Leakage

Ever since digital circuits have been designed and implemented, it has been important to know at some point in the design process how much power such a circuit will finally consume. Due to the steadily increasing integration densities of digital circuits, the importance of this question has even significantly increased over time.

In modern VLSI design flows, it is essential to know rather exactly and early in the design process how the power consumption of a digital circuit will finally look like and if it will meet the design requirements. The composition and distribution of the power consumption on a chip determines things like the degree and type of cooling that is necessary and the applicability of the chip in domains with restricted power budgets (e.g. contactlessly-powered or battery-powered environments). For cryptographic devices, the appearance of power analysis attacks added another dimension to the impacts of the power consumption. It determines how effective such a device can be attacked in the field.

The power consumption of digital circuits is in general determined via simulations. Such power simulations can be already performed very early in the design process. The farther the design process advances, the more accurately and detailed the power consumption of digital circuits can be simulated. This accuracy obviously also influences how well the part of the power consumption that can be exploited in a power analysis attack is simulated, i.e. how well the data- and operation-dependent effects in the power consumption are considered. We denote the part of the power consumption that can be exploited in a power analysis attack for a specific attack scenario as *power leakage*. For designers of cryptographic circuits it is very helpful to have a technique to simulate this power leakage. It helps them to quantitatively verify that the security requirements in

the context of power analysis attacks are met.

In this chapter, we first investigate the most common power simulation techniques currently used for digital circuits. We discuss their general properties to find out the advantages and disadvantages of the different techniques when it comes to the simulation of the power leakage of cryptographic devices. Next, we present the approach we have chosen and implemented based on this discussion to evaluate the PA resistance of cryptographic circuits with power simulations.

3.1 Power Simulation in General

Power simulations can be performed at different abstraction levels of digital circuits. In our discussion, we distinguish three levels of abstraction (starting from the lowest one): the physical or *analog* level, the *logic* level, and the *behavioral* level. The abstraction levels differ basically in the following properties:

- what signals or values are simulated,
- how are they simulated,
- what devices are part of the netlist that describes the simulated circuit, and
- what power models are used to map the simulated signals and values to power consumption values.

It is clear that at a lower level of abstraction, a higher level of simulation accuracy can be achieved. In the context of power leakage simulation and simulation accuracy, the following features and properties are of importance:

- how well is the fact considered that different signals in a digital circuit usually contribute differently to the overall power consumption,
- to what degree are the timing of signals and the resulting effects like signal glitches considered,
- how much knowledge about the design is necessary, and
- how many resources (simulation time, memory) are required for the simulation.

In general, the higher the level of accuracy, the more resources are required. For a particular simulation case (e.g. a single cell or a circuit implementing a complete encryption algorithm), a reasonable trade-off between accuracy and the usage of resources must be found.

The amount of knowledge required about a design to perform a specific type of power simulation is a measure for how well this type of simulation is also suitable for an attacker of a cryptographic device. During a power analysis attack, the attacker does nothing else than some kind of power leakage simulation based on hypothetical data values, i.e. data values which are assumed to be

processed. The more accurate the resulting power leakage can be simulated, the better attack results can be achieved. However for attackers, it is usually not possible to get detailed knowledge about the circuit design.

An exception to this assumption are the very effective template attacks, which have been introduced in Section 2.2.3. In a template attack, a very accurate power model is constructed by an attacker by characterizing the power consumption with a fully controllable (i.e. the secret key can be configured) cryptographic device, which must be more or less identical to the attacked one. The result of such a characterization is not only a very precise power model of the attacked device but also of the power measurement setup. A typical source of "identical devices" are for example evaluation kits containing the attacked cryptographic device.

In the following sections, the different abstraction levels of power simulations are discussed in detail according to the various properties mentioned before.

3.1.1 Analog Level

The most precise way to simulate the power consumption of digital circuits is to perform a transient analog simulation, i.e. a simulation of the instantaneous voltage and current values in a circuit over time. The instantaneous power consumption is then simply the product of both values. From the point of analog simulation, a digital circuit is just a specific type of analog circuit.

The basis of such a simulation is a netlist that contains all transistors of a digital circuit and the connections between them. Furthermore, this netlist usually contains the parasitic elements of the circuit. Parasitic elements (capacitors, resistors, indictors) occur in digital circuits due to the way the circuits are manufactured. These elements are actually unwanted. In particular, there are parasitic capacitances between the wires of a circuit, and there are also unwanted capacitances in the transistors. The size of the parasitic elements mainly depends on the used process technology.

The number of parasitic elements that occur in digital circuits is very large in practice. This is why certain simplifications are usually made in order to reduce the complexity of the analog simulation. A very common simplification is to model all parasitic capacitances of a cell or wire as a single capacitance at the output of a cell. This model is commonly known as the lumped-C model. In the lumped-C model, it is also assumed that wires have a negligible resistance compared to the resistance between the drain and source terminals of MOS transistors. Due to these simplifications, simulations based on the lumped-C model not always precisely describe the instantaneous power consumption of a circuit. Clearly, the more precisely the parasitic elements of a circuit are modeled, the more precise is the simulation result.

Independent of the fact whether the used transistor netlist is based on certain simplifications or not, a transient analog simulation is always performed in the same way. The analog circuit simulator takes the transistor netlist and calculates the instantaneous voltage and current values that occur in the circuit based on difference equations and comprehensive device models (e.g. BSIM transistor

models [BSI]). This requires a lot of resources. Hence, analog simulations are typically only used for small, critical blocks of a digital circuit and not for the whole circuit. On the other hand, in analog simulations all the relevant effects for power analysis attacks (individual signal weighting, exact signal timing) can be considered very well.

Examples of analog circuit simulators are *SPICE* [Rab] from the University of California at Berkeley [Unib], *Spectre* from Cadence Design Systems [Cad], and *NanoSim* from Synopsys [Syn]. SPICE (Simulation Program with Integrated Circuit Emphasis) is the most famous analog circuit simulator and this is the reason why analog simulations are often also called SPICE simulations. SPICE is the ancestor of many other simulators [Pes]. Analog circuit simulators differ mainly in terms of their field of application (analog circuits, digital circuits), their speed, and their accuracy.

3.1.2 Logic Level

Power simulations at the logic level usually require less resources than analog circuit simulations at the price of a lower precision. The basis of power simulations at this level is a netlist that contains the logic cells of the circuit and the connections between them. Power simulations at the logic level conceptually happen in two steps. First, the logic values and transitions that occur in a digital circuit are simulated. In the second step, these logic simulation results are mapped to power consumption values.

To simulate the logic values and transitions in a digital circuit, the Boolean equations describing the logic cells in the netlist are evaluated for the given input signals. An important property that can be considered in logic simulations are the signal delays that are caused by logic cells and wires. If these delays are not considered (i.e. all cell and wire delays are assumed to be 0), we speak of a zero-delay simulation. In such a case, effects like signal glitches do not occur. In order to basically consider the effects of signal delays, unit-delay simulations can be performed. Here, the propagation delays of all cells from any input to any output are set to some default "unity" value, e.g. $1\,ns$. It is clear that such an assumption is in general still very inaccurate. However, it already allows to at least rudimentarily consider signal glitches and other effects.

The highest level of accuracy for logic simulations with respect to timing can be achieved if the signal delays are determined based on the final layout of a digital circuit. In this case, the exact positions of the logic cells and the interconnecting wires are known. Thus, the values of the parasitic elements can be estimated quite well (most importantly the overall output loads of cells), which in turn allows to accurately estimate the signal delays occurring in a circuit. The process of introducing accurate signal delay information into logic simulations is called back-annotation.

Clearly, the power simulation results are the most precise if the underlying logic simulation is done with back-annotated delay information. In particular the occurrence of glitches depends heavily on the exact values of signal delays. In summary, the final result of the logic simulation step is a list that states for

each signal in the netlist which logic value the signal has and when transitions occur. The most common logic simulators used today are *ModelSim* from Mentor Graphics [Men], *Incisive* (also known as NCSim) from Cadence Design Systems [Cad], and *VCS* (Verilog Compiler Simulator; the simulator is not restricted to the hardware description language Verilog) from Synopsys [Syn].

The second step of a power simulation at the logic level is to map the logic simulation results to power consumption values. For this purpose, it is necessary to have some kind of power model that describes how the logic simulation results are quantitatively related to power consumption values. Typically, such power models take as input not the individual transitions of a specific signal and the time when they happen, but the more abstract value of the "transition probability" of each signal. A transition probability of 100 % means that a signal toggles in every clock cycle (thus, this value is often also called toggle probability). This indicates that the result of such power models is in general only the average power consumption of the simulated digital circuit and not the instantaneous consumption. Unfortunately, such average power consumption values are not very useable in power analysis attacks. This is the reason why we use a different type of power model in our approach of power simulation at the logic level. Our approach is presented in detail in Section 3.2.

The power models used for power simulation at the logic level are often included in modern standard-cell libraries and describe the contribution of each cell to the overall, average power consumption of a digital circuit. Usually, these models are parameterized by the capacitive load at the output of a cell and by the slew rates of the input and output signals. Besides the power that is necessary to load the wire at the output of a cell, the models also include the internal power consumption of a cell when input and/or output signals toggle.

The tools in a VLSI design flow for digital circuits that actually calculate power consumption values are typically the logic synthesizers or the place-and-route tools. For this task, these tools take as input the transition probabilities from the logic simulators. Cell output loads and signal slew rates can usually be determined by the tools itself (at different degrees of accuracy of course). Looking at the preceding discussion, it is clear that power simulations at the logic level require a very detailed knowledge about the design of the digital circuit. Effects that are relevant for power analysis attacks like the individual weights of signal transitions and signal timings can be considered quite well. The main problem is that power simulations at this level usually provide average power consumption values and not instantaneous values.

3.1.3 Behavioral Level

Power simulations of digital circuits at the behavioral level are a very fast method, but compared to simulations at analog and logic level a much less accurate one. The basis of a power simulation at this level is a high-level description of the digital circuit. This high-level description contains the major components of the digital circuit (microcontrollers, memories, dedicated hardware modules, functional blocks, etc.). Furthermore, the functionality of the

various components is described in a data-flow manner, i.e. with functions that describe how the data inputs of a component are mapped to output values. The border between behavioral and logic level is reached just before all components in the "high-level description" actually represent logic cells that are available in standard-cell libraries.

In order to map the activities of the components to power consumption values, some high-level power models of these components are necessary. These models might only consider that a component is active at some point in time or it might also consider the values of the data entering and leaving a component. In the context of power analysis attacks, only those behavioral-level power simulators are of interest that also include the data-dependent and the operation-dependent portions of the power consumption. Power simulations at the behavioral level are usually used to assess the average power consumption of complex circuits. As described for power simulation at the logic level, this makes such simulators rather inappropriate in the field of power analysis attacks. Examples of behavioral-level power simulators are instruction set power simulators for microcontrollers like *SimplePower* [IKV01] and *JouleTrack* [SC01].

An important step in power analysis attacks is to map data values, which are assumedly being processed in cryptographic devices, to power consumption values. This can be regarded as a kind of power simulation of a very small part of the cryptographic circuit on the behavioral level. In general, attackers do not have much knowledge about the design of the cryptographic device they attack. On the one hand, this means that usually only a high-level description of the circuit is available, e.g. only the algorithmic description of a circuit that performs encryptions. What intermediate results actually occur in the circuit can only be assumed. On the other hand, this also means that attackers can only employ rather simple power models. However, the power models do not need to provide absolute power consumption values in the case of power analysis attacks. Relative values which indicate whether the power consumption is "higher" or "lower" are basically sufficient.

In the following, we present the most common power models for attackers in detail. Currently, these are the *Hamming-distance* (HD) model, the *Hamming-weight* (HW) model, and the *zero-value* (ZV) model. Also some variants of them are briefly addressed. What all these power models commonly assume is that all occurring data values are binary encoded in a straight-forward manner. These binary encoded values then directly represent the logic values carried on individual signal wires. Furthermore, a specific logic value of a signal wire corresponds to a specific voltage level in the circuit, which means that when a logic value changes, also a voltage level of a wire changes and power is consumed.

Hamming-Distance Power Model

As said before, an attacker usually does not have detailed knowledge about the attacked design. Nevertheless, the information about the basic functionality and type of a cryptographic device (e.g. what cryptographic algorithms are used; is

the device a microcontroller or an ASIC design) and an understanding of the architectures of digital circuits in general allow an attacker to make educated assumptions about the structure of some parts of a cryptographic device. This usually allows a basic estimation of the power consumption of this part of the device.

The basic idea of the HD power model is that the power consumption of the circuit is proportional to the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions that occur at a specific point in time. If two intermediate values v_1 and v_2 appear consecutively in a part of the circuit, the HD model simply counts the number of transitions that occur. The HD of two values v_1 and v_2 is formally defined as the HW of $v_1 \oplus v_2$. The HW corresponds to the number of bits that are one, and hence, $HW(v_1 \oplus v_2)$ corresponds to the number of bits that differ in v_1 and v_2.

$$HD(v_1, v_2) = HW(v_1 \oplus v_2) \tag{3.1}$$

In case of the HD power model, the basic assumption thus is that the attacker knows two intermediate results in a circuit which are processed consecutively. For example in a microcontroller, it is very likely that the intermediate results occurring in the executed algorithm are transferred over an internal bus in the order the results appear in the algorithm. Also glitches typically do not occur on internal buses. The capacitive loads of buses is usually quite big, and hence, activities on the buses contribute significantly to the overall power consumption of a microcontroller. Therefore, glitches are often avoided by driving the bus lines only through sequential cells to keep the power consumption low. In case of glitches, the actual consecutive data values are unknown and the HD model would not be appropriate. This is also the reason why intermediate results of combinational logic blocks can typically not be attacked with such a generic power model like the HD model [MPO05]. Furthermore, as postulated by the HD power model, it can typically be assumed that the capacitive loads of the individual wires of a bus are about equal and that it does not matter whether $0 \rightarrow 1$ or $1 \rightarrow 0$ transitions occur, i.e. all transitions of individual bus signals cause a similar power consumption.

Also the power consumptions of registers can be described rather well with the HD model. The reason is that glitches do not occur at the output of registers because they are triggered by a clock signal. Therefore, the HD of the two values stored consecutively in a register describes its power consumption usually very well. It can typically be assumed that registers are used at the outputs of most functional blocks in a digital circuit.

The HD power model does not consider differences in individual transitions due to parasitic capacitances of signal wires or logic cells. Another point is that the HD model also completely ignores the static power consumption of the cells. In variants of the HD model, this is changed. For example, transitions of individual signals can be weighted differently or a difference can be made between $0 \rightarrow 1$ and $1 \rightarrow 0$ transition. Such models are of course less generic and typically specific for a particular kind of cryptographic device.

Hamming-Weight Power Model

In case of the HW power model, the basic assumption is that the power consumption of a specific part of a digital circuit depends solely on the number of bits that are one at a specific point in time. Obviously, this power model is not very accurate for digital circuits, which are nowadays mainly implemented in the CMOS logic style. In such a circuit, the power consumption is mainly determined by its dynamic part caused by signal transitions (see Section 2.3.1) and not by the values of signals itself.

However, it can happen that the HD power model is not useable because the attacker cannot identify data values that are processed consecutively. In these cases, changing to the simpler HW power model, which needs only information about a single data value, might still lead to successful attacks. The reason is that what the HW model predicts is usually not completely unrelated to the actually occurring power consumption. We show this by looking at the general case where v_1 and v_2 are consecutively processed and an attacker uses $HW(v_2)$ as estimation of the power consumption. In the general case, the bits in v_1 and v_2 are both uniformly distributed and each bit is independent of the corresponding bit in the other value. Note that the following considerations are also valid for the transition from v_2 on to the next processed value v_3.

In a real digital circuit, the assumption that the power consumptions caused by $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions are equal (as made by the HD model) is usually not completely true. For example, the power consumption for a $0 \rightarrow 1$ transition might be higher than for a $1 \rightarrow 0$ transition. The basic reason for this behavior is described in Section 2.3.3. In such a case, the actual power consumption is on average higher if the HW of the processed value v_2 is high, because this leads on average to more $0 \rightarrow 1$ transitions than to $1 \rightarrow 0$ transitions. Thus, there is in most cases an at least small relation between the HW of the processed data and the power consumption.

Other scenarios, for example where the bits in v_1 are not uniformly distributed, are discussed in detail in [MOP07]. Note that these cases are usually more advantageous for the HW power model. However, the general rule is to use the HD power model whenever possible. A variation of the basic HW power model is to weight the individual bits in a value according to their (estimated) capacitive weights in the digital circuit.

Zero-Value Power Model

Various reports of successful attacks on cryptographic circuits in the last years, e.g. [GT03, Gou03, AT03], show that also another generic power model is of importance. In this power model, one assumes that a part of a circuit consumes a significantly different amount of power if a particular data value is processed.

A very common case is that this happens for the value 0. For example, a multiplier often requires much less power in case one operand is zero. This power model is called the zero-value model. Such a model is often also suitable to attack combinational blocks that show an according behavior [MPO05]. This

is a major advantage compared to the HD or the HW power models.

3.1.4 Comparison

Table 3.1 gives a compact overview of the different power simulation techniques discussed in the preceding sections. In particular, the techniques are categorized whether or not they are also useable by attackers of cryptographic circuits. What is also very important in the context of power analysis attacks is how a specific simulation technique can consider the individual weights of signal transitions and the individual timing of signals.

3.2 Power Leakage Simulation at the Logic Level

Designers of cryptographic circuits want to know during the implementation process how much power leakage (i.e. exploitable information in the power consumption for a specific attack scenario) their devices eventually have. Furthermore, they want to know if the applied countermeasures against power analysis attacks work efficiently. Calculating the power leakage formally is usually not possible. The first reason is that cryptographic circuits are very complex systems, which makes this task practically infeasible for a circuit as a whole. But also for small parts of it, which can in principle be handled, very complex models would be necessary to consider all effects that influence the power consumption of a real circuit.

A usual solution to this problem is simulation. This means, the power consumption of a cryptographic device is simulated during implementation in a suitable way and power analysis attacks are performed on the simulated data to determine the power leakage. Note that a designer performs in this case two power simulations: one to determine the power consumption of the cryptographic circuit and one in the power analysis attack when mapping apparently processed data values to power consumption values. A classical attacker that has to work with the cryptographic devices in real silicon and power measurements performs only the second type of power simulation.

The way how the power consumption of cryptographic circuits is simulated during implementation is defined by a trade-off between simulation accuracy and resource usage. On the one hand, simulation time and memory requirements of the simulation should not be too high. When looking at Table 3.1, this indicates that a power simulation at the analog level is often not suitable for entire cryptographic circuits. Such simulations are typically used for small, crucial parts of a circuit, if maximum accuracy is desired (e.g. [TV03, RBE+07]). On the other hand, the main effects that determine the power leakage like all data- and operation-dependent circuit activities, the capacitive loads of signal wires, and the individual timing of signals must be considered sufficiently well in the power simulations. Thus, also simulations at the behavioral level as shown in Table 3.1 are in general not suitable.

Table 3.1: Compact comparison of different power simulation techniques at different abstraction levels.

Simulation level	Analog	Logic	Behavioral			
Simulated signals	Instantaneous voltage + current values	Logic values and transitions	Data values			
Basis of simulation	Difference equations + detailed device models	Boolean functions + cell and wire timing models	Functions describing data flow			
Netlist content	Transistor netlist with parasitic elements (opt.)	Logic cells	Circuit components			
Power models	$p(t) = u(t) * i(t)$	Cell power models	Component power models	HD power model	HW power model	ZV power model
Simulation result	Instantaneous power in Watt	Average power in Watt	Average power in Watt	Number of transitions	Number of logic ones	Is Value zero?
Transition weighting	Individually per voltage/current signal	Individually per logic signal	Individually per data value	Equally for all values	–	–
Signal timing	Individually per voltage/current signal	Individually per logic signal	No	No	No	No
Useable also by attackers	No	No	No	Yes	Yes	Yes
Resource usage	Very high	High/medium	Low	Very low	Very low	Very low

Hence, power simulations at the logic level are often considered as the most suitable compromise to simulate the power leakage of cryptographic devices. According to Table 3.1, their demand for resources is moderate and all major effects influencing the power leakage can be considered in the simulation to a reasonable degree. However, the table also indicates that existing power simulators that work on the logic level cannot be directly used since they usually provide only average power consumption values and not instantaneous values. Therefore, the general approach must be adapted to overcome this problem.

3.2.1 Deriving the Instantaneous Power Consumption from Logic Simulation

The proposed approach for power simulations at the logic level is the following. First, the behavior of all logic signals in the cryptographic circuit is determined by a logic simulation of the final cell netlist (i.e. the cell netlist after placing and routing the circuit - see Section 5.2.1). Such a simulation includes the activities of all signals in the cryptographic circuit. Thus, also all the data- and operation-dependent signal activities are considered. Furthermore, also the accurate timing of signals can be considered in the logic simulation (in the best case through back-annotation from the circuit layout). In the second step, the signal values and activities are mapped to instantaneous power consumption values with the help of an appropriate power model. In the power model, it is possible to consider the individual weights of transitions due to the different parasitic loads of the wires in the circuit.

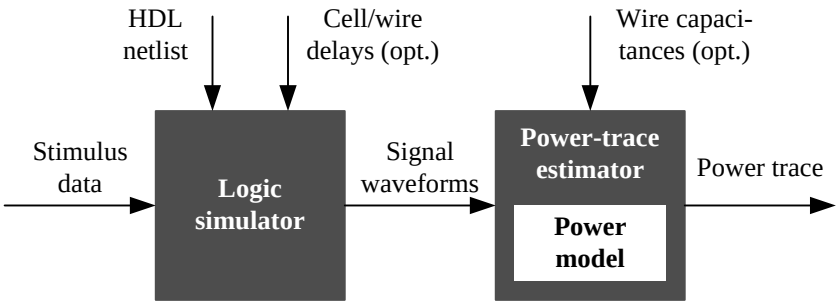


Figure 3.1: Concept of estimating the instantaneous power consumption of a digital circuit from logic simulation.

In Figure 3.1, the basic concept of estimating the instantaneous power consumption of a digital circuit based on the results of logic simulation is depicted graphically. The logic simulator takes as input the HDL description of the circuit netlist and some stimulus data that is applied to the circuit inputs. The HDL netlist also includes the Boolean descriptions of primitive logic cells if these are present. Optionally, also the delay information of cells and wires is read, which allows a more accurate timing of the activities of the logic signals.

The output of the logic simulation—the signal waveforms—are next fed into the power-trace estimator. The estimator converts the information in the signal waveforms into instantaneous power consumption values with the help of an appropriate power model. The instantaneous power consumption values, which are nothing else than a simulated power trace, constitute the output of the power-trace estimator.

There are several possibilities for the interface from the logic simulator to the power-trace estimator. A flexible approach is to write the signal waveforms in an appropriate format to a file, which is then read by the power-trace estimator. A common format for signal waveforms is for example the value change dump (VCD) format, which is defined in the IEEE standard of the Verilog HDL [IEE01]. A VCD file basically contains the initial value of all recorded signals and time markers with the new values of signals that change at these points in time.

Another approach is to hook the power-trace estimator directly into the simulation engine of the logic simulator to get the signal waveforms. This is for example possible via standardized programming language interface (PLI) routines of the respective HDL or the simulator itself provides a proprietary interface to directly access the simulation data. The main advantage of this approach is that it is in general faster and requires less memory since no intermediate file must be written. Its main drawback is lacking flexibility because the approach depends much more on the used HDL and logic simulator.

In the next two sections, the steps "logic simulation" and "power-trace estimation" are discussed in detail.

Logic Simulation

Logic simulation of a digital circuit is possible at different implementation steps. An early, high-level behavioral description of a circuit will not yet contain all signals that are finally present. Therefore, it is recommended to simulate the circuit after place and route if the power leakage is analyzed. At that design stage, the netlist already contains all logic cells and signals that will be present on the final chip. However, it typically also makes sense to simulate already at an earlier stage and perform a power leakage analysis. This often allows drawing first conclusions on the most vulnerable parts of a design in terms of power analysis attacks. In such simulations, detailed information about a circuit like signal delays and wire capacitances are not yet available.

In the logic simulation, usually the waveforms of all signals are recorded in order to include all data- and operation dependent activities. The internal signals of any primitive logic cells are not included, since these signals are usually not modeled as they behave in the "real" cells. The risk that internal signals leaking information are completely missed is generally not given since the activities of internal signals are always related to input and/or output signals of logic cells. Furthermore, the contribution of internal signals to the overall power consumption is in general very small. If only the power leakage of a specific module should be analyzed, the logic simulation is configured to record only the

waveforms of the relevant signals.

In the logic simulation, it is important to consider the individual signal delays caused by logic cells and wires as accurate as possible. These delays are directly responsible for effects like signal glitches and describe a possible early-propagation behavior of the cells. Both effects strongly influence the PA resistance of a cryptographic circuit. The possible timing modes of a logic simulation consider the individual signal delays very differently.

Zero delay mode: In a logic simulation using this timing mode all cell and wire delays are set to 0. No signal glitches and no early propagation occur. All signal transitions either occur at clock events or when input signals change. This very elementary timing mode is used to simulate the basic power leakage that is caused by the occurring intermediate values directly and not by any higher-order effects like glitches.

Unit delay mode: If the simulation is based on this timing mode the propagation delay of each primitive cell is set to the same unity value (typically $1\,ns$). Glitches occur in such a simulation. While no early propagation of primitive cells is considered, a unit delay simulation already shows if any modules of the circuit calculate faster or slower for specific input values (e.g. for the input value 0). If this is the case for a module, it indicates a power leakage that is exploitable with the ZV power model. A unit delay simulation is as simple as a zero delay simulation but already allows to basically consider the main higher-order effects that lead to power leakage. Still, some leakage might go undetected since the delay model is very uniform.

Random delay mode: This timing mode is very similar to the unit delay mode. The main difference is that the propagation delay of each logic cell is set to a random value. While keeping the simplicity of the unit delay mode, the random delay mode helps to detect power leakage that would go undetected if too uniform delay values are used.

Back-annotated delay mode: In the most accurate timing mode, the individual signal delays caused by cell and wires are extracted from the layout of the design and are back-annotated to the logic simulation. Only the knowledge of the layout of a circuit allows to rather exactly determine the parasitic loads of wires, which have a significant impact on the individual signal delays. On the other hand, this also means that the back-annotated delay mode requires the highest effort because the design must already be placed and routed and a parasitics extraction and delay calculation must be run. In the context of power analysis attacks based on logic simulation, this timing mode provides the most exact figures about the amount of power leakage.

Power-Trace Estimation

The logic simulation provides detailed information about signal values and transitions in a digital circuit. The second step in the proposed simulation methodology of the instantaneous power consumption of a digital circuit is the mapping of the signal activities to power values.

The power models used for this mapping usually consider only the dynamic part of the power consumption, i.e. that part that is caused by signal transitions. The dynamic power consumption still makes up the major part of the overall power consumption of digital circuits and shows the highest data dependency. However, for process technologies with feature sizes smaller than $130\,nm$, also the static power consumption and its data dependency is becoming important [WH04]. Therefore, it will be essential to consider also this type of power consumption when analyzing the PA resistance of cryptographic circuits implemented in such advanced process technologies.

An important aspect of the power-trace estimation is how the signal values and transitions reported by the logic simulation are weighted in their contribution to the overall power consumption. This contribution basically depends on the parasitic properties of the cells driving the signal wires and on the parasitic properties of the signal wires itself. It is clear that this contribution is different for each signal in reality. Various simulation power models are possible which consider the weighting of signals in a variety of ways.

Unit weighting model: In this power simulation model, each signal transition is weighted with the same value (usually 1), independent of the particular signal and the transition type. This is a very simple and inaccurate power model, but it can be used very early in the design process and can help to basically identify critical parts of a design. However, various effects causing a power leakage in the context of PA attacks will not be detected with such an elementary model.

Random weighting model: A power model that avoids the shortcomings of the very uniform unit weighting model is the random weighting model. In this case, a random weighting value is assigned to each signal transition. This comes much closer to the real situation and is still possible relatively early in the design process.

Back-annotated weighting model: This power model is only possible after the placement and routing steps of the circuit design flow have been finished. Only at this stage, the individual signal weights can be estimated rather exactly from the layout of the design. In the context of power analysis attacks based on logic simulation, this power model provides the most exact figures about the amount of power leakage.

In the power-trace estimation methodology outlined above, the simplest approach is to consider the dynamic contribution of a signal transition only at the point in time when the transition occurs. In the next time step, the power

consumption caused by this signal transition is zero again. A much more sophisticated method would be to disperse the whole energy of a switching event over the time range the event actually occurs (according to the signal rise or fall time). This would be a much more complex approach and would also significantly reduce the speed of the proposed power estimation technique.

In a power-trace estimator based on logic simulation, it is easily possible to analyze the individual contribution of each signal to the overall power consumption and to select which type of signal transitions shall actually be considered. Very common is the so-called "toggle count" mode. In this case, the $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions of all signals in a circuit are considered with unit weighting in the analysis.

The main pros and cons of power-trace estimation based on logic simulation in the context of PA-resistance analysis can be summarized as follows. This simulation approach is rather fast and has a low memory usage compared to the very accurate but slow analog-level simulation. On the other hand, its underlying power model still allows considering most of the major PA-leakage effects. What is usually not considered correctly is the internal power consumption of primitive logic cells, because their behavior is only specified in a logical way via Boolean functions. However, the internal power consumption of primitive cells is typically very small. The presented power simulation approach is also limited in considering how the effective size of a wire capacitance is influenced by the state of adjacent wires. Furthermore, the short-circuit power consumption and its dependence on signal rise and fall times is usually not included. In power-trace estimations based on logic simulation, it is also usually assumed that power is only consumed at the point in time a signal transition occurs. In reality, this event takes a specific amount of time depending on the cell driving strength and the electrical properties of the signal wire. A main advantage of the simulation approach is that it can be used very early in the design process, even already with the high-level circuit model. In this case, not all signals will already be present in the circuit, but the existing ones are often sufficient to identify the critical parts of a design.

3.3 Power-Trace Estimation Based on Logic Simulation in Practice

Two versions of power trace estimators based on logic simulation have been practically implemented. The first implementation is a stand-alone tool called "Transition-Trace Generator" (TTG) written in the programming language *Java*. An earlier name of the TTG was "vcd2pt" (VCD file to power trace). This tool has been used and evaluated in a master thesis [Ste06]. In another investigation, a comparison of the run times of power trace simulation on the analog level (simulator NanoSim from Synopsys [Syn]) and on the logic level (logic simulator NCSim from Cadence [Cad] and power trace estimator vcd2pt) has been made. The power consumption of an AES encryption module suitable for RFID tags

[FDW04] implemented in the PA-resistant logic style SABL (see Section 2.4.2 for the details of SABL) has been simulated with NanoSim and NCSim/vcd2pt, respectively. In both cases, the netlist of the AES RFID module with extracted circuit parasitics from the design after place and route has been simulated. This means for the power trace estimation with vcd2pt that the logic simulation has been done with back-annotated delay information. The simple "toggle count" power model has been used by vcd2pt. The power simulation results have subsequently been used in a DPA attack. In case of NanoSim, the power simulation and the DPA attack took about 15 hours while in the case of NCSim/vcd2pt, the whole process took just around 20 minutes. This tremendous speed-up factor of 45 comes at the expense of accuracy for the NCSim/vcd2pt simulation. However, the NCSim/vcd2pt simulation has already helped to successfully identify and solve PA-related problems in the design.

The second version of a power trace estimator that has been practically implemented is called "vcd_analyzer". It is part of IAIK's Side-Channel Analysis Toolbox (SCA Toolbox) [Ins] for MATLAB [The]. The vcd_analyzer has not been entirely written in the MATLAB scripting language, for performance reasons parts of it have been implemented in the programming language *C*. The main features of the vcd_analyzer are the following:

- As the tool name suggests, VCD files are read and the signals in it are analyzed. If recorded signals should be excluded from the analysis, they must be removed in the VCD file header.
- Time ranges need to be specified. All signal activities within a time range are analyzed as if they happened at the same time, which allows for example to summarize all activities per clock cycle. No summation in the time dimension occurs if the time ranges match the logic simulation resolution of, for example, $1\,ns$.
- It can be specified to sum up the activities of all signals within a time range or to separately count the activities per signal.
- It can be specified which signal activities are counted: either only the transitions $0 \to 1$ and $1 \to 0$ are counted as a single sum value or all 16 types of transitions ($0 \to 0$, $0 \to 1$, $0 \to X$, $0 \to Z$, $1 \to 0$, ..., $Z \to X$, $Z \to Z$) are counted as 16 separate values.
- Per signal and for each of the counted signal transitions, either the weight 1 can be used for all of them ("unit weighting" mode) or individual weight values can be specified. The latter allows realizing the analysis modes "random weighting" and "back-annotated weighting".

In [KP07], we compared the results of power-trace estimation based on logic simulation, the results of analog-level simulations, and the results of real power measurements. The results of power-trace estimation based on logic simulation show a close resemblance to the other results, in particular when the power traces are used in DPA attacks. Recently, it has also been proposed in [MSSE09]

to extend the toggle-count model with individual signal weights and to treat $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions differently. This approach significantly increases the accuracy of power-trace estimation based on logic simulation while its good performance is maintained.

3.3.1 Simulated DPA-Attack on the Demo Circuit "AES-encinit" Implemented in CMOS

For a practical investigation on how well the power trace estimation based on logic simulation with the tool "vcd_analyzer" works, the demo circuit "AES-encinit" described in the following has been analyzed in a simulated DPA-attack. The demo circuit implements parts of the initial operations of an AES encryption [Nat01] run, namely an 8-bit AddRoundKey operation and the subsequent 8-bit SubBytes operation as shown in Figure 3.2. The flip-flops InFF, KeyFF, and OutFF store the input byte, the key byte, and the resulting output byte. The SubBytes operation is calculated arithmetically in the finite field $GF(2^8)$ according to [WOL02]

The high-level description of the demo circuit has been synthesized with the $0.35\,\mu m$ CMOS standard-cell library C35 from austriamicrosystems [aus]. Since this design has also been used to investigate the masked logic styles mCMOS, MDPL, and iMDPL (see Chapters 4, 6, and 7), the cell usage during synthesis has been restricted so that the resulting netlist could have been directly translated into such circuits. More specifically, this means that only buffers, inverters, basic combinational 2-input cells like NAND, NOR, and XNOR, flip-flops including the versions with asynchronous reset and/or preset functionality, and tie cells to provide constant logic values have been allowed.

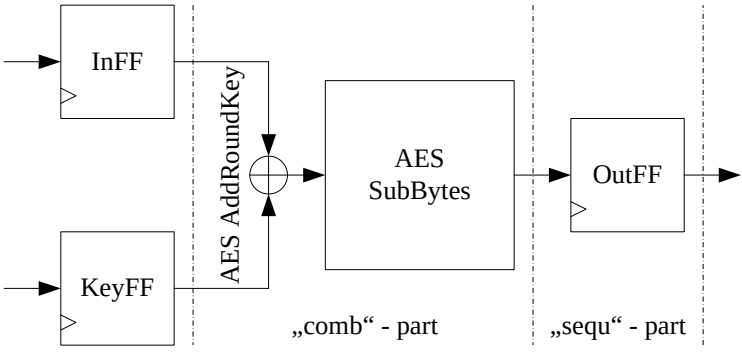


Figure 3.2: Separation of the AESencinit demo circuit into a combinational ("comb") part and a sequential ("sequ") part.

Simulation Flow, Attack Flow and Terminology

As indicated in Figure 3.2, two parts of the synthesized demo circuit have been thoroughly simulated with the logic simulator NCSim from Cadence [Cad]: a

combinational part including the AddRoundKey and the SubBytes operation, which is in the following called **"comb"** and a sequential part including the eight flip-flops storing the result of the comb-part, which is in the following called **"sequ"**. Both parts represent the two general circuit types of digital circuits. The comb-part of the circuit has been simulated for all possible combinations of input value transitions of the AESencinit circuit. This means, since we have an 8-bit input value (key value is fixed), that $256^2 = 65\,536$ power traces have been simulated. The first trace represents the power consumption when the input value switches from 0 to 0, the second trace represents it when the input value switches from 0 to 1, etc. The sequ-part of the circuit has been simulated for 256^2 random input values of the AESencinit demo circuit, which are processed consecutively. The reason for the different simulation approaches is the following. When the masked implementations of the AESencinit circuit are simulated (where also mask bits are inputs), doing that also for all possible combinations of input value transitions in case of the sequ-part would lead to far too many simulations. Thus, all combinations of input value transitions are only used when simulating the comb-part and the same number of random values is used to simulate the sequ-part.

The logic simulation has been done down to the CMOS cells, i.e. all logic signals have been recorded, excluding the internal signals of the logic cell models since they often do not represent the real situation and are assumed to cause a rather low contribution to the overall power consumption. In some investigations, only the 8-bit output signals of the comb-part or the sequ-part have been used. In this case, the simulation results have been denoted as **"combout"** or **"sequout"**, respectively. Furthermore, the logic simulation has been performed in two timing modes: zero delay and unit delay. In the mode **"zerodelay"**, effects like signal glitches and early propagation of cells do not occur while they occur in the mode **"unitdelay"**. This approach allows evaluating the influence of these effects on the DPA resistance of the demo circuit.

The simulated signals have then been read into MATLAB with the tool vcd_analyzer. Basically, only the signal transitions $0 \to 1$ and $1 \to 0$ have been considered. With respect to signal timing, the analysis has been performed in two ways. Either all signal activities within a clock cycle have been summarized (analysis denoted as **"cc"** in the following) or the analysis has been made with the same time resolution as the logic simulation (analysis denoted **"tr"** in the following). The reason for the cc-analysis is the following: The combination of a unitdelay simulation, which includes glitches and early propagation, and the cc-analysis allows getting a power simulation result that only includes the effect of glitches and not the effect of early propagation. This allows to separately investigate the impacts of both effects. With respect to signal weighting, the analysis with vcd_analyzer has been performed in two settings. Either all signal transitions have been weighted equally with 1 (analysis denoted in the following as **"equal"**) or the signal transitions have been weighted with uniformly distributed random values from the interval $[1.5, 2.5]$ (analysis denoted in the following as **"random"**).

The subsequent first-order DPA-attack has also been performed with MATLAB. As a first point of attack, the intermediate result after the AddRoundKey operation has been selected. This result is equal to the data value entering the SubBytes circuit and has been denoted as **"SBin"** in the following analysis. The power model that has been used at this attack point is the **ZV** model considering the whole resulting byte. As secondly attacked intermediate result, the output of the SubBytes circuit has been chosen. This result is also equal to the data value entering (in the current clock cycle) and leaving (in the next clock cycle) the eight flip-flops at the output of the demo circuit. This intermediate result has been called **"SBout"** in the analysis. The power models that have been used to attack the intermediate result SBout are the **HW** and the **HD** model considering the whole resulting byte and the **HWbit**$\langle x \rangle$ and the **HDbit**$\langle x \rangle$ model considering only a specific bit of the result ($\langle x \rangle$ is a number between 0 and 7, which specifies the attacked bit; bit0 denotes the least significant bit).

An example identifier for an attack result is **combout/zerodelay/random/cc/SBout/HD**. It means that the logic simulation has only included the eight output bits of the comb-part of the AESencinit demo circuit and that the simulation has been done in the zero delay timing mode (no glitches, no early propagation). Furthermore, the tool vcd_analyzer has used random weighting for the signal transitions and the signal activities have been summarized within each clock cycle. In the DPA attack, the output value of the SubBytes module has been predicted and the HD power model has been used to map intermediate values to power consumption values.

The various simulation and attack variants introduced above allow to thoroughly analyze which parts of cryptographic circuits are vulnerable to which types of PA attacks.

Attack Results

The logic simulation of the CMOS implementation of the AESencinit demo circuit has been performed with a clock cycle time of 60 ns, i.e. the clock transitions happen at 0 ns, 60 ns, etc. The 8-bit secret key used in the simulation has been 165 (0xA5).

Attacks on the combinational part of the demo circuit: The attack result **comb/zerodelay/equal/cc/SBout/HD** depicted in Figure 3.3 shows the well-known result that a DPA leakage occurs for CMOS circuits even if effects like glitches and early propagation are not considered. A correlation peak of around 0.147 (and not 1.0) for the correct key has been achieved because of the parallel activity of other signals in the comb-part. Furthermore, all signal activities have been summed up in one point. The attack **.../SBout/HW** does not show a DPA peak because the HW power model does not work if the power is simulated in the "toggle count" mode, i.e. if the transitions $0 \rightarrow 1$ and $1 \rightarrow 0$ are both weighted with 1 (unit weighting).

With the argumentation at the beginning of the last paragraph in mind, one might assume that if the power simulation includes glitches, i.e. with

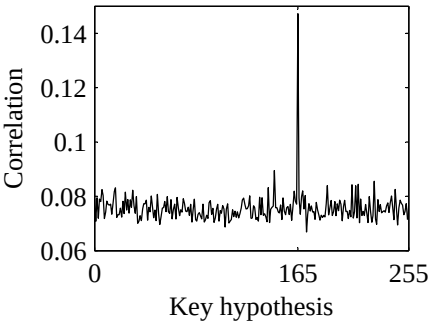


Figure 3.3: DPA attack result **comb/zerodelay/equal/cc/SBout/HD**.

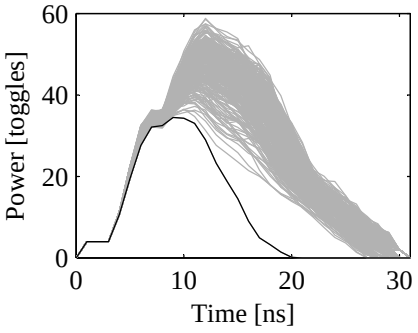


Figure 3.4: Average power traces for the simulation **comb/unitdelay/equal/tr** for different input values of the comb-part; black curve: input of comb-part is 165 ⇒ input of SubBytes block is 0.

a unitdelay simulation, the DPA peak will get higher. However, this is not the case because the inclusion of glitches has also reverse effects that are more dominant in case of a combinational circuit. Since glitches occur also for signals that are not attacked, the noise level rises. Furthermore, glitches significantly reduce the accuracy of the HD power model because the value transitions taken as basis to calculate the HD actually do not occur. This explains why the attack **comb/unitdelay/equal/cc/SBout/HD** has not been successful. The situation gets even worse if the more realistic power simulation approach **unitdelay/equal/tr** has been chosen. In this case, additionally a randomization effect in time occurs for the attacked result because early propagation is also considered. All in all, these results confirm the statement that it is much harder to attack the outputs of combinational circuit blocks than the outputs of sequential cells [MPO05]. At the outputs of sequential cells like flip-flops, no glitches and effectively no randomization in time occur. In order to get a DPA peak in case of unitdelay simulation of combinational logic, even changing to attacks on the **combout**-part of the CMOS demo circuit has not improved the situation. Including only the combout-part, e.g. in a **combout/unitdelay/equal/cc/SBout/HD** attack, significantly reduces the amount of noise.

A detailed analysis of the simulated power traces for **comb/unitdelay/equal/cc** revealed that the combinational AES-SubBytes circuit shows a severe ZV effect. If the input to the SubBytes block is 0, i.e. if the data in-

put to the comb-part is 165 (the key value), the average number of transitions in the comb-part has been 340.2. For the other cases, the average number of transitions has been 705.07. This difference can be exploited in the attack **comb/unitdelay/equal/cc/SBin/ZV** and has led to a distinct DPA peak of 0.172 for the correct key. The simulated power traces for **comb/unitdelay/equal/tr** (with included timing information) are shown in Figure 3.4. The black curve denotes the average power consumption in case the input of the SubBytes block is 0 in the current clock cycle. The figure shows that the ZV effect also occurs in the time dimension. This is the reason for the very broad DPA peak in the related attack **comb/unitdelay/equal/tr/SBin/ZV**, which is shown in Figure 3.5.

Attacks on the combinational part with random weighting: As mentioned before, a DPA attack using the HW power model does not work if the underlying power simulation only considers $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions and if these transitions are weighted with 1. However, this has been the case in all our **equal**-simulations. As a dramatic example, we look at the results of the attacks **combout/zerodelay/equal/cc/SBout/HD** and **.../SBout/HW**. In the first case, we have got a perfect DPA peak of 1.0 because of no glitches and no early propagation in the power simulation and no noise due to parallel signal activity in the DPA attack. On the other hand, the second attack has shown no correlation. In fact, in our setting we have got a perfect line of zero correlation. In order to get a more realistic situation, we have also used **random** weights (values from the interval $[1.5, 2.5]$) for each transition direction and signal. Note that this alone might not be enough because the requirement for successful DPA attacks based on the HW power model is the following: the mean value of the transition weights of the attacked signals for transition direction $0 \rightarrow 1$ must be significantly different from the mean value for the opposite transition direction. If this requirement is not fulfilled when attacking a byte result, an attacker typically switches to the HWbit power model.

With transition means of 2.127 ($0 \rightarrow 1$) and 2.017 ($1 \rightarrow 0$), the attack result **combout/zerodelay/random/cc/SBout/HW** shows a distinct DPA peak of 0.0266 for the correct key. The attack result **combout/zerodelay/random/cc/SBout/HD** still shows a peak of 0.9936 for the correct key, which is not a significant change to the result for equal weighting.

Attacks on the sequential part of the demo circuit: In a final analysis step, the sequ-part of the CMOS demo circuit has been simulated. The attack result **sequ/zerodelay/equal/cc/SBout/HD** shows, as the identical attack on the comb-part, that CMOS circuits already have a DPA leakage in this setting (no glitches, no early propagation). The correlation peak of 0.706 for the correct key is significantly higher than in the comb-attack because of less noise. A similar DPA peak occurs both in the clock cycle when the attacked result is produced and enters the flip-flops (because of the zerodelay simulation; see Figure 3.6) and in the next clock cycle when the attacked result appears at the flip-flop outputs.

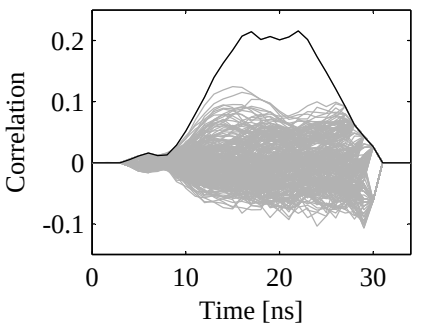


Figure 3.5: DPA attack result **comb/unitdelay/equal/tr/SBin/ZV**. The black curve shows the correlation trace for key guess = 165.

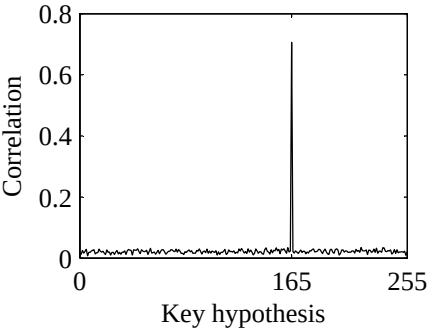


Figure 3.6: DPA attack result **sequ/zerodelay/equal/cc/SBout/HD** in the clock cycle where the attacked result is produced by the preceding combinational circuit.

When switching to unitdelay-simulations, the sequ-attacks show different results than the comb-attacks. Although the DPA peaks disappear in the clock cycle where the attacked result is produced by the preceding combinational circuit, the DPA peak in the next clock cycle (attacked result appears at FF outputs) persists. With a height of 0.126 for the correct key, it is lower compared to the corresponding zerodelay-attack peak due to the noise caused by glitches at the FF inputs.

4

Masking at Cell Level - mCMOS

Countermeasures against PA attacks aim at reducing the exploitable information in the power consumption of cryptographic devices. Various countermeasures have so far been proposed for the architecture level where they are implemented either in software or in hardware. Another very important group of countermeasures against power analysis attacks are those at the cell level. A countermeasure at this level means that the logic cells in a circuit itself counteract power analysis attacks, i.e. the cells belong to a so-called *PA-resistant logic style*. The first approaches to implement such logic styles were all based on the concept of hiding (see Section 2.4.1), both in the semiconductor industry (as far as we know) and, proposed some time later, also in the scientific world (e.g. [TAV02]). It took another while until the first PA-resistant logic styles based on masking were proposed (e.g. [ISW03]).

The main aspect why PA-resistant logic styles have come more and more into the focus of countermeasure developers is that such logic styles promise to make the securing of cryptographic circuits an automatic task. If there is a PA-resistant logic style that efficiently prevents PA attacks, the basic idea is to use this logic style in a standard semi-custom design flow to implement cryptographic circuits. A PA-resistant logic style treats the problem right where it arises—at the logic cell—and software and hardware designers of cryptographic devices do not need to be designated experts in the field of PA attacks to implement appropriate countermeasures. The automatic use of PA-resistant logic styles is possible because they are independent of the functionality and architecture of the actual cryptographic algorithm they implement. The automatic nature of the countermeasure application furthermore helps to reduce errors that would cause a power leakage to a minimum.

However, it soon turned out that PA-resistant logic styles based on hiding

usually place a hard to fulfill constraint on the semi-custom design flow. Such logic styles usually try to make the power consumption of each logic cell constant for all input and output values by using the dual-rail precharge (DRP) principle. Additionally, the parasitic loads of dual-rail wire pairs need to be balanced in order to achieve such a constant power consumption. Unfortunately, this "balancing of wires" is a rather unusual task in a semi-custom design flow (see Section 5.2.2) and can in practice never be handled perfectly [TV04c]. Furthermore, it is very hard to test and verify that the balancing achieved during the design phase is still present in each chip after production. Imbalances are commonly introduced by "natural" process variations. In some cases, imbalances are also caused by faults that are introduced passively (or even actively by very sophisticated attackers) in the mostly redundant balancing circuitry [KKT06a]. Traditional chip testing methods fail in such a case because these imbalances and faults do not manifest as logical errors.

This is the point where masked logic styles come into play, because it was assumed that they can circumvent this problem while not putting other tough constraints on the design. The basic working principle of masked logic styles—randomizing the values in a design with the help of a random mask—does not directly rely on the balancing of dual-rail wire pairs. Furthermore, such logic styles do not consume always the maximum amount of power as hiding logic styles with a constant (and thus maximum) power consumption do. Unfortunately, in the course of research it turned out more and more that the PA resistance of masked logic styles depends on various side-effects (e.g. signal glitches), which are in the end at least as hard to tackle as the balancing problem.

In this chapter, we present the results of our research when we started to work on masked logic styles. When working on a proposal for a masked logic style called Masked CMOS (mCMOS), we realized that signal glitches significantly reduce the PA resistance of such logic styles [MPG05]. The chapter starts with a discussion of the general concept of masking at the cell level.

4.1 General Concept of Cell-Level Masking

In a masked implementation of a cryptographic algorithm, each intermediate value v is represented by the two shares v_m and m according to the following equation: $v_m = v * m$. It is also possible to have three or more shares. In this case, we no longer talk about "masking" but about "secret sharing" [CJRR99, GP99].

The value m in the equation above is called the "mask" and is a random value, which is changed for every execution of the algorithm. The mask is generated inside a cryptographic device and must not be known by an attacker. For the operation $*$ several functions are possible, e.g. an arithmetic function like "modular addition $+$" (arithmetic masking) or the Boolean function "exclusive-or $\oplus$" (Boolean masking). The chosen function is typically related to the operations used in the cryptographic algorithm.

For the different intermediate values in an algorithm, usually several masks are used. In general, more masks increase the security but also decrease the

performance. Thus, a reasonable trade-off has to be found. Additionally, this trade-off is influenced by the fact that the rate at which new random values for the masks can be provided is limited. How inputs, intermediate values, and outputs of a cryptographic algorithm are masked is specified in a *masking scheme*.

Applying masking to the cell level means that the logic cells in a circuit only work on masked values and the corresponding masks. Cells that are used in such circuits are called *masked cells* and belong to a specific *masked logic style*. The circuits themselves are called *masked circuits*. Usually, Boolean masking is used for masked circuits. If the masked cells are not activated in a data- or operation-dependent manner, masked logic styles counteract both SPA and DPA attacks.

Figure 4.1 shows a 2-input unmasked cell and a corresponding 2-input masked cell. The input signals a, b and the output signal q of the unmasked cell are carried on single wires. In case of the masked cell, the input signals and the output signal are split into masked values and the corresponding masks. The following equations hold for the masked cell: $a_m = a \oplus m_a$, $b_m = b \oplus m_b$, and $q_m = q \oplus m_q$. The logic signals of the unmasked cell occur randomized in the masked cell.

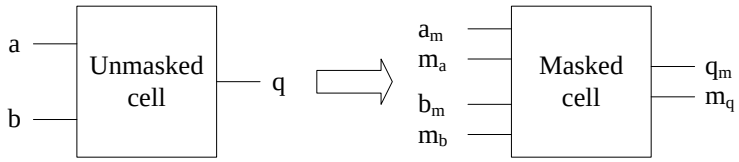


Figure 4.1: A 2-input unmasked cell and a corresponding 2-input masked cell.

The theory of masked circuits is the following. Since the masked values are independent of the unmasked values, the power consumption of the masked cells is theoretically independent of the unmasked values. As a result, the total power consumption of a cryptographic device is also independent of the processed data and the performed operations. However, like in case of all other countermeasures, complete independence cannot be achieved in practice. It is only possible to make the power consumption largely independent of the corresponding unmasked values.

In the following, we look at the power consumption of a node in a masked circuit in detail and perform a first-order DPA attack as described for a node in a CMOS circuit in Section 2.2. Table 4.1 shows all transitions that can occur at the node storing the masked value v_m when the time moves from clock cycle $t-1$ to t and the random mask m is updated in every clock cycle. The following equations hold: $v_m(t-1) = v(t-1) \oplus m(t-1)$ and $v_m(t) = v(t) \oplus m(t)$. The table furthermore shows the required energy and the probability of each transition. The value E_{01} for example denotes the energy consumption when the value at the node switches from 0 to 1, i.e. $v_m(t-1) = 0$ and $v_m(t) = 1$. The value p_{01} for example denotes the probability of the unmasked value having a transition

from 0 to 1, i.e. $v(t-1) = 0$ and $v(t) = 1$. The probability in line 1 is for example calculated as follows: $\frac{1}{4}p_{00} = p(m(t-1) = 0) \cdot p(m(t) = 0) \cdot p(v(t-1) = 0) \cdot p(v(t) = 0) = \frac{1}{2} \cdot \frac{1}{2} \cdot p_{00}$.

Table 4.1: Possible transitions of the masked value v_m on a node and corresponding energy consumptions when the mask m is updated in every clock cycle.

Line	$v(t-1)$	$m(t-1)$	$v_m(t-1)$	$v(t)$	$m(t)$	$v_m(t)$	Energy	Prob.
1	0	0	0	0	0	0	E_{00}	$\frac{1}{4}p_{00}$
2	0	0	0	1	1	0	E_{00}	$\frac{1}{4}p_{01}$
3	1	1	0	0	0	0	E_{00}	$\frac{1}{4}p_{10}$
4	1	1	0	1	1	0	E_{00}	$\frac{1}{4}p_{11}$
5	0	0	0	0	1	1	E_{01}	$\frac{1}{4}p_{00}$
6	0	0	0	1	0	1	E_{01}	$\frac{1}{4}p_{01}$
7	1	1	0	0	1	1	E_{01}	$\frac{1}{4}p_{10}$
8	1	1	0	1	0	1	E_{01}	$\frac{1}{4}p_{11}$
9	0	1	1	0	0	0	E_{10}	$\frac{1}{4}p_{00}$
10	0	1	1	1	1	0	E_{10}	$\frac{1}{4}p_{01}$
11	1	0	1	0	0	0	E_{10}	$\frac{1}{4}p_{10}$
12	1	0	1	1	1	0	E_{10}	$\frac{1}{4}p_{11}$
13	0	1	1	0	1	1	E_{11}	$\frac{1}{4}p_{00}$
14	0	1	1	1	0	1	E_{11}	$\frac{1}{4}p_{01}$
15	1	0	1	0	1	1	E_{11}	$\frac{1}{4}p_{10}$
16	1	0	1	1	0	1	E_{11}	$\frac{1}{4}p_{11}$

When a DPA attack is performed on the masked circuit, the power traces are for example split according to the value of $v(t)$ in the clock cycle t. However, in the circuit now the masked value $v_m(t)$ is actually processed. In order to calculate the expected value of the mean of the traces where $v(t) = 0$, $\mathcal{E}(M_{v(t)=0})$, the odd lines in the table are used while $\mathcal{E}(M_{v(t)=1})$ is calculated based on the even lines. As shown in Equation 4.1, the two expected values are equal. Therefore, the expected value of the difference of the means $\mathcal{E}(DM_{v(t)})$ is zero and first-order DPA attacks are not possible any more for the given power model. Note that the power consumption of the other nodes in the masked circuit during clock cycle t can be typically modeled as Gaussian noise (see for example [MDS02]). Thus, it does not influence the expected values of the means and consequently the difference of the means.

$$\mathcal{E}(M_{v(t)=0}) = \mathcal{E}(M_{v(t)=1}) = \frac{1}{4}\left(E_{00} + E_{01} + E_{10} + E_{11}\right) \tag{4.1}$$

Two basic design decisions of masked logic styles are how many different masks are used in a circuit and how frequently each of these masks is updated. These to basic aspects are discussed in detail in the following sections.

4.1.1 The Number of Different Masks in a Circuit

There exist three approaches to build masked circuits. First, *one distinct mask for each signal* can be used. As a result, all masked values are pairwise independent of each other, no matter whether the corresponding unmasked values are independent or not. Using this approach, the functionality of the logic cells is typically very complex, because every input and output signal is masked separately. The number of necessary masks for such a circuit is huge. Therefore, this approach is not practical.

The second possibility is to partition the signals of a circuit in several groups and to use *the same mask for a group of signals*. This decreases the number of required masks significantly. Furthermore, the complexity of cells working on input signals that are masked with the same mask is typically reduced. For every masked signal that passes over from one signal group $G1$ to another signal group $G2$, some additional circuitry is necessary to change the mask from m_{G1} to m_{G2}. Defining the number of signal groups with different masks is a non-trivial task.

The third approach is to use *the same mask for all signals* in a circuit. Hence, there is no overhead for handling different masks. If unmasked values depend on each other, also the corresponding masked values depend on each other. A drawback of this approach is that updating the single mask value can typically be detected via the power consumption of the huge mask net and possible re-masking activities. The mask net is responsible for distributing the mask to every cell in the circuit. Possible solutions to this problem are to resort to the second approach discussed above or to implement the mask net in a DRP-like and balanced manner.

4.1.2 Mask Value Updating Frequency

When specifying how often the mask values in a circuit are updated, the following considerations are important. If the *mask values are updated in each clock cycle*, the rate at which new random values must be generated is very high. This is especially true if many different masks are used in a circuit. A main advantage of updating the mask values in each clock cycle is that higher-order DPA attacks (see Chapter 2) are more difficult. Thus, this approach is the most common one for masked logic styles.

In order to reduce the rate at which new random values are required, the *mask values can be used in several clock cycles*. As a result, the rate at which new random values must be generated is reduced. However, higher-order DPA attacks become more effective. Therefore, this approach is usually avoided.

4.2 Masked CMOS (mCMOS)

In many patents [MDP01, KKG02, MP03] and papers [ISW03, Tri03, GM04, TKL05] that first proposed masking at the cell level as an efficient countermeasure, the security of the masking approach is proven similar to the argumentation

presented in Section 4.1. However, this proof is based on the substantial assumption that the output of a logic cell switches only once per clock cycle. In general, this assumption does not hold true in practice. Most of today's digital circuits are implemented in CMOS logic. The signal delays in CMOS circuits cause the input signals of the CMOS cells to arrive at different times. Since the outputs of combinational CMOS cells directly react on input signal changes, these outputs potentially switch several times during a clock cycle. These additional switching events are called *signal glitches* and significantly impact the power consumption of a CMOS circuit (see Section 2.3.4 for further details). Thus, glitches play also an important role in the context of the PA resistance of cryptographic circuits.

At the time the masked logic styles listed above have been proposed, we also worked on a masked logic style called Masked CMOS (mCMOS). In this section, mCMOS and its properties will be presented in detail. **Note that this presentation does not take the findings about the impact of glitches on the PA resistance of masked circuits into account! mCMOS was developed shortly before this issue was discovered.** During the work on mCMOS, we realized at some point that glitches in general reduce the PA resistance of masked circuits. These findings have been published in [MPG05] and will be discussed in detail both from a theoretical and also a practical standpoint in the subsequent two sections. Because of the occurrence of the "glitch problem" for masked circuits, we never published the mCMOS logic style proposal.

mCMOS cells are single rail (i.e. a logic value is encoded by two distinct voltage levels on a single wire) and use the same Boolean mask m for the inputs and outputs of a cell. This leads to very simple functions that must be realized by the basic mCMOS cells. Table 4.2 shows the truth tables of the masked versions of the basic combinational cells AND, OR, and XOR with two inputs. a and b denote the unmasked input values. The masked input values are calculated as follows: $a_m = a \oplus m$, $b_m = b \oplus m$. The combinational mCMOS cell takes these masked values as input and calculates the respective output value $q_m = q \oplus m$ according to the intended cell function f so that the following equation is fulfilled: $q = f(a, b)$.

From the truth tables in Table 4.2, the Boolean functions of the basic combinational mCMOS cells can be worked out. Table 4.3 summarizes them together with the NOT function. A few very nice and important conclusions can be drawn from Table 4.3. The first finding is that an mCMOS inverter is just a CMOS inverter that takes the masked value a_m as input, i.e. the inversion function is invariant under the Boolean masking operation. Next, the 2-input masked AND function is nothing else than the so-called *majority (MAJ)* function with three inputs. A 3-input majority function provides the value 1 at its output if more inputs are 1 than 0, i.e. two or three inputs are 1. The masked NAND function is then simply the *inverting majority (IMAJ)* function.

A very important aspect is that MAJ and IMAJ cells are typically available in standard-cell libraries. This means that mCMOS circuits can be com-

Table 4.2: Truth tables of the masked versions of the basic combinational cells.

					AND		OR		XOR	
a	b	m	a_m	b_m	q	q_m	q	q_m	q	q_m
0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	1	1	1	1
1	0	0	1	0	0	0	1	1	1	1
1	1	0	1	1	1	1	1	1	0	0
0	0	1	1	1	0	1	0	1	0	1
0	1	1	1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1	0	1	0
1	1	1	0	0	1	0	1	0	0	1

pletely built based on the standard cells in common libraries. This is a very big advantage since it avoids the usually huge effort of designing special masked cells and makes a switch of the underlying CMOS logic style or the process technology rather easy. Figure 4.2 shows the transistor schematic of a 3-input IMAJ cell ($q = IMAJ(a, b, c)$), which directly realizes an mCMOS NAND cell ($q_m = IMAJ(a_m, b_m, m)$). Note that the only "full-grown" internal node in the IMAJ cell also constitutes the output node of the cell. Thus, this internal node is well masked. It is assumed that the other four, "non-full-grown" internal nodes (i.e. nodes which are not for all input values either connected to V_{DD} or GND) in the IMAJ cell do not produce a significant power leakage that is measurable from the outside of an integrated circuit (IC).

Table 4.3: Basic combinational cell functions and their masked realizations.

Function	Masked realization
NOT	$\overline{a} \oplus m = \overline{a_m}$
AND	$(a \wedge b) \oplus m = (a_m \wedge b_m) \vee (a_m \wedge m) \vee (b_m \wedge m)$
NAND	$\overline{(a \wedge b)} \oplus m = (\overline{a_m} \wedge \overline{b_m}) \vee (\overline{a_m} \wedge \overline{m}) \vee (\overline{b_m} \wedge \overline{m})$
OR	$(a \vee b) \oplus m = (a_m \wedge b_m) \vee (a_m \wedge \overline{m}) \vee (b_m \wedge \overline{m})$
NOR	$\overline{(a \vee b)} \oplus m = (\overline{a_m} \wedge \overline{b_m}) \vee (\overline{a_m} \wedge m) \vee (\overline{b_m} \wedge m)$
XOR	$(a \oplus b) \oplus m = a_m \oplus b_m \oplus m$
XNOR	$\overline{(a \oplus b)} \oplus m = a_m \oplus b_m \oplus m \oplus 1$

Another finding from Table 4.3 is that the masked OR (NOR) function can be realized by using the MAJ (IMAJ) function and inverting the mask input. Furthermore, the 2-input masked XOR (XNOR) function can be implemented with a 3-input XOR (XNOR) function. However, such a 3-input XOR (XNOR) cell cannot be used directly in most cases because it is typically implemented in a way that the correct masking of full-grown internal nodes of the cell would not be given. This happens for example if the masked values a_m and b_m are fed internally into an asymmetric function like AND, which leads to an output value that is not completely uncorrelated to the unmasked values. Therefore,

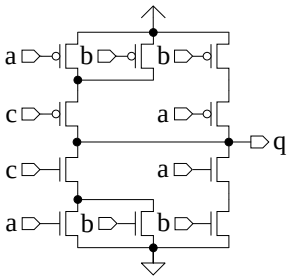


Figure 4.2: Transistor schematic of a CMOS 3-input inverting-majority cell.

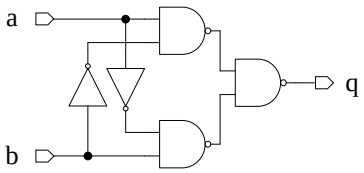


Figure 4.3: Implementing an XOR function with inverters and NANDs.

it is suggested to implement the XOR function based on its realization with inverters and NANDs as shown in Figure 4.3. To get a masked version of the implementation, the three NANDs must be replaced by IMAJ cells. In case of the XNOR function, the NAND at the output must be replaced by a MAJ cell. Table 4.4 lists the basic mCMOS cells and their implementation with CMOS cells. Also the mCMOS flip-flop implementation is included, which is explained in the next paragraphs.

Table 4.4: Basic mCMOS cells and their implementation with CMOS cells.

mCMOS cell	CMOS implementation
Inverter	Inverter
2-input AND (mAND)	3-input MAJ
2-input NAND (mNAND)	3-input IMAJ
2-input OR (mOR)	3-input MAJ + inverter
2-input NOR (mNOR)	3-input IMAJ + inverter
2-input XOR (mXOR)	3×(3-input IMAJ) + 2×inverter
2-input XNOR (mXNOR)	2×(3-input IMAJ) + 3-input MAJ + 2×inv.
Flip-flop (mFF)	2-input XOR + FF

Figure 4.4 shows the general architecture of an mCMOS circuit. The mask bit, which is updated in every clock cycle, is provided by a pseudo-random number generator (PRNG) that is initially seeded by a true-random number generator (TRNG). The PRNG produces every clock cycle a mask bit called m_n, which is the mask bit that will be used in the next clock cycle. The mask bit that was produced one cycle before is the mask bit m of the current clock cycle. The PRNG does not output m_n directly, but in the form $m \oplus m_n$.

The situation in a specific clock cycle is the following. All circuit inputs get masked with the current mask bit m and are fed into the mCMOS combinational block, which is also provided with the mask m. The data stored in the mCMOS flip-flops (mFFs) is masked with m as well and is likewise fed back into the mCMOS combinational block. If a masked value is required as an output value,

the current mask m must be removed. It is clear that only uncritical signals may leaf the masked circuit. The mCMOS combinational block calculates the intended masked result, which enters the mFFs. The mFFs not only store data, they also need to update the mask for the next clock cycle. This is achieved by xor'ing a data value $d_m = d \oplus m$ that enters an mFF with the value $m \oplus m_n$: $d \oplus m \oplus m \oplus m_n = d \oplus m_n = d_{m_n}$. The newly masked data value d_{m_n} is stored in the CMOS flip-flop at the next active clock edge, which also marks the beginning of a new clock cycle. At the beginning of the new clock cycle, m_n becomes m and the PRNG generates a new m_n. This finishes a cycle and the process starts again.

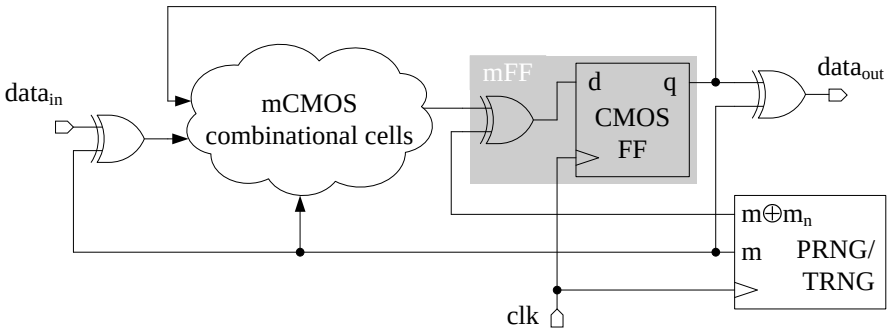


Figure 4.4: Architecture of an mCMOS circuit.

A main problem of any single-rail masked circuit which uses a single mask bit is that it can be usually detected via the power consumption of the circuit whether the mask bit changes between two clock cycles or not. A changing mask bit causes a significantly higher power consumption and is thus attackable via SPA. There are three main reasons for that. Firstly, the typically big mask net requires a lot of power when the mask bit changes. Secondly, the mask bit arrives generally at a different time at the inputs of cells. Thus, if the mask value changes, it is more likely that glitches occur in the circuit than in case the mask value stays the same. Thirdly, the unmasked values of many signals in a circuit are not random in each clock cycle. In other words, the unmasked values of a signal in consecutive clock cycles may tend to stay the same. The reason for that lies in the particular Boolean function of the circuit. The extreme case is constant signal values over two or more clock cycles. For example, control signals or flip-flop (register) contents often behave in such a way. If the mask bit changes, the masked values of such more or less constant signals will also change, which increases the power consumption. In the papers [TS07, ST07], this effect is studied in detail for more recent masked logic styles. Also the logic-level power simulations of an mCMOS demo circuit presented in Section 4.4.2 show that significantly more transitions occur if the mask bit changes.

As general solution to this problem, it is proposed to divide the whole circuit that is going to be implemented in mCMOS into various *masking domains*, which all have their own mask bit. However, this approach increases the area

requirements of mCMOS circuits because interfacing circuitry is necessary for signals that pass between masking domains. Furthermore, it is not trivial to decide how many mask domains are necessary for a specific circuit and how a circuit should be divided. And the PRNG has to provide more new mask bits in each clock cycle. Other methods that might help to solve this SPA problem are to build the mask net in a DRP-like manner to basically balance its power consumption and to avoid implementing "constant" signals in mCMOS (e.g. by excluding the state machine of a design, because the control logic is a typical source of such "constant" signals and should in most cases not include PA-critical signals).

Even when taking the effort for protecting the mask bit transitions from SPA attacks into account, it is assumed that mCMOS circuits are still much easier to implement in a semi-custom design flow than PA-resistant circuits based on logic styles with a constant power consumption. There is no need to balance complementary wires or a similar requirement. Furthermore, the protection of mask bit transitions must only be done against SPA attacks. Semi-custom circuit design in the context of masked logic styles is discussed in detail in Section 5.2.

It is also assumed that mCMOS circuits are secure against higher-order DPA attacks (see Section 2.2). Such attacks are based on the assumption that an attacker can separately measure the power consumptions when processing several related masked values (i.e. masked with the same mask) or several masked values and their respective masks. For example, in a second-order attack, an attacker might measure the power consumption when processing m and the power consumption when processing a_m. However, in an mCMOS circuit these two values are processed together in one clock cycle. The time interval between the two processing events is usually somewhere in the nanosecond range. Due to the low-pass filtering behavior of the power net, it seems to be impossible to separately measure the power consumption of the two processing events.

An interesting feature of mCMOS circuits is the possibility to switch easily between secure (masked) and insecure (unmasked) mode. This is simply achieved by keeping the mask constant. In the insecure mode, the mCMOS circuit should behave basically like the corresponding CMOS circuit and should have a significantly lower power consumption than in secure mode.

4.2.1 Comparison of CMOS, mCMOS, and SABL

In this section, the basic properties of the logic styles CMOS, mCMOS, and SABL are compared. SABL is a hiding logic style based on the DRP principle and is implemented as full-custom cells (see Section 2.4.2 for more details). The compared properties include area requirements, circuit speed, power consumption, and DPA resistance. All technology-specific comparisons are based on the 0.35 μm process technology from austriamicrosystems [aus] and the related standard-cell library C35. Note that the comparisons are based on rather rudimentary figures because the work on mCMOS was stopped immediately after the glitch problem of masked logic has been discovered. Thus, for example no figures of a large test circuit in mCMOS are available for the comparison.

Area Requirements

For a technology-independent comparison of the area requirements of the basic CMOS, mCMOS and SABL cells, the transistor count is considered. The figures in Table 4.5 are based on common implementations of the cells using a typical number of transistors. For mCMOS, the transistor count is derived from CMOS and the equivalences shown in Table 4.4. The figures for SABL are derived from schematics presented in [TAV02] and [Sch03].

Table 4.5 indicates that the area requirements of mCMOS are located somewhere between CMOS and SABL—with a general orientation towards SABL. However, the size of a cell layout is not only determined by the transistor count but also by the intra-cell routing complexity. Practical implementations of layouts of SABL cells [Pop04] and of mCMOS cells (based on their CMOS implementations according to Table 4.4) in a 0.35 μm process technology clearly show an advantage for mCMOS cells. The work in [Pop04] states an area increase factor of 3.5 for an SABL circuit compared to the corresponding CMOS circuit. Hence, it is assumed that the average increase factor for mCMOS circuits is somewhat below this value.

What specifically increases the area requirements of mCMOS circuits is the partitioning into multiple masking domains in order to halt SPA attacks on the mask bit. Furthermore, a PRNG is required for an mCMOS circuit.

Table 4.5: Transistor counts of basic CMOS, mCMOS, and SABL cells.

Function	CMOS	mCMOS	mCMOS:CMOS	SABL	SABL:CMOS
Buffer	4	4	1	8	2
NOT	2	2	1	0	0
AND	6	12	2	18	3
NAND	4	10	2.5	18	4.5
OR	6	14	2.3	18	3
NOR	4	12	3	18	4.5
XOR	10	34	3.4	18	1.8
XNOR	10	36	3.6	18	1.8
FF	26	36	1.4	28	1.1

Circuit Speed

For the figures related to circuit speed, only rough estimations have been made for mCMOS circuits. These figures are based on a comparison of the average propagation delays of a CMOS NAND cell and an mCMOS NAND cell (3-input IMAJ cell) in a 0.35 μm process technology. The NAND cell is the most representative cell for both logic styles. The propagation delays of the 3-input IMAJ cell for different input-output paths are all approximately 1-2 times bigger than the related propagation delays of the 2-input NAND cell of the same driving strength. Thus, this factor is also taken as a first estimate for the speed reduction of an mCMOS circuit compared to the corresponding CMOS circuit.

The work in [Pop04] shows a decrease of the circuit speed of an SABL circuit by a factor of around 2. This reduction is mainly caused by the additional precharge phase of the SABL circuit. An mCMOS circuit does not have such a precharge phase. In summary, it is estimated that the mCMOS circuit speed is higher than the circuit speed of the corresponding SABL circuit.

Power Consumption

A severe disadvantage of SABL is its high power consumption. In a clock cycle, every SABL cell dissipates the same, worst-case amount of power in order to achieve a constant power consumption in every clock cycle. Furthermore, all combinational SABL cells are also (additionally to the sequential SABL cells) connected to the clock tree, which considerably increases its load. Since the clock signal switches unconditionally two times per clock cycle, this significantly contributes to the increased power consumption of SABL circuits.

In a CMOS circuit much less transitions occur in a clock cycle compared to the corresponding SABL circuit even when glitches are considered (note that no glitches occur in SABL circuits). Therefore, the power consumption of CMOS circuits is significantly lower than that of SABL. Simulations of SABL circuits have shown an overall increase of the power consumption by a factor of around 4 [TV03, Pop04] compared to CMOS circuits.

In mCMOS circuits, more transitions will occur compared to the corresponding CMOS circuits. One of the reasons is that for unmasked signals which are (mostly) constant, transitions will occur if they are masked and the mask bit changes. For (mostly) random signals, masking does not have a significant effect on the number of transitions because the amounts of newly created signal transitions and avoided signal transitions due to a mask bit change will approximately compensate each other. A second reason for the signal transition increase in mCMOS circuits is that glitches are more likely due to the additional mask bit input of the logic cells. However, the power consumption of mCMOS circuits still lies significantly below that of the corresponding SABL circuits because the mCMOS cells do not consume the maximum amount of power in every clock cycle. And the increase of the power consumption due to the mask net in mCMOS circuits is usually excelled by the increased power consumption of the extended clock tree in SABL circuits. Therefore, the power consumption increase of mCMOS compared to CMOS is well below the factor 4.

DPA Resistance

In order to compare the DPA resistance of CMOS, SABL, and mCMOS, a NAND cell implemented in all three logic styles has been attacked. According to the output value of the NAND cell, the simulated power traces were separated into two groups and the mean power trace of each group was calculated. Finally, the energies of these two mean traces were determined and a difference calculated. The result, called the *difference of mean energies (DME)*, indicates the height of the peak in a DPA attack. The assumption made in this case is that an attacker

can only measure the total energy consumed by a cell during a clock cycle. It seams reasonable that due to the limited bandwidth of a chip's or smart card's power supply net and the measurement setup, the instantaneous course of the power consumption during a clock cycle cannot be determined.

The power simulations have been performed with the analog-level simulator *Spectre* from Cadence Design Systems [Cad]. In the simulations, the output loads of the NAND cells have been varied. In case of the single-rail logic styles CMOS and mCMOS, the output load has been modeled by a variable capacitance connected to the output nodes of the NAND cells. For the SABL NAND cell, two variable capacitances have been connected to the dual-rail output of the cell. In order to analyze the effect of mismatch on the DPA resistance of SABL, the power simulation has also been performed with one output capacitance having a 5 % and a 10 % lower value than the nominal one. The dependency of the DME on the load capacitance of the CMOS, mCMOS, and SABL NAND cells is shown in Figure 4.5.

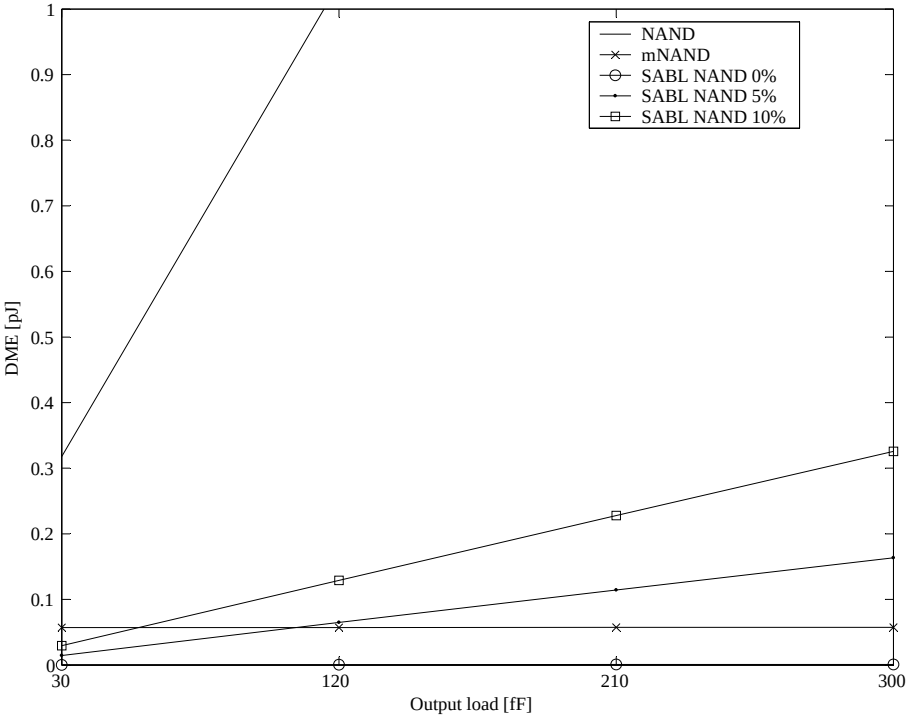


Figure 4.5: Difference of mean energies (DME) for a DPA attack on a NAND cell implemented in CMOS, mCMOS, and SABL depending on the (nominal) cell output load. In the case of SABL, the imbalance of one complementary output node with respect to the nominal value is given in percent.

Figure 4.5 clearly shows that as long as the balancing of complementary wires in an SABL circuit is ideal, its DPA resistance is outstanding (see line "SABL

NAND 0 %"). But ideal balancing is impossible to achieve on a real chip due to layout parasitics, process variations etc. From the two curves in Figure 4.5, where an unbalanced differential output of the attacked SABL NAND cell is assumed (see lines "SABL NAND 5 %" and "SABL NAND 10 %"), it can be seen that already an absolute difference of 5.5 *fF* leads to the same amount of side-channel information leakage as for an mCMOS NAND cell. The 5.5 *fF* have been determined by looking at the crossing points of the line for the mNAND cell and one of the lines for the SABL NAND cell with unbalanced complementary outputs: with 5 % mismatch, the two lines cross at around 110 *fF* nominal output load while with 10 % mismatch, the two lines cross at around 55 *fF* mismatch of the complementary output capacitances.

Achieving only a capacitance difference of 5.5 *fF* for complementary wire pairs is assumed to be rather tough, especially if a semi-custom design flow is used for layout generation. Therefore, the DPA resistance of mCMOS logic seems to be higher than that of SABL in a real environment. mCMOS also suites better for a semi-custom design methodology because the routing process is not constraint by a troublesome balancing requirement.

The DPA attack as described above has also been performed on other basic cells. The DME of SABL and mCMOS cells have been the same for the following capacitive mismatches of the complementary output nodes: 3 *fF* (NOR), 2 *fF* (XNOR), and 1.5 *fF* (XOR).

4.3 The Glitch Problem of Masked Logic Styles in Theory

The security proofs of the first proposed masked logic styles are all based on a similar argumentation as presented in Section 4.1. This is in particular true for the masked logic style mCMOS, which has been introduced in the last section. However, this argumentation does not take the effects of signal glitches into account, which have a significant impact on the power consumption of a circuit. It was first shown in [SSI04, MPG05] and then later in [SSI05, SSI07a] that glitches are a source for power leakage in masked circuits.

The reasons for glitches are that the output values of combinational logic cells depend directly on the input values and that the new input values of a clock cycle usually arrive at different points in time. Each time a new input value arrives, the cell output possibly switches to a temporary, "wrong" value. Only if all input values have reached their new values, the final, correct output value of the current clock cycle can be calculated by a combinational cell. These switching events to wrong output values are called glitches and contribute significantly to the overall power consumption of a circuit. Note that glitches at the output of a cell are typically provided to the inputs of subsequent combinational cells and cause even more glitches in these cells. Sequential cells like flip-flops do not produce glitches and also stop them if they occur at the input.

Whether or not the outputs of combinational cells directly depend on the

inputs is defined by the functionality of the used logic style. In the by far most common logic style today, CMOS, the combinational cells show such a direct behavior. This is also true for mCMOS, which is built based on CMOS cells. The different arrival times of the cell input values are caused by the varying propagation delays of cells and wires and the fact that logic signals usually must take paths of different logic depth (i.e. different numbers of logic cells) until they arrive at the particular inputs.

In order to theoretically show the effect of glitches on the PA resistance of a masked logic style, the analysis model must be extended. This extended model considers the sequence of input value transitions as well as the resulting energy consumption of cells due to output value transitions including glitches. A possible scenario for a 2-input mCMOS AND cell is for example: the mask bit m arrives first while the masked input values a_m and b_m arrive later at the same time. It is in fact quite a likely case that m arrives first because it is produced by a separate part of the circuit (the PRNG) and does not need to propagate through many levels of combinational logic. Common assumptions to simplify the analysis are that the unmasked input values of a cell are independent and uniformly distributed and that cell inputs switch only once. Furthermore, it is typically assumed that—in the context of the given example—the delay between m and the other input values is big enough so that the output value of the mAND cell switches fully to the temporary values in case of glitches. Each of the three input values of the mAND cell can have 4 possible transitions ($0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0$, $1 \rightarrow 1$. This leads to $4^3 = 64$ possible combinations of input value transitions. Together with the chosen delay scenario of the inputs, this results in a specific energy consumption of the cell for each of these combinations. The energy values are finally used in a DPA attack as described in Section 4.1. A difference of means (DME) value that is unequal to 0 indicates a power leakage.

In the appendix of [Man04b] the analysis of an mAND cell is presented in detail. In [MPG05], the more general case where each input and output signal has its own mask bit is thoroughly discussed. Furthermore, also cells implementing other basic Boolean functions are investigated. In both analyses, various delay scenarios of the input values have been used. The results clearly show that in all circumstances, a power leakage occurs for the non-linear cells (AND, NAND, OR, NOR) if glitches are considered. Interestingly, linear cells like XOR and XNOR are not susceptible to DPA attacks when glitches occur. However, non-linear Boolean functions are essential to implement digital circuits.

The conclusion of the theoretical analysis is that glitches must be avoided in masked circuits to be secure against first-order DPA attacks. This can for example be achieved by using variants of so-called domino logic styles [RCN03]. The typical drawback of such an approach is a significant increase of area requirements and power consumption. In some cases, this also means that new cells must be implemented from scratch, which is a rather costly and inflexible approach.

4.4 The Glitch Problem of Masked Logic Styles in Practice

In the following sections, results are presented that verify the findings about glitches and masked circuits practically.

4.4.1 Verification with Analog-Level Power Simulation

In this case study, the power consumption of different implementations of 2-input masked AND cells has been simulated at the analog level with the simulator *Spectre* from Cadence Design Systems. The simulations have been performed for different delay scenarios of the input values to include the effects of glitches. The simulated power traces have then been used in a DPA attack to investigate how glitches influence the DPA results. These results have originally been published in [MPG05].

The masked cells have been implemented using the C35 CMOS standard cells from austriamicrosystems, which are based on a $0.35\,\mu m$ process technology. Two implementation approaches for masked cells have been investigated. The first one has been presented by Messerges et al. in a patent [MDP01]. They propose to implement masked cells using multiplexors and crossbar switches. This approach leads to rather big masked cells but has nevertheless been used in a proposal to secure a data scrambling technique against PA attacks [Gol03]. A 2-input masked AND cell requires 3 multiplexors, 3 crossbar switches, and 4 XOR cells. The second investigated implementation approach for masked cells is that of Trichina et al. [Tri03, TKL05]. In this case, normal CMOS cells are used to implement masked cells. When following this approach, a 2-input masked AND cell consists of 4 AND cells and 4 XOR cells. Both implementation approaches have in common that all input and output values (a_m, b_m, q_m) have their own mask bit (m_a, m_b, m_q). In the power simulations, an output load of $10\,fF$ has been set for the masked AND cells.

Both masked AND cells have been simulated for two delay scenarios. In the first scenario, all input values of a cell switch at the same point in time (i.e. at time $1\,ns$). In the second delay scenario, the mask bit m_q switches $1\,ns$ before (i.e. at time $1\,ns$) the other four input signals (i.e. at time $2\,ns$). Since glitch-free input signals are assumed, $4^5 = 1024$ combinations of input transitions are possible. The DPA attack is performed by dividing the simulated power traces in two groups according to the cell output value $q = q_m \oplus m_q$ when all input signal transitions have occurred. A value unequal to zero in the difference of the means of the two sets shows a power leakage of the masked cell. For reference, also a normal CMOS AND cell has been simulated and attacked. The inputs (a, b) of this cell have been specified to switch at the time $1\,ns$.

Figure 4.6 shows the results of the DPA attacks on the normal CMOS AND cell, on the masked AND cell according to Messerges, and on the masked AND cell according to Trichina. In the power simulations for the figures of the first and second row, the input signal transitions occur all at the same time while for

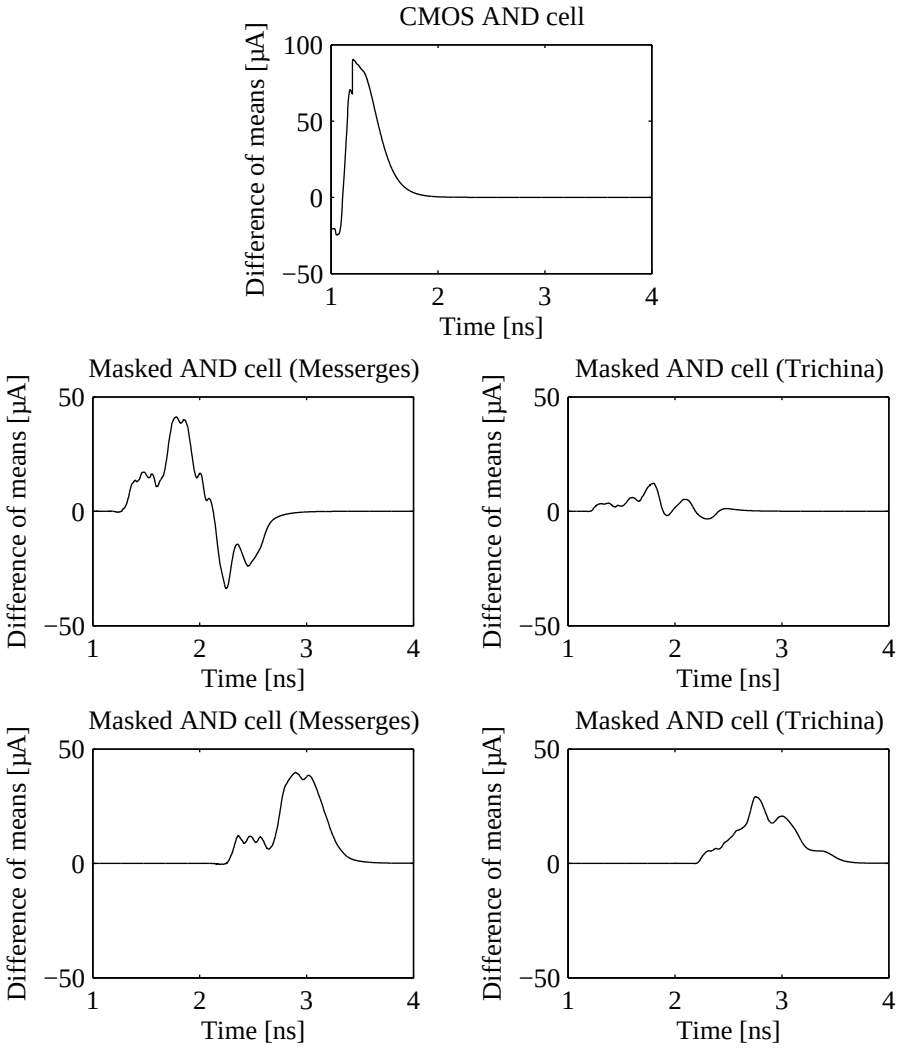


Figure 4.6: Results of the DPA attacks on the normal CMOS AND cell, on the masked AND cell according to Messerges, and on the masked AND cell according to Trichina for different delay scenarios: input signal transitions occur all at the same time (figures in first and second row) and transitions of the output mask bit m_q occur $1\,ns$ earlier than the other transitions (third row).

the figures in the third row, the transitions of the output mask bit m_q occur $1\,ns$ earlier than the other transitions. A significant increase of the power leakage can especially be observed in the latter scenario for the masked AND cell following Trichina's approach. The masked AND cell according to Messerges already shows a high power leakage if all input value transitions occur at the same time. The reason is that the complex structure of Messerges' masked AND cell already provokes internal glitches even in this delay scenario.

In the analysis, it has not been searched for the delay scenario which maximizes the power leakage of the masked AND cells. Further investigations have indicated that there are delay and attack scenarios which show a similar amount of leakage as for the CMOS AND cell. In the presented case, the power leakage of the masked AND cells is only approximately halved compared to the leakage of the normal CMOS AND cell. Nevertheless, this result shows already that masking at the cell level is quite ineffective if glitches occur in the masked circuits.

4.4.2 Verification with Logic-Level Power Simulation - Simulated DPA-Attack on the Demo Circuit "AESencinit" Implemented in mCMOS

In order to investigate the influence of glitches and other timing effects like early propagation on masked circuits, we have applied the power trace estimation technique based on logic simulation. This technique has been presented in detail in Section 3.2. The specific tool that was used to map logic simulation results to power traces is called "vcd_analyzer" and is described in detail in Section 3.3. As demo circuit, the design "AESencinit" has been used, which is described in Section 3.3.1.

The implementation of the demo circuit in mCMOS, the power simulation with vcd_analyzer, and the DPA attack have been performed similarly as described for the CMOS implementation of the demo circuit in Section 3.3.1. In that section, also the used terminology to denote the various simulation and analysis results is explained in detail. The main differences of the mCMOS and the CMOS implementation of the demo circuit are the following. After synthesis, the CMOS circuit has been translated to an mCMOS circuit. Secondly, since an mCMOS circuit also takes the mask bit as an input, $256^2 * 4$ simulations are necessary to simulate all possible combinations of input value transitions for the **comb**-part of the demo circuit (256^2 transition possibilities for the 8-bit input value; 4 transition possibilities for the mask bit). The **sequ**-part has been simulated for the same number ($256^2 * 4$) of random 8-bit input values and random 1-bit masks. Simulating all transition possibilities of the sequ-part would require too many simulations.

Attack Results

The logic simulation of the mCMOS implementation of the AESencinit demo circuit has been performed with a clock cycle time of $60\,ns$. The 8-bit secret key

used in the simulation has been 165 (0xA5).

Attacks on the combinational part of the demo circuit: The attack result **comb/zerodelay/equal/cc/SBout/HD** has shown no DPA peak for the mCMOS demo circuit because of masking and the fact that no glitches (and no early propagation) have been considered in the power simulation. For the same attack on the CMOS demo circuit, a DPA peak already appeared in this scenario. Thus, when applying such a simple power model, mCMOS completely prevents DPA attacks on a circuit. Note that the attack **.../SBin/ZV** obviously also does not show a DPA peak in the simple power model.

However, in the more realistic attack **comb/unitdelay/equal/cc/SBin/ZV**, where glitches but no early propagation has been considered in the power simulation, a DPA peak of around 0.095 has occurred for the correct key. This practically demonstrates the "glitch problem" of combinational mCMOS circuits. Like in the CMOS attack, the leakage has only been exploitable via the ZV model. In the attack **comb/unitdelay/equal/tr/SBin/ZV**, which considers also early propagation, the DPA peak for the correct key has increased to around 0.245. The result of this attack is depicted in Figure 4.7. The six false peaks after 40 ns are caused by the wrong key hypotheses 22, 74, 138, 144, 156, and 184 and originate from a minor ZV effect of the S-box for the input values $22 \oplus 165 = 179$, $74 \oplus 165 = 239$, etc.: the S-box calculation for these input values takes longer than for all other input values. However, the shape of the DPA peaks easily allows identifying the peak of the correct key. Figure 4.8 shows the corresponding simulated power traces where the major ZV effect can be easily identified.

Some general observations for single-rail masked circuits that use a single mask bit have been verified by looking at the power simulation results of the mCMOS circuit in detail. The power simulation result **comb/zerodelay/equal/cc** (no glitches, no early propagation) shows that the average toggle count for each input value depends in the following way on the occurring mask transition between two clock cycles. For mask transitions $0 \rightarrow 0$ and $1 \rightarrow 1$, the average toggle counts for each input value are identical. This is clear because only the directions of the occurring data signal transitions change in the two cases, which has no influence on the toggle count. The same is true when comparing the average toggle counts for mask transitions $0 \rightarrow 1$ and $1 \rightarrow 0$ in the given power simulation scenario.

As a matter of fact, masked signals which stay constant for the mask transitions $0 \rightarrow 0$ and $1 \rightarrow 1$ undergo a transition in case the mask transition is $0 \rightarrow 1$ or $1 \rightarrow 0$ and vice versa. A second observation is that the average toggle count over all possible input values in case the mask bit changes is significantly higher than in case the mask does not change between subsequent clock cycles: 207.2 toggles on average compared to 145.8 toggles on average. This shows that in the unmasked combinational circuit, on average more signals stay the same (approx. 3/5 of them) than change between two clock cycles. This is an inherent property of the specific circuit and its logical implementation. The difference

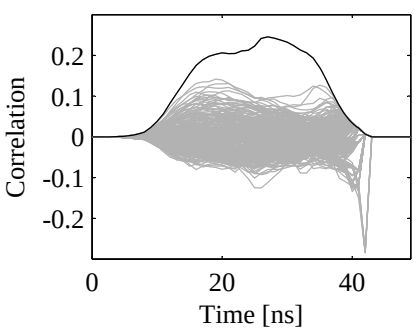


Figure 4.7: DPA attack result **comb/unitdelay/equal/tr/SBin/ZV**. The black curve shows the correlation trace for key guess = 165.

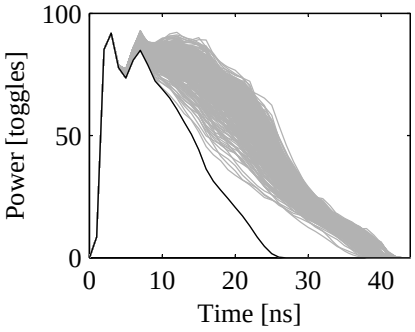


Figure 4.8: Average power traces for the simulation **comb/unitdelay/equal/tr** for different input values of the comb-part and random mask bits; black curve: input of comb-part is 165 ⇒ input of SubBytes block is 0 (unmasked).

in the number of toggles with respect to whether the mask bit changes or not is a problem because it opens up the possibility to detect mask transitions in mCMOS circuits by SPA attacks. To counteract such attacks, a separation of the masked circuit into masking domains as described in Section 4.2 has been proposed.

Another interesting observation for the power simulation result **comb/zero-delay/equal/cc** is that even in this case, a ZV effect is visible in the average toggle counts for a specific mask transition. If the input to the S-box is 0 (unmasked) the least signal transitions occur for the mask transitions $0 \rightarrow 0$ and $1 \rightarrow 1$ and the most signal transitions occur for the mask transitions $0 \rightarrow 1$ and $1 \rightarrow 0$. These effects cancel out in the actual attack where mask transitions occur randomly.

For the more realistic power simulation results **comb/unitdelay/equal/cc** (glitches, no early propagation) and **../tr** (glitches, early propagation) the situation is as follows. Like in the zero delay simulation, the average toggle counts for each input value are the same for the mask transitions $0 \rightarrow 0$ and $1 \rightarrow 1$. The same is still true for the mask transitions $0 \rightarrow 1$ and $1 \rightarrow 0$.

On the other hand, the ratio between the average toggle count in case the mask bit changes in subsequent clock cycles and the average toggle count when the mask bit stays constant rises. The reason is that additional glitches are caused by changing mask bits. Such glitches occur more often for cells located

deep in combinational paths, because there, the arrival times of masked data inputs and mask input are typically very different. This effect increases the problem of the detectability of the mask transitions via SPA.

Finally, the power simulation results based on unit delay show severe ZV effects in the average toggle counts for specific mask transitions. Furthermore, these effects do not cancel out any more in the actual attack (with random mask transitions) as it was the case in the zero delay simulation. Figure 4.8 depicts the simulated power traces, which clearly show the ZV effect.

Attacks on the sequential part of the demo circuit: The attack **sequ/zerodelay/equal/cc/SBout/HD** has confirmed that also mCMOS flip-flops are secure against first-order DPA attacks if glitches (and early propagation) are not considered in the underlying power simulation. This is different to CMOS FFs, which also have had a leakage in this attack scenario.

The more realistic attacks **sequ/unitdelay/equal/cc/SBout/HD** and **../tr/SBout/HD** on the mCMOS FFs have demonstrated the following. In the clock cycle where the attacked intermediate result appears at the FF inputs, no DPA peak could be achieved due to the effect of glitches (and early propagation in the tr-simulation). This result is identical to the corresponding one in CMOS. On the other hand, also in the next clock cycle, where the attacked result appears at the FF output, no DPA peak could be achieved. The reason is that masking is used and that no glitches and practically no early propagation occur at FF outputs. The CMOS results showed a DPA peak in that scenario.

By switching to **../SBin/ZV** attacks, it has been possible to turn the attacks into successful ones. In both clock cycles, where the attacked result is processed by the mCMOS FF, DPA peaks have been achieved for the correct key. In both cases, the leakage has been caused by the signal activities at the FF inputs. In the first clock cycle, the ZV effect occurs when the value 0 (unmasked) overwrites the old value at the input of the SubBytes module: DPA peak of around 0.166 (cc). In the subsequent clock cycle, again a noticeable ZV effect occurs when the value 0 (unmasked) gets overwritten by a new value at the FF input: DPA peak of around 0.0433 (cc). In case also early propagation is included in the power simulation, the DPA peaks for the correct key rise to around 0.243 (tr) in the first clock cycle and to around 0.116 (tr) in the second clock cycle.

Attacks on the combinational part with random weighting: A final investigation that used random weights for the signal transitions in the power simulation confirmed the following. The attack **comb/zerodelay/random/cc/SBout/HD** (no glitches, no early propagation), which has not been successful with unit weighting due to the use of masking, does still not lead to a DPA peak for the correct key. Thus, the problem of masked logic styles like mCMOS is not directly the different capacitive loads of wires but the indirectly caused timing effects. The important conclusion at this point is that any effective masking logic style must take these timing effects into account.

5

Solving the Glitch Problem - MDPL

The theoretical and practical results presented in the last chapter have clearly shown that signal glitches are a source of PA leakage if they appear in cryptographic circuits masked at the cell level (see Sections 4.3 and 4.4 for details). After this so-called "glitch problem" emerged, researchers started to propose masked logic styles which avoid the occurrence of signal glitches. Two main approaches have been pursued so far to achieve this goal. The first one is to prevent the evaluation of the combinational cells at a specific logic level of a design until all cell input signals have reached their final value of the current clock cycle. After this happened, the combinational cells get enabled and their output signals can propagate to the next level of presently still disabled cells. The second approach is to restrict a logic circuit to a behavior and functionality that per se avoids signal glitches.

An example of the first approach (delayed evaluation) is random switching logic (RSL), which is presented in detail in Section 2.4.5. In the context of glitch prevention, the disabling and enabling of combinational cells must be implemented carefully. On the one hand, if the input signals of a disabled cell are just stopped until all have reached their final value, glitches might still be produced after enabling the propagation of the input values through the cell. On the other hand, glitches might still happen internally of a cell if only the outputs of it are disabled until the final output value has been calculated. At least, it avoids that glitches propagate to the inputs of the subsequent cells.

The second approach of glitch prevention does not have this problem because in that case, glitches are intrinsically avoided due to the cell functionality and circuit behavior. In our research, we chose the second approach and developed the masked dual-rail precharge logic (MDPL) style [PM05]. In the next section, MDPL and variants of it are presented in detail. The second part of this chapter

is dedicated to introducing semi-custom design flows for digital circuits in general and to discuss the necessary modifications and extensions if PA-resistant logic styles like MDPL are used in the design process.

5.1 Masked Dual-Rail Precharge Logic (MDPL)

MDPL uses masking at the cell level and avoids glitches in the circuit by using a dual-rail precharge (DRP) approach. Due to the use of masking, no special constraints for the place and route process like the balancing of complementary wires have been assumed to be necessary. As a main feature, all MDPL cells can be built from standard CMOS cells that are commonly available in standard-cell libraries. This saves the costs and efforts of designing MDPL standard cells from scratch and also allows to easily switch the underlying process technology (other feature size, other semiconductor foundry).

MDPL uses the same mask m for all signals in the circuit. Each masked signal v_m that is processed in an MDPL circuit corresponds to an unmasked value $v = v_m \oplus m$. The mask is updated in every clock cycle. MDPL circuits are furthermore implemented as dual-rail precharge circuits. Hence, for each signal v_m also the complementary signal $\overline{v_m}$ is present in the circuit. This is solely done to avoid glitches. Therefore, it is not necessary to balance complementary wires in an MDPL circuit.

Table 5.1: Possible energy consumptions caused by the masked value v_m, which is encoded on two complementary and precharged nodes, during the evaluation phase of clock cycle t.

$v(t)$	$m(t)$	$v_m(t)$	$\overline{v_m(t)}$	Energy	Probability
0	0	0	1	$\overline{E_{01}}$	$\frac{1}{2}p_0$
0	1	1	0	E_{01}	$\frac{1}{2}p_0$
1	0	1	0	E_{01}	$\frac{1}{2}p_1$
1	1	0	1	$\overline{E_{00}}$	$\frac{1}{2}p_1$

Similar as for single-rail, non-precharged masked circuits in Section 4.1, we investigate the basic DPA resistance of MDPL in the following. Table 5.1 shows the possible energy consumptions in the evaluation phase of clock cycle t for a value $v(t)$ that is masked with $m(t)$ and encoded on two complementary and precharged wires. E_{01} denotes the consumed energy when the non-inverted wire has a $0 \rightarrow 1$ transition. Likewise, $\overline{E_{01}}$ denotes the consumed energy when the corresponding inverted wire has a $0 \rightarrow 1$ transition. These two energy values are typically not equal because complementary wires are not balanced in MDPL circuits. The values p_0 and p_1, respectively, denote the probability that $v(t) = 0$ and $v(t) = 1$, respectively. In a first-order DPA attack, the expected values of the mean energy M for the cases $v(t) = 0$ and $v(t) = 1$ are calculated as shown in Equations 5.1 and 5.2. Furthermore, the difference of the means is determined.

Since the two expected values for the means are equal, the expected value of the difference of the means is zero. This shows that first-order DPA attacks are not possible in the given power model.

$$\mathcal{E}(M_{v(t)=0}) = \frac{\frac{1}{2}p_0\overline{E_{01}} + \frac{1}{2}p_0E_{01}}{\frac{1}{2}p_0 + \frac{1}{2}p_0} = \frac{1}{2}\left(\overline{E_{01}} + E_{01}\right) \tag{5.1}$$

$$\mathcal{E}(M_{v(t)=1}) = \frac{\frac{1}{2}p_1E_{01} + \frac{1}{2}p_1\overline{E_{01}}}{\frac{1}{2}p_1 + \frac{1}{2}p_1} = \frac{1}{2}\left(\overline{E_{01}} + E_{01}\right) \tag{5.2}$$

5.1.1 MDPL Cells

In the following, we discuss combinational MDPL cells in general. The structure of sequential MDPL cells is quite different from the one of combinational MDPL cells. Sequential MDPL cells are introduced later by looking at example implementations of MDPL D-flip-flops.

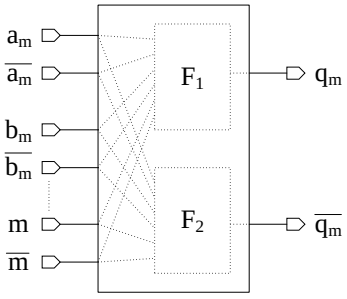


Figure 5.1: Generic structure of a combinational MDPL cell.

Figure 5.1 shows the generic structure of a combinational MDPL cell. It basically consists of two circuits that realize the Boolean functions F_1 and F_2. These functions must be defined in the following way: If the masked input signals a_m, $\overline{a_m}$, b_m, $\overline{b_m}$, ... and the mask signals m and $\overline{m}$ are set to complementary values, masked complementary output values are calculated according to the intended logic function of the cell. Furthermore, it must be ensured that all intermediate values which are necessary to calculate the masked complementary output values are also masked appropriately.

In order to avoid signal glitches at the cell outputs q_m and $\overline{q_m}$, F_1 and F_2 must be *positive monotonic* Boolean functions. The second important effect of using positive monotonic Boolean functions is that combinational MDPL cells set their outputs automatically to the precharge value if all their inputs have been set to the precharge value. In the following, we first discuss positive monotonic Boolean functions in general. Afterwards, we look in detail at the behavior of a combinational MDPL cell during a clock cycle, i.e. during an evaluation phase and a subsequent precharge phase. In all considerations, we assume that the precharge value is 0.

Monotonic Boolean functions: A *monotonic* Boolean function is a function that changes its output value in a monotonic way whenever its input values also change in a monotonic way. Monotonic changes mean that the logic values change only in one direction, i.e. either only $0 \rightarrow 1$ or only $1 \rightarrow 0$ transitions occur. As a result, an output value only performs either a single $0 \rightarrow 1$ or a single $1 \rightarrow 0$ transition as long as the input values of the function are changed in a monotonic way. An example of a monotonic Boolean function is the AND function. An example of a Boolean function that is not monotonic is the XOR function.

A *positive monotonic* Boolean function is a monotonic Boolean function where the directions of the monotonic input changes and the resulting monotonic output change are the same. Accordingly, a *negative monotonic* Boolean function is a monotonic Boolean function where the directions of the monotonic input changes and the resulting monotonic output change are opposite. An example of a positive monotonic Boolean function is the OR function. An example of a negative monotonic Boolean function is the NAND function.

Positive monotonic Boolean functions have the property that if all input values are set to 0 the output value of the function is also 0. Otherwise, $0 \rightarrow 1$ changes of the input values could not cause a $0 \rightarrow 1$ change of the output value of the function, because the output value would already be 1. With the same argumentation, it can be shown that if all input values are set to 1, the output of a positive monotonic Boolean function is 1.

Evaluation phase: At the onset of the evaluation phase, all complementary input signals of a combinational MDPL cell have been set to the precharge value 0 because of the preceding precharge phase. Since only positive monotonic Boolean functions are allowed, the complementary output signals of the cell have also been set to 0. When the input signals are then set to complementary values, only $0 \rightarrow 1$ transitions occur at the inputs of the MDPL cell and thus at the inputs of the positive monotonic Boolean functions F_1 and F_2. As a result, only one transition can occur at each of the two complementary outputs q_m and $\overline{q_m}$ of an MDPL cell. But since F_1 and F_2 always calculate complementary output values for complementary input values, only one complementary output of the MDPL cell performs a $0 \rightarrow 1$ transition while the other output stays at 0. No glitches occur at the MDPL cell's outputs.

Precharge phase: In the subsequent precharge phase, all complementary input signals of the combinational MDPL cell that have been set to 1 are now set back to 0. This means that only $1 \rightarrow 0$ transitions occur at the inputs of the positive monotonic Boolean functions F_1 and F_2. As a result, only one $1 \rightarrow 0$ transition can occur at the output of each function, and thus, at each complementary output of the MDPL cell. A transition happens when the output that has been set to 1 in the preceding evaluation phase is set back to 0. Also during the precharge phase, no glitches occur at the outputs of combinational MDPL cells.

Examples of Combinational MDPL Cells

In the following section, we show the implementations of MDPL AND, OR, and XOR cells with single-rail CMOS cells. The most basic combinational cell is the MDPL AND, all other combinational MDPL cells are based on this cell.

MDPL AND: An MDPL AND cell takes six values as input (a_m, $\overline{a_m}$, b_m, $\overline{b_m}$, m, $\overline{m}$) and produces two output values (q_m, $\overline{q_m}$). The truth table of an MDPL AND cell for complementary input values is shown in Table 5.2. The masked output values of the MDPL AND cell are calculated according to the following equations: $q_m = ((a_m \oplus m) \wedge (b_m \oplus m)) \oplus m$ and $\overline{q_m} = ((\overline{a_m} \oplus \overline{m}) \wedge (\overline{b_m} \oplus \overline{m})) \oplus \overline{m}$.

Table 5.2: Truth table of an MDPL AND cell for complementary input values.

Line no.	a_m	b_m	m	q_m	$\overline{a_m}$	$\overline{b_m}$	$\overline{m}$	$\overline{q_m}$
1	0	0	0	0	1	1	1	1
2	0	0	1	0	1	1	0	1
3	0	1	0	0	1	0	1	1
4	0	1	1	1	1	0	0	0
5	1	0	0	0	0	1	1	1
6	1	0	1	1	0	1	0	0
7	1	1	0	1	0	0	1	0
8	1	1	1	1	0	0	0	0

In Table 5.2, it can be seen that q_m and $\overline{q_m}$ can be calculated by the so-called majority (MAJ) function. The output of this function is 1 if more inputs are 1 than 0. Otherwise, the output is 0: $q_m = MAJ(a_m, b_m, m)$ and $\overline{q_m} = MAJ(\overline{a_m}, \overline{b_m}, \overline{m})$. A majority cell is a commonly used cell and it is available in a typical CMOS standard-cell library. The transistor schematic of a CMOS majority cell is shown in Figure 5.2. The cell schematic of an MDPL AND built with two CMOS majority cells is shown in Figure 5.3. An MDPL NAND cell can be built by swapping (=inverting) the complementary wires of the output signal.

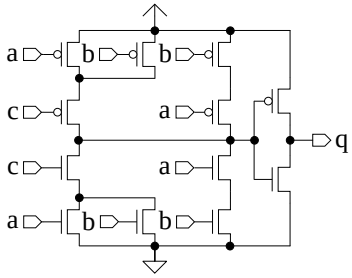


Figure 5.2: Transistor schematic of a CMOS majority cell.

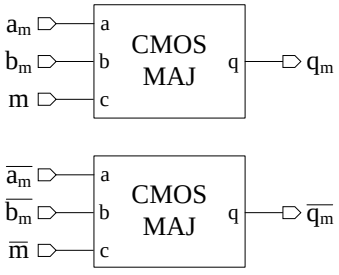


Figure 5.3: Cell schematic of an MDPL AND cell.

The majority function is a positive monotonic Boolean function. Thus, an MDPL AND (NAND) cell produces no signal glitches at its outputs. Furthermore, the outputs of the MDPL AND cell are precharged correctly if all six input values are set to the precharge value. This can also be seen in the truth table of the MDPL AND given in Table 5.2 when looking at lines 1 and 8.

MDPL OR: According to the truth table of an MDPL OR cell shown in Table 5.3, an MDPL OR can be constructed from an MDPL AND by swapping the complementary wires carrying the mask value. An MDPL NOR is then built by furthermore swapping the complementary wires of the output signal. Note that the swapping of complementary wires does not invalidate the considerations made for the MDPL AND cell concerning propagation of the precharge value and glitches. Therefore, also the MDPL OR and NOR cells propagate the precharge value correctly and produce no signal glitches.

Table 5.3: Truth table of an MDPL OR cell for complementary input values.

Line no.	a_m	b_m	m	q_m	$\overline{a_m}$	$\overline{b_m}$	$\overline{m}$	$\overline{q_m}$
1	0	0	0	0	1	1	1	1
2	0	0	1	0	1	1	0	1
3	0	1	0	1	1	0	1	0
4	0	1	1	0	1	0	0	1
5	1	0	0	1	0	1	1	0
6	1	0	1	0	0	1	0	1
7	1	1	0	1	0	0	1	0
8	1	1	1	1	0	0	0	0

MDPL XOR: Figure 5.4 shows the cell schematic of an MDPL XOR that is built with three MDPL NAND cells. Note that the connections for the mask signals have been omitted for the sake of clarity. Using two 3-input CMOS XOR cells to build an MDPL XOR (like the MAJ cells are used in the MDPL AND) would also lead to a functionally correct cell. However, it would not be free of glitches since an XOR is not a monotonic Boolean function. Implementing the MDPL XOR cell with three MDPL NAND cells prevents glitches. Also the precharge value propagates correctly trough such an MDPL XOR cell. An MDPL XNOR is realized by swapping the complementary wires of the output signal.

MDPL D-Flip-Flop

Figure 5.5 shows the cell schematic of an MDPL D-flip-flop. It is a complex cell that has to perform three operations. First, the mask of the input signal must be changed from the current mask $m(t)$ to the mask $m(t+1)$ that is used in the next clock cycle. This change of the mask is performed by the CMOS AND, OR, and MAJ cells at the input of the MDPL D-flip-flop. The circuit built by these

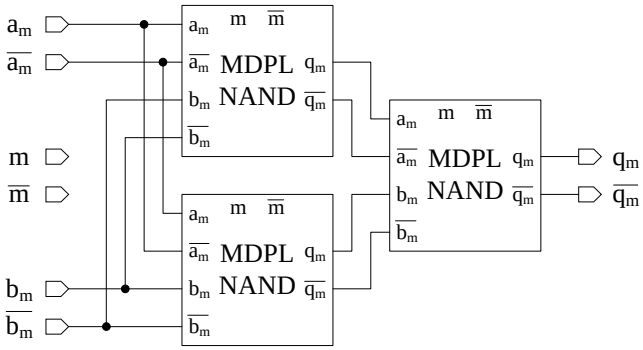


Figure 5.4: Cell schematic of an MDPL XOR cell.

cells takes as inputs $d_{m(t)} = d \oplus m(t)$ and $m(t) \oplus m(t+1)$ (both non-inverted and inverted). It calculates as output $d_{m(t+1)} = d \oplus m(t+1)$. The AND, OR, and MAJ cells are positive monotonic functions, and hence, the circuit responsible for mask changing is precharged correctly during the precharge phase and produces no glitches.

The second operation of the MDPL D-flip-flop is realized by the CMOS D-flip-flop. This flip-flop stores the value $d_{m(t+1)}$, which is masked with the new mask $m(t+1)$ of the next clock cycle, at the positive clock edge. Note that no precharge value needs to be stored in the CMOS D-flip-flop. This is because the MDPL D-flip-flip stores masked values only.

The third operation is realized by the two CMOS NOR cells at the output of the MDPL D-flip-flop. During the precharge phase ($clk = 1$) the outputs of the NOR cells and thus the outputs $q_{m(t+1)}$ and $\overline{q_{m(t+1)}}$ of the MDPL D-flip-flop are set to the precharge value 0. This starts the precharge wave for the combinational MDPL cells connected to the outputs of the MDPL D-flip-flop. In the subsequent evaluation phase ($clk = 0$), the complementary values at the outputs of the CMOS D-flip-flop are just passed through the CMOS NOR cells.

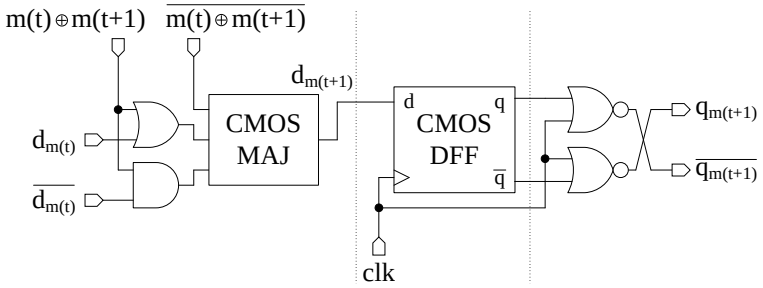


Figure 5.5: Cell schematic of an MDPL D-flip-flop.

MDPL D-flip-flop without timing constraint: In the MDPL D-flip-flop shown in Figure 5.5, there is a timing constraint at the onset of the precharge phase that must be satisfied. When the positive edge of the clock signal *clk* arrives, the two CMOS NOR cells must switch their outputs to the precharge value 0 before the CMOS D-flip-flop provides the new value $d_{m(t+1)}$ at its output. Otherwise, there may be glitches introduced in the circuit. However, this timing constraint is satisfied because NOR cells are faster than D-flip-flops. Furthermore, the CMOS D-flip-flop and the CMOS NOR cells are both leaf cells of the clock tree, which is separately build during the implementation process of a circuit. Therefore, the skew between the clock signals supplied to the CMOS D-flip-flop and the NOR cells is minimal.

There is also an implementation of the MDPL D-flip-flop possible that does not have such a timing constraint. However, it is much bigger and requires a doubling of the clock frequency in order to keep the same data rate. Such an MDPL D-flip-flop is shown in Figure 5.6 and uses four CMOS D-flip-flops: two for storing the precharge value and two for storing the complementary masked data value. With this approach, the two CMOS NORs at the output of the MDPL D-flip-flop can be omitted and the timing constraint is avoided. Note that also the mask changing circuit at the input of the MDPL D-flip-flop gets more complex, because not only $d_{m(t+1)}$ must be calculated but also its inverse $\overline{d_{m(t+1)}}$.

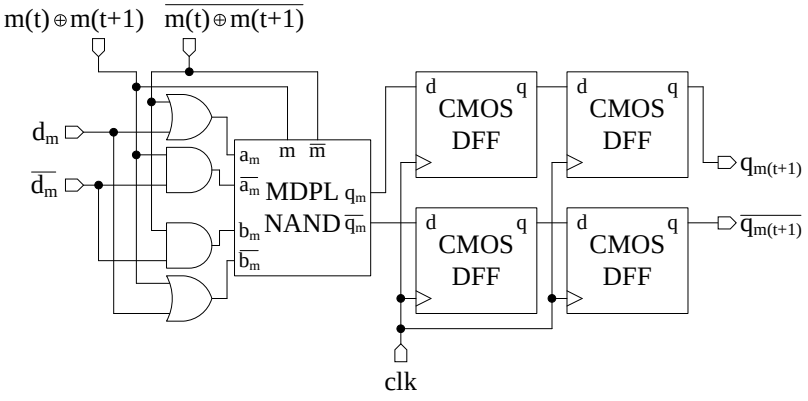


Figure 5.6: Cell schematic of an MDPL D-flip-flop without timing constraint.

MDPL Cells Summary

Table 5.4 summarizes the basic MDPL cells and their respective implementation with standard CMOS cells. The table also shows the area complexity of the MDPL cells when the 0.35 μm standard-cell library C35 of austriamicrosystems [aus] is used for implementation. The area values are given in gate equivalents (GEs) - one GE corresponds to 54.6 μm^2. The area requirements of the MDPL

cells are furthermore compared to the area requirements of their CMOS counterparts. Note that for the MDPL D-flip-flop, the implementation with the timing constraint is considered. As experience has shown, the size of an MDPL circuit compared to the size of the corresponding standard CMOS circuit can be expected to increase by a factor of 4-5.

Table 5.4: MDPL cells and their CMOS implementations.

MDPL cell	CMOS implementation of MDPL cell	Area in GEs of		Ratio $\frac{MDPL}{CMOS}$
		MDPL cell	CMOS cell	
Inverter	Wire swapping	0	0.67	0
Buffer	2×Buffer	2	1	2
AND, OR	2×MAJ	4	1.67	2.4
NAND, NOR	2×MAJ	4	1	4
XOR	6×MAJ	12	2.33	5.14
XNOR	6×MAJ	12	2	6
D-flip-flop	AND, OR, MAJ D-FF, 2×NOR	12.33	5	2.47

5.1.2 MDPL Circuits

The general architecture of an MDPL circuit is shown in Figure 5.7. In MDPL circuits, only the sequential cells are connected to the clock signal. Therefore, only these cells precharge at the same time when the clock signal switches to 1. Furthermore, they also evaluate at the same time when the clock signal switches to 0. Combinational MDPL cells precharge when their inputs have been set to the precharge value. They evaluate, when their inputs have been set to complementary values. MDPL D-flip-flops perform three operations. In the precharge phase, they start the precharge wave. In the evaluation phase, they provide the stored complementary values that are masked with the mask value $m(t)$ of the current clock cycle. Third and last, the MDPL D-flip-flops perform the mask change from $m(t)$ to $m(t+1)$. The mask $m(t+1)$ is the mask value of the next clock cycle.

During the precharge phase, the precharge wave is started at the outputs of the MDPL D-flip-flops. The precharge value that is used in MDPL circuits is 0. In the precharge phase, also the mask signals $m(t)$, $\overline{m(t)}$, $m(t) \oplus m(t+1)$, and $\overline{m(t) \oplus m(t+1)}$ are precharged. This causes the combinational MDPL cells that are directly connected to the outputs of the MDPL D-flip-flops to precharge. Then, the combinational MDPL cells in the next logic level are switched into the precharge phase and so on. At the end of the precharge phase, all MDPL cells in the combinational circuit have been precharged. Furthermore, the MDPL D-flip-flops store the masked values for the subsequent evaluation phase. Let us assume that these values are masked with the mask value $m(t)$. In the subsequent evaluation phase (starts with the negative clock edge), the MDPL D-flip-flops provide their stored masked values to the combinational MDPL cells. Since these

stored values are masked with mask $m(t)$, this mask must also be provided to the combinational MDPL cells. The MDPL D-flip-flops are provided with the mask $m(t) \oplus m(t+1)$ and its inverse. The flip-flops change the masks of the input values from $m(t)$ to $m(t+1)$ before these re-masked values are stored at the beginning of the next precharge phase (starts with the positive clock edge).

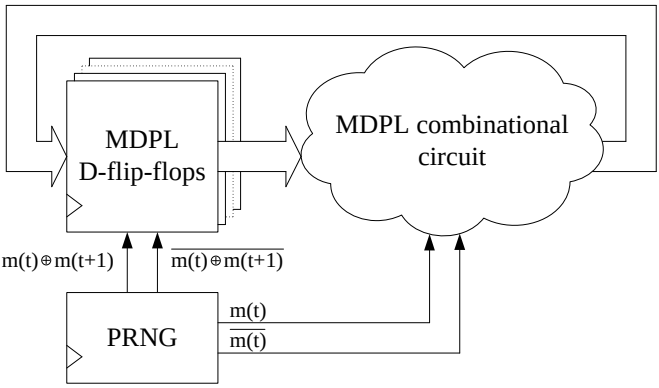


Figure 5.7: Architecture of an MDPL circuit.

Signals entering the MDPL domain must be precharged, made complementary, and masked with the mask value $m(t)$. Signals leaving the MDPL domain must be unmasked, converted to single-rail, and the precharge phase must be removed. Interfacing cells that perform these operations are presented in Section 7.1.1.

The transitions of the mask values cause significant current peaks in MDPL circuits because the mask nets are usually quite big. The situation is made worse by the fact that the transitions of the mask values coincide with the transition of the clock signal. Note that the mask nets in MDPL circuits are complementary wires that are precharged. This allows balancing the power consumption of the mask nets to a degree that prevents SPA attacks, i.e. it is not possible to determine the mask values by looking at the power traces of the circuit.

The masks of the MDPL circuit are generated by a pseudo-random number generator (PRNG), which is in the best case seeded by a true-random number generator (TRNG) like [BBL04]. An MDPL circuit requires only one new masking bit per clock cycle. A mask preparation unit that generates the necessary mask values $m(t)$, $\overline{m(t)}$, $m(t) \oplus m(t+1)$, and $\overline{m(t) \oplus m(t+1)}$ from the new masking bit is presented in Section 7.1.1.

The area requirements of MDPL circuits are in general at least quadrupled compared to corresponding unmasked CMOS circuits, while the maximum clock rates are typically halved. The power consumption of MDPL circuits is significantly increased because MDPL circuits are DRP circuits and the mask nets must be switched. In order to get more concrete values for the circuit properties of MDPL circuits, we have implemented an AES module in MDPL [PM06]. The implementation results are summarized in the following. An in-depth analysis

of the PA resistance of MDPL is presented in Chapter 6. The main outcome of the analysis is that the PA resistance of MDPL circuits suffers from an early-propagation behavior of the MDPL cells

AES MDPL Module

An AES module suitable for RFID tags [FDW04, FWR05] has been implemented in MDPL. For the implementation of the MDPL cells, the 0.35 μm standard-cell library C35 of austriamicrosystems has been used. To implement the MDPL circuit, the semi-custom design flow suitable for masked logic styles which is described in Section 5.2.2 has been used. For comparison, the AES module has also been implemented in the unprotected 0.35 μm CMOS logic style.

In Table 5.5, the main properties of the MDPL and the CMOS implementations of the AES module are compared. The figures given in the table have been determined for circuits including the clock tree. The power simulations were done with transistor netlists including extracted parasitic capacitances using the simulator NanoSim from Synopsys.

The area requirements of the MDPL circuit are almost five times higher than the area requirements of the corresponding CMOS circuit. The speed of the MDPL circuit is approximately halved. This is due to the fact that not only the evaluation wave but also the precharge wave must propagate through the combinational logic. The increase of the power consumption of the MDPL circuit compared to the CMOS circuit is significant. However, the comparison given herein is not generally representative because the CMOS reference implementation has been specifically trimmed to low power consumption. The signal activities in the CMOS circuit have been minimized by low-power techniques like clock gating and sleep-mode logic [WH04]. Porting these techniques to MDPL and similar masked logic styles significantly reduces the power consumption of such circuits as well [FP08]. However, these low-power techniques have not been applied to the current MDPL circuit.

Table 5.5: Comparison of the properties of an AES module implemented in CMOS and MDPL.

	Area (GE)	Speed (MHz; worst-case corner)	Avg. current (μA@100 $kHz, 3.3\,V$)
CMOS	3452	17.02	7.02
MDPL	16421	10.04	122.34
Ratio $\frac{MDPL}{CMOS}$	4.76	0.59	17.43

5.1.3 MDPL Variant with Disengageable Masking and Precharging

The power consumption of an MDPL circuit is quite high compared to a CMOS implementation of the same circuit. This is due to the fact that MDPL ran-

domizes data values with a masking bit that changes every clock cycle with a probability of $\frac{1}{2}$. Furthermore, MDPL is a dual-rail precharge logic style. Besides the low-power techniques that have been mentioned in the last section, there exists another approach to reduce power consumption if an MDPL circuit does not need to operate in a PA-resistant manner at all times.

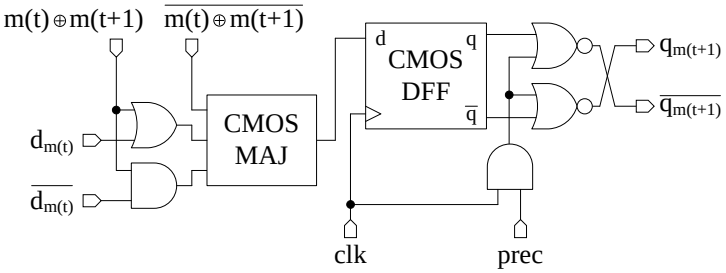


Figure 5.8: MDPL D-flip-flop with disengageable precharging.

In MDPL circuits, it is easily possible to turn off precharging. It requires only a small modification of the MDPL D-flip-flops as shown in Figure 5.8 and of the CMOS-to-MDPL interface where input signals get precharged. The signal *prec* controls whether or not precharging is activated. Furthermore, the mask can be set to a fixed value. These two measures lead to a significant decrease of the power consumption of the AES MDPL module that has been investigated in the last section.

Table 5.6 summarizes the circuit properties of the AES MDPL module that uses the new MDPL D-flip-flop with disengageable precharging. Three different operation modes are distinguished: normal (full precharging and masking); mask value is fixed and complementary mask nets are not precharged; mask value is fixed and complementary mask nets and circuit signals are not precharged. In the latter operation mode, the power consumption is reduced by almost a factor of three compared to the results shown in Table 5.5. However, it is clear that while the MDPL circuit is in this operation mode the circuit is no longer PA-resistant. Such an insecure operation mode is for example suitable for the rounds three to eight of AES (ten rounds in total) where no DPA attack is possible. It is also suitable for microprocessors which do not always perform security-critical operations that are susceptible to DPA attacks.

Table 5.6 also shows that the modification of the MDPL D-flip-flop increases the area requirements of the AES implementation. The major contribution to the area increase is not caused by the additional AND gate required in a modified MDPL D-flip-flop. However, the additional AND gate is part of the clock network and leads to a much bigger clock tree than before. Optimization of the clock tree should be possible. The bigger clock tree also leads to an increased power consumption of the new AES module in normal mode compared to the original AES module. Table 5.6 furthermore shows that when precharging is completely deactivated, the AES circuit can be operated at double speed, i.e. at

Table 5.6: Operation modes of MDPL circuits and the resulting circuit properties.

Operation mode	Area (GE)	Speed (MHz)	Avg. current (μA@100 $kHz, 3.3\,V$)
Normal	18831	9.73	171.59
Mask value fixed, mask nets not precharged	18831	9.73	164.24
Mask value fixed, mask nets and circuit signals not precharged	18831	19.46	43.58

approximately the same speed as the original CMOS circuit (see Table 5.5).

5.2 Semi-Custom Design and PA-Resistant Logic Styles

Digital circuits are typically implemented on a chip by the help of a semi-custom design approach. Semi-custom design is characterized by the fact that a circuit is built from a limited set of predefined basic cells (combinational cells, sequential cells, I/O cells, etc.). These predefined cells are the so-called standard cells. Standard cells are defined in a library that describes them in various ways. The description typically includes the logic function, the timing behavior, the power consumption, and the layout of the cells. A standard-cell library needs to be created only once for a given process technology.

Semi-custom design allows to implement very large and complex designs and to produce chips that are functionally correct with a high probability. Therefore, it is important that also PA-resistant logic styles are useable in semi-custom design flows. In the following, we first present the general steps of semi-custom circuit design. Afterwards, we will discuss the modifications and extensions to the design process that are typically necessary if PA-resistant logic styles are used. In particular, we will show the major differences in the semi-custom design flows of hiding and masked logic styles.

5.2.1 Semi-Custom Circuit Design in General

In a semi-custom design flow, the high-level description of a digital circuit is mapped to the logic cells of the standard-cell library. This conversion process is called synthesis. Many steps of the design flow can be done automatically due to specific restrictions that have been applied to the logic cells. There exists a large number of so-called electronic design automation (EDA) tools that can be used for this purpose.

The typical steps of a semi-custom design flow using standard cells are illustrated in Figure 5.9 and described in the following.

- **High-level design capture:** In this step, a description of the digital circuit is entered into the design system. This can be done at different

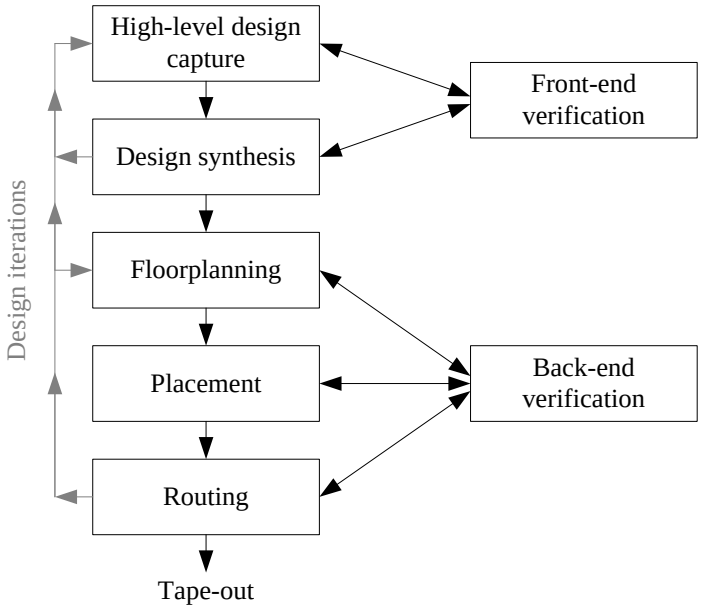


Figure 5.9: Semi-custom design flow using standard cells.

levels of abstraction. For high levels of abstraction, behavioral description languages like SystemC, SystemVerilog, or SystemVHDL are used. For low levels of abstraction, hardware description languages (HDLs) like Verilog or VHDL are used. Descriptions based on HDLs are usually done at the register-transfer level (RTL). In this case, the combinational and the sequential behavior of the circuit are already defined in a cycle-accurate fashion.

- **Design synthesis:** Depending on the abstraction level used for design capture, different types of design synthesis are necessary. *Behavioral synthesis* supports the process of taking various high-level architectural decisions like the separation of a design into a hardware and a software part. Often, such a process can be automated only partially. The result of behavioral synthesis can for example be an RTL description of the digital circuit expressed using an HDL. Then, *logic synthesis* is used to continue in the design process. In logic synthesis, the RTL description of the digital circuit is mapped to the actual cells that are available in the standard-cell library. Logic synthesis often also involves different optimizations of the digital circuit. It is nowadays done quite efficiently by EDA tools in an automated way. The result of logic synthesis is a *netlist* that consists of standard cells and sometimes also of so-called macro cells. These are predefined, complex modules like multipliers, memories, or even complete microcontrollers.

- **Floorplanning:** In this step, the overall layout of the chip (height, width, aspect ratio, etc.) is defined based on the number and size of the cells in the circuit. Regions for the core logic (logic cells that implement the actual functionality of the digital circuit), the I/O cells, and the power supply cells are defined. Furthermore, the layout of the power supply grid of the chip is designed.

- **Placement:** In this step, the precise placement of the cells of the digital circuit is done. The placement is influenced by the connections between the cells. As a result, the placement tool tries to place cells that are connected closely to each other. During placement, an important subtask is the generation of the clock tree. The clock tree distributes the clock signal to all (sequential) cells in the circuit. In a synchronous digital circuit, it is important that the clock signal arrives at all cells at nearly the same time.

- **Routing:** In this step, the wire connections between the cells are established. This is done according to the connections that are specified in the netlist and the actual placement of the cells. Modern routing tools can change the cell placement to some extent in order to solve problems like infeasible routes.

- **Front-end verification:** In this step, the functionality of the design is verified. This is mainly done by functional and logical simulations. In addition, a so-called static timing analysis of the digital circuit is done for performance checking. Typically, simulations and static timing analysis are based on the estimated physical properties of cells and wires in the digital circuit. For increased accuracy, it is possible in later design steps to extract more accurate physical information (e.g. signal timing information) from the layout in order to include this information in the simulations and the timing analysis. This process is called *back-annotation*.

- **Back-end verification:** In this step, many different checks are performed to verify the functionality of the layout of a digital circuit. It relies heavily on the fact that circuit parasitics can be accurately estimated at this level. *Circuit parasitics* are unwanted electronic elements that occur in all fabricated circuits, e.g. capacitances between wires. The main tasks of back-end verification are design-rule checking (checking geometric design rules of transistor structures and wires in the layout), circuit extraction (extraction of the transistor netlist from the layout including exact transistor sizes and circuit parasitics), and layout-versus-schematic checking (checking that the placed and routed netlist is functionally still the same as the synthesized netlist).

- **Tape-out:** When all design constraints are met and the various verification steps were successful, a file containing the circuit layout is generated and sent to the semiconductor foundry. The file contains all information that is necessary for the chip manufacturing process.

In modern semi-custom design flows, the steps are often not performed strictly sequentially as Figure 5.9 might suggest. Today, an integrated approach is necessary that usually includes more than one iteration through the design process. The reason is that it is a very challenging task to accurately estimate the physical behavior of digital circuits that are implemented using modern process technologies.

5.2.2 Semi-Custom Circuit Design for PA-Resistant Logic Styles

In general, also PA-resistant logic styles based on the concepts of masking or hiding can be used in semi-custom design flows. If this does not apply to a particular PA-resistant logic style, its practical usability is significantly reduced. The PA-resistant standard cells of adequate logic styles for semi-custom design can be applied independently of the selected cryptographic algorithms and the chosen circuit architectures. Therefore, the high-level design of cryptographic devices can be done independently of the underlying PA-resistant logic styles, which are automatically inserted later in the design process.

When using PA-resistant logic styles, some extensions to a standard semi-custom design flow as depicted in Figure 5.9 are necessary. There are essentially two main reasons for these extensions. First, logic synthesizers typically cannot use PA-resistant standard cells directly. Most logic synthesizers are specialized in using single-rail (SR) cells like standard CMOS cells and do not support the complex functionality of PA-resistant cells. The usual solution to this problem is that logic synthesis of the high-level design is at first performed using an SR cell library. The resulting SR cell netlist is then converted to a netlist containing the corresponding PA-resistant cells. This process is usually called *logic-style conversion*. It also includes various additional tasks that are necessary to implement a circuit in a particular PA-resistant logic style. Examples are the inclusion of interfacing circuitry for the signals going into and out of PA-resistant domains and the insertion of a PRNG in case random numbers are required. In the course of the work for this thesis, we have implemented and analyzed a logic-style conversion tool using a proprietary design database and one that is based on the standardized OpenAccess (OA) framework [VHUP05]. While the OA framework is very versatile and eases the tool implementation, the resulting converter was significantly slower than that one using the proprietary database. The analysis indicated that this might change only for very large designs.

The second reason for extensions of the design flow is that in most cases, special constraints of the PA-resistant logic styles must be enforced. Typical constraints are the pairwise balanced routing of complementary wires especially in the case of hiding logic styles or a specific timing sequence of signals propagating through the PA-resistant circuits. The necessary extensions to a standard semi-custom design flow are shown in gray in Figure 5.10. In the following, the steps of a semi-custom design flow for PA-resistant circuits are discussed in more detail.

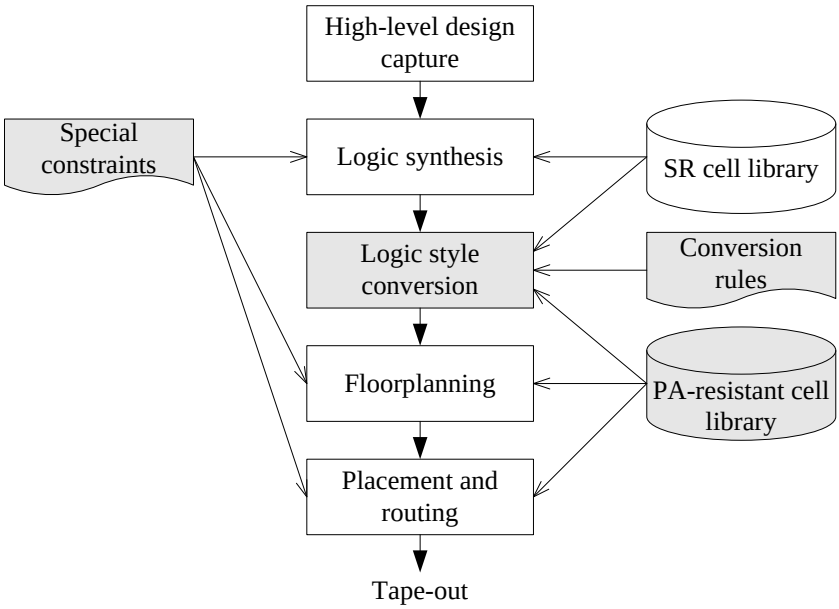


Figure 5.10: Semi-custom design flow using PA-resistant logic styles.

- **High-level design capture:** A main issue during high-level design is that sensitive intermediate results must not leave the modules of cryptographic circuits that are implemented in PA-resistant logic styles. It must be assumed that all other circuit parts are not sufficiently protected against power analysis attacks. Therefore, any sensitive intermediate result that leaves the secured part of a cryptographic circuit would cause a vulnerability to PA attacks. PA-resistant logic styles typically significantly increase the area requirements and the power consumption of cryptographic circuits and reduce their speed. Thus, usually not all modules of a cryptographic circuit are implemented in a secure manner.

- **Logic synthesis:** Logic synthesizers typically cannot use PA-resistant cells directly. Therefore, logic synthesis is done using an appropriate SR cell library. Special constraints are applied to ensure that only such SR cells are used for which corresponding PA-resistant cells exist.

- **Logic-style conversion:** The main steps of the logic-style conversion process are cell substitution, adaptation of the signal nets, and addition of SR interfaces to the inputs and outputs of the converted circuit. During cell substitution, all SR cells are replaced by the corresponding PA-resistant cells. The logic function, the timing behavior, and the layout of the inserted cells are provided by a PA-resistant cell library. The mapping between SR cells and PA-resistant cells is defined by conversion rules. In the second step, the signal nets of the SR circuit are adapted if necessary, e.g. doubled

to get complementary wires. Any additional wires must be connected to the appropriate inputs and outputs of the PA-resistant cells according to the conversion rules. The adaptation is usually done for all signal nets in the SR circuit except for dedicated SR signals like the clock signal or the asynchronous reset signal. Furthermore, it might be necessary to also connect all combinational PA-resistant cells to the clock tree or that one or more mask nets must be added to the circuit and connected appropriately to the masked cells. In the third step of the logic-style conversion process, typically an SR interface is added to the PA-resistant circuit. This interface ensures that all SR signals entering and leaving the PA-resistant circuit are conditioned as necessary. Examples of necessary signal modifications are adding and removing a precharge phase and the conversion of single-rail signals to dual-rail signals and vice versa. At the end of the logic-style conversion process, it must be ensured that the output load of each PA-resistant cell in the circuit is still below the maximum value specified in the cell library.

- **Floorplanning, placement, and routing:** During these steps of the semi-custom design flow, the special constraints of the used PA-resistant logic style must be enforced. In the following, we illustrate a common situation by means of the concrete scenario of hiding logic styles that require complementary wires to be pairwise balanced. "Pairwise balanced" means that the capacitances and resistances of a complementary wire pair are identical, which ensures that signal timing and power consumption is the same no matter what wire is charged in a clock cycle. The same must hold true for the discharging event. However, perfectly balanced complementary wires are impossible to achieve in practice because their exact capacitances and resistances on the final chip are simply not known during the design process. The used models of the wires always introduce inaccuracies. Furthermore, the capacitances of a wire are not only located between the wire and the supply lines (V_{DD} and GND) but also between the wire and its neighboring wires. These capacitances are called cross-coupling capacitances. Thus, the effective capacitance of a wire depends on the states of the neighboring wires, which means that it is data dependent. Unfortunately, the cross-coupling capacitances become more and more dominant over the capacitances to the supply lines in modern process technologies.

  Two of the proposed methodologies to place and route circuits with hiding logic styles in a balanced way are *differential routing* [TV04c] and *back-end duplication* [GHMP05]. In the differential routing approach, the complementary wires are routed in parallel. A prototype chip that has been implemented using this approach is presented in [THH+05]. In the back-end duplication method, two circuits are produced with the same layout, and one circuit always processes the complementary signals of the other circuit.

6

Evaluating MDPL in Theory and Practice

In the last chapter, the masked logic style MDPL was presented in detail. This logic style avoids signal glitches by construction, so that they do not decrease the PA resistance of the masked circuits (see Sections 4.3 and 4.4). In this chapter, various theoretical and practical evaluation results of MDPL circuits are presented and discussed. The results allow judging the PA resistance of MDPL circuits.

First, evaluation results of MDPL in the context of the *balancing problem* are presented. The term "balancing problem" refers to a main disadvantage many PA-resistant logic style proposals based on hiding have: they often require that the capacitive loads and other parasitic properties of complementary wires are pairwise balanced, i.e. that they are the same (see Section 2.4.1).

The evaluation shows that the masking of MDPL solves the balancing problem if the timing differences caused by input signals with different arrival times and by unbalanced complementary wires are neglected, i.e. if only the energy consumption per switching event (clock cycle) is considered. In the earlier phases of MDPL development, our research focused on solving the balancing issue with respect to energy consumption. The basic assumption was that the timing differences in the power consumption would not have a severe impact on the PA resistance of MDPL circuits.

However, this assumption was subsequently proven to be wrong. In the second part of this chapter, we discuss the evaluation results that clearly show that early propagation as it appears in MDPL circuits has a dramatic effect on the PA resistance of such circuits. This was shown not only theoretically but also verified practically with the help of an MDPL prototype chip. Interestingly,

we found out during analysis that the AES coprocessor that is also part of the prototype chip cannot be attacked quite as easily as the MDPL microcontroller. This indicates that there are circumstances where the early propagation problem does not completely remove the PA resistance of MDPL. We analyze this issue in the third part of this chapter.

In the remainder of this chapter, other specific issues that arose during the evaluation of MDPL circuits in particular and masked circuits in general are discussed. Topics include the question how vulnerable MDPL is due to the use of a single mask bit per clock cycle for all signals and if MDPL has special requirements with respect to the randomness of the mask bit.

6.1 The Balancing Problem

To start the investigation of the balancing problem and its impact on MDPL, we first present analysis results for the PA-resistant DRP logic styles SABL (logic cells are built from scratch) and WDDL (logic cells are built based on standard CMOS cells). These DRP logic styles are introduced in Section 2.4.2 and Section 2.4.3, respectively. The analysis results clearly show that unbalanced complementary wires significantly reduce the PA resistance of the DRP logic styles SABL and WDDL. Subsequently, analysis results will be presented that show that MDPL is immune against unbalanced complementary wires (as long as timing effects are neglected—this issue will later be analyzed in Section 6.2).

6.1.1 PA-Resistant DRP Logic Styles with Balanced Complementary Wires

The PA resistance of DRP logic styles is indirectly proportional to the variance of the instantaneous power consumption of their cells [MOP07]. In the following, we analyze the PA resistance of SABL and WDDL firstly for the (artificial) balanced case and secondly for unbalanced complementary wires by determining the energy variations of NAND cells implemented in both logic styles. For further comparison, the results of the balanced case are also compared to the energy variation of a CMOS NAND cell.

Note that in order to quantify the PA resistance of the cells, we use the variance of the energy consumption over a clock cycle instead of the variance of the instantaneous power consumption. The reason is that an attacker can typically only measure a signal that is proportional to the energy consumption of a cell and not a signal that resembles its instantaneous power consumption. However, this approach also hides the leakage caused by timing variations in the instantaneous power consumption curve, which are still detectable by an attacker. This is the main reason why the early propagation problem of MDPL was not discovered in the beginning.

The energy consumption of the NAND cells implemented in CMOS, SABL, and WDDL has been determined by analog circuit simulations using the simulator *Spectre* from Cadence Design Systems. The schematics of the NAND cells

(SABL: see Figure 2.4; WDDL: see Figure 2.5) have been implemented using the $0.35\,\mu m$, $3.3\,V$ CMOS process technology C35 of austriamicrosystems [aus]. No intra-cell routing parasitics have been considered in the simulations since these parasitics heavily depend on the particular cell layouts. The nominal capacitance at the outputs of the NAND cells (output to *GND*) has been chosen to be $100\,fF$. The power simulations have been performed for all 16 possible combinations of input signal transitions shown in Table 6.1. Note that the transitions have been sorted in a sequence that allows applying them consecutively to the cell inputs.

Table 6.1: All 16 possible combinations of input signal transitions and the resulting output signal transitions of a 2-input NAND cell.

Input signal transitions		NAND output signal transition
Input a	Input b	Output q
$0\to0$	$0\to0$	$1\to1$
$0\to0$	$0\to1$	$1\to1$
$0\to0$	$1\to1$	$1\to1$
$0\to1$	$1\to1$	$1\to0$
$1\to1$	$1\to1$	$0\to0$
$1\to1$	$1\to0$	$0\to1$
$1\to0$	$0\to1$	$1\to1$
$0\to0$	$1\to0$	$1\to1$
$0\to1$	$0\to0$	$1\to1$
$1\to1$	$0\to1$	$1\to0$
$1\to0$	$1\to1$	$0\to1$
$0\to1$	$1\to0$	$1\to1$
$1\to1$	$0\to0$	$1\to1$
$1\to0$	$0\to0$	$1\to1$
$0\to1$	$0\to1$	$1\to0$
$1\to0$	$1\to0$	$0\to1$

Figure 6.1 shows the 16 power traces for each of the three different implementations of the NAND cell. The power traces of the DRP NAND cells show the power consumption of the evaluation phase and the subsequent precharge phase. The variations in the power traces of the CMOS NAND cell are due to the fact that for some input transitions, the output capacitance is charged (either via one or via two transistors connected in parallel—this explains the different power peaks), while for other input transitions, this is not the case. For the DRP NAND cells, all input transitions lead to a charging event for one output capacitance. Since the output capacitances are balanced in this case, the variation of the power consumption is much smaller than for the CMOS NAND cell.

Variance and standard deviation of the energy consumption of the different NAND implementations for balanced complementary wires are shown in Table 6.2. The variance of the energy consumption of the SABL NAND cell is

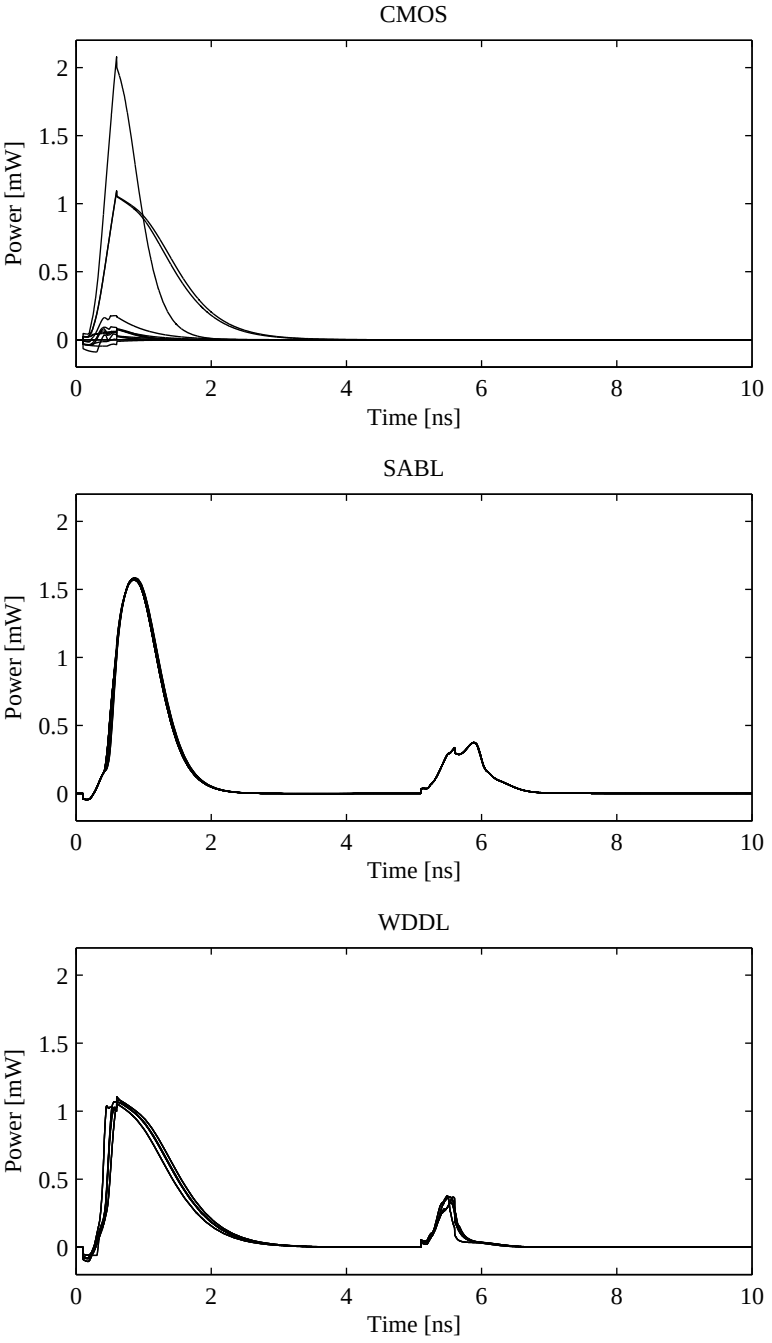


Figure 6.1: Simulated power traces of a CMOS NAND cell, an SABL NAND cell, and a WDDL NAND cell for different input transitions and balanced complementary output wires.

approximately four orders of magnitude smaller compared to the CMOS NAND cell. This is a very good result and shows that SABL has a very high PA resistance in case perfectly balanced complementary wires could be achieved. The variance of the WDDL NAND cell is around three orders of magnitude smaller than the one of the CMOS NAND cell. The reason for the higher variance is that combinational WDDL cells are built in a simple way from basic single-rail cells like CMOS AND and CMOS OR cells. Therefore, the possibilities for fine-tuning the functionality and the layouts of WDDL cells to achieve a lower energy variance (i.e. a higher PA resistance) are limited.

While the variance of the SABL NAND cell is also representative for the SABL D-flip-flop due to the similar cell structure, this is not the case for the WDDL NAND cell and the WDDL DFF. The variance of the energy consumption of the WDDL DFF is typically much lower than the one of the WDDL NAND cell. The reason is that identical single-rail DFFs are used in both datapaths of the WDDL DFF. This leads to a much more balanced power consumption. A WDDL DFF consists of four single-rail CMOS DFFs connected together like those in the special MDPL DFF shown in Figure 5.6.

Table 6.2: Variance and standard deviation of the energy consumption of the CMOS NAND cell, the SABL NAND cell, and the WDDL NAND cell for balanced complementary wires.

Logic style	CMOS	SABL	WDDL
$Var(E_{NAND})$	$22\,469 \cdot 10^{-29}\ J^2$	$1.6954 \cdot 10^{-29}\ J^2$	$26.853 \cdot 10^{-29}\ J^2$
$Std(E_{NAND})$	$474\ fJ$	$4.12\ fJ$	$16.4\ fJ$

6.1.2 PA-Resistant DRP Logic Styles with Unbalanced Complementary Wires

As shown in the last section, the estimated PA resistance of DRP cells is very high if the capacitances at the complementary outputs are balanced. However, this balancing is never perfect in practice, and thus, the PA resistance of DRP cells is reduced. Figure 6.2 shows how the variance of the energy consumption of an SABL NAND cell and a WDDL NAND cell increases when the complementary output wires of the cells become less balanced.

The graphs show that the PA resistance of the SABL NAND cell is maximal in the balanced case (the energy variation at this point is minimal). The PA resistance decreases quadratically (i.e. the variance increases quadratically) with the difference of the capacitances at the complementary outputs q and $\overline{q}$ of the cell. The graph of the WDDL NAND cell shows basically the same behavior as the graph of the SABL NAND cell. An interesting point is that the WDDL NAND cell does not reach its maximum PA resistance when the capacitances at the outputs q and $\overline{q}$ are perfectly balanced. The graph shows that the maximum DPA resistance is reached when the capacitance at output q is around 2 fF lower than the capacitance at output $\overline{q}$. This indicates an unbalanced internal

structure of the WDDL NAND cell. The reason is that the single-rail AND and the single-rail OR cells within the WDDL NAND cell (see Figure 2.5) do not consume exactly the same power when switching their outputs in the same manner. During the design of the cell layout, it is possible to correct the unbalanced internal structure of the WDDL NAND cell to some degree. However, this increases the design and implementation effort of WDDL cells significantly. Note that the minimum energy variance of the WDDL NAND cell (i.e. for $-2\,fF$ capacitance difference) is still slightly higher than the minimum energy variance of the SABL NAND cell.

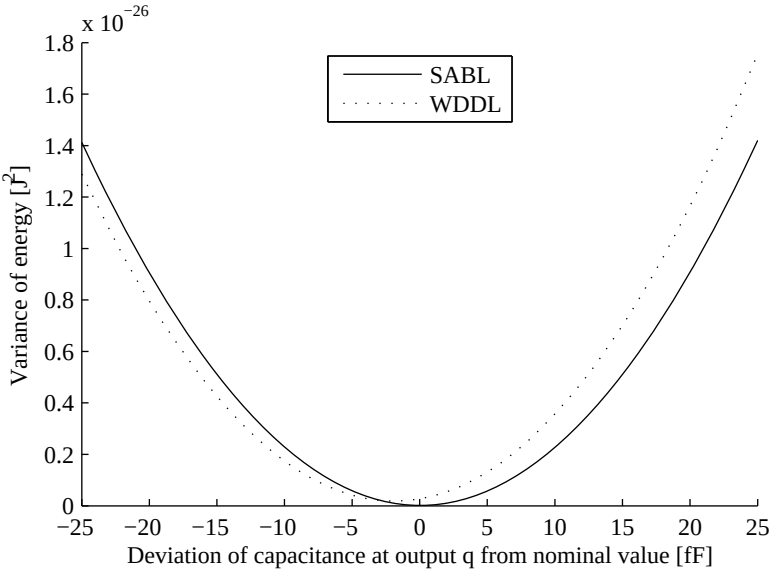


Figure 6.2: Variance of the energy consumption of an SABL NAND cell and a WDDL NAND cell as a function of the difference of the capacitances at the complementary cell outputs q and $\overline{q}$. The nominal capacitance at the outputs is 100 fF.

6.1.3 MDPL with Balanced and Unbalanced Complementary Wires

The first, elementary evaluations of the PA resistance of MDPL indicated that it is completely immune to unbalanced complementary wires. In these evaluations, the figure of merit was based on energy values and their variation for different processed data. The problem is that timing effects in the power consumption caused by differing arrival times of cell input signals and unbalanced complementary wires are not considered in such an analysis. Therefore, the negative effects of this phenomenon on the PA resistance of MDPL went unnoticed in the beginning. Ultimately, it was discovered as shown in Section 6.2.

In a similar practical experiment, we analyzed the effectiveness of MDPL with balanced complementary wires as we did for the DRP logic styles SABL and WDDL in Section 6.1.1. Furthermore, we analyzed how unbalanced complementary wires influence the PA resistance of MDPL and compared these results to the results for SABL presented in Section 6.1.2.

We did this by analyzing the PA resistance of an MDPL NAND cell, which is implemented by an MDPL AND cell shown in Figure 5.3 with swapped complementary outputs. As for the DRP NAND cells in Section 6.1.1, the PA resistance of the MDPL NAND cell has been determined by simulating the variance of its energy consumption for different input transitions. The simulation environment has been the same as for the DRP NAND cells. Figure 6.3 shows the 16 different power traces of an MDPL NAND cell when the 16 combinations of signal transitions given in Table 6.1 have been applied to the inputs of the cell. Every combination of input signal transitions has been simulated for the four possible mask signal transitions. Afterwards, the mean of these four traces has been calculated. Since an attacker does not know the values of the mask, he implicitly does the same during a DPA attack. The power traces show the power consumption of the evaluation phase and the subsequent precharge phase.

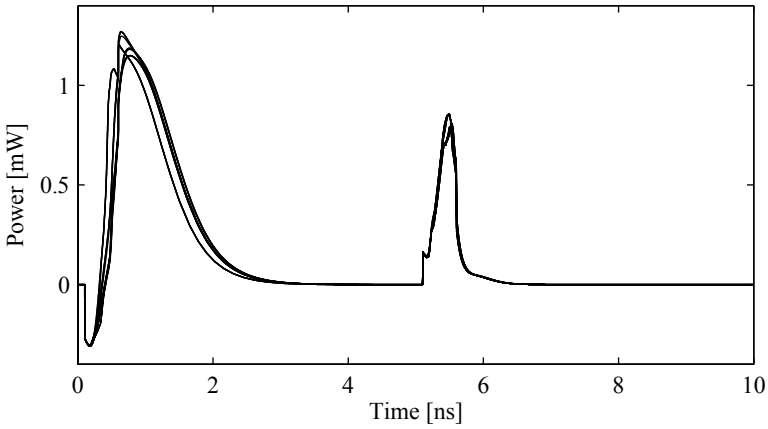


Figure 6.3: Simulated power traces of an MDPL NAND cell for different input transitions and balanced complementary output wires.

Table 6.3: Variance and standard deviation of the energy consumption of the CMOS NAND cell, the SABL NAND cell, and the MDPL NAND cell for balanced complementary wires.

Logic style	CMOS	SABL	MDPL
$Var(E_{NAND})$	$22\,469 \cdot 10^{-29}\ J^2$	$1.6954 \cdot 10^{-29}\ J^2$	$170.48 \cdot 10^{-29}\ J^2$
$Std(E_{NAND})$	$474\ fJ$	$4.12\ fJ$	$41.3\ fJ$

Variance and standard deviation of the energy consumption of the MDPL

NAND cell for balanced complementary wires are shown in Table 6.3. The variance of the energy consumption of the MDPL NAND cell decreases approximately by two orders of magnitude compared to the CMOS NAND cell and is around two orders of magnitude higher than that of the SABL NAND cell. The values for the CMOS and the SABL NAND cells have been copied from Table 6.2 to allow for an easy comparison. The values in Table 6.3 clearly show that the PA resistance of an MDPL NAND cell is lower than that of the DRP NAND cells for balanced complementary wires. However, in contrast to the DRP NAND cells, the DPA resistance of the MDPL NAND cell does not depend on the balancing of the complementary output wires in the used evaluation model. This was initially (i.e. before the discovery of the timing problem, see next section) thought to be the major advantage of MDPL.

Figure 6.4 shows how the variance of the energy consumption of an MDPL NAND cell and an SABL NAND cell depends on the balancing of the complementary output wires of the cells. The result for the SABL NAND cell has been taken from Figure 6.2. While the PA resistance of the SABL NAND cell decreases quadratically with the increasing difference of the capacitances at the complementary outputs, the PA resistance of the MDPL NAND cell is more or less independent of the size of the difference. In this particular case, if the difference of the capacitances is approximately 10 fF or more, the PA resistance of the MDPL NAND cell is higher than the one of the SABL NAND cell. More basic experiments showing PA-resistance simulation results of MDPL circuits are presented in [PM05] and in [PM06].

6.2 The Early Propagation Problem

As extensively discussed in the last section, the PA resistance of MDPL is not decreased by unbalanced complementary wires as long as timing variations in the instantaneous power consumption are not considered. It was mentioned that a rather common analysis model which considers only the energy consumption per clock cycle introduces such a simplification. However, data-dependent timing variations of the power consumption are very common in modern CMOS circuits. Therefore, such a simplification is not acceptable in PA-resistance evaluations.

Besides unbalanced complementary wires in dual-rail circuits, the main cause of data-dependent timing variations of the power consumption of digital circuits is an effect called *early propagation*. Early propagation occurs in all combinational circuits implemented with cells whose input values directly and immediately influence the output values. The combinational cells of the nowadays predominate CMOS logic style (see Section 2.3) show this effect. This "directly" also implies that a cell produces an output value as soon as it is unambiguously defined by some or all input values, e.g. the output of a CMOS NAND cell evaluates to 1 if only one input is 0—the actual state of all other inputs is irrelevant. In other words, early propagation describes the effect that combinational cells evaluate prematurely, i.e. before all inputs have reached valid new values. In combination with delay differences of the input signals, early propagation leads

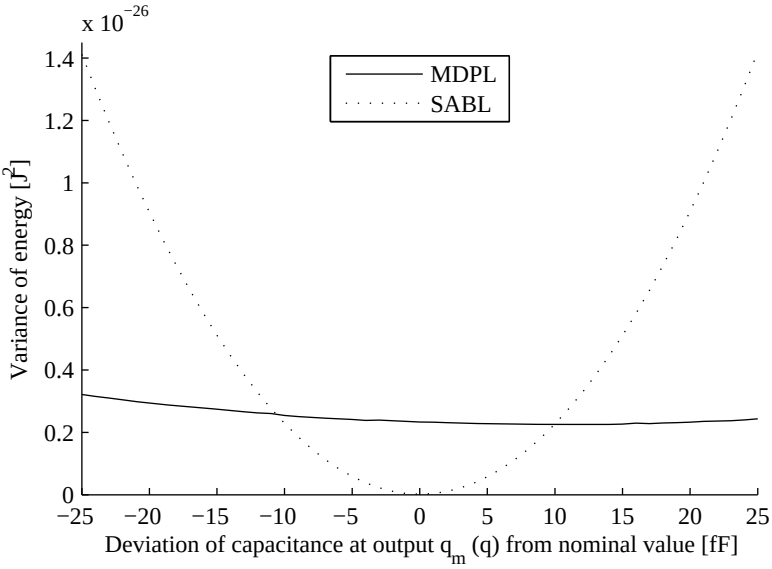


Figure 6.4: Variance of the energy consumption of an MDPL NAND cell and an SABL NAND cell as a function of the difference of the capacitances at the complementary cell outputs q_m (q) and $\overline{q_m}$ ($\overline{q}$). The nominal capacitance at the outputs is 100 fF.

to data-dependent timing variations of the power consumption.

An interesting fact to notice is that combinational cells with early propagation are basically a necessary condition for the occurrence of glitches in combinational circuits. This is the main difference between early propagation and glitches: it is possible to have early propagation but no glitches in a circuit (e.g. see the discussion for MDPL in the following), but to have glitches without early propagation is usually not possible (if we neglect the fact that glitches might even also be produced internally in a cell). A further necessary condition for glitches is that there are actually delay differences in the input signals of a combinational cell (if we again neglect the fact that glitches might also be produced internally). However, due to the usually different parasitic loads of signal wires and the fact that the input signals typically originate from paths with different logic depth, this condition is in most cases fulfilled.

Early propagation has already been mentioned in various publications for its disadvantageous impact on hiding logic styles aiming at a constant instantaneous power consumption, e.g. [GHM+04]. However, this problem has often simply been overlooked or its impact has been strongly underestimated. For example, the original version of an SABL NAND cell [TAV02] has an early propagation problem as pointed out in [Sch03] and [KKT06b]. A rather simple modification of the SABL NAND cell (two additional transistors) as described in [Sch03], [TV04a], and [TV05] solves this problem for SABL. Early propagation also oc-

curs for the combinational WDDL cells [SS06]. Unfortunately, a low-overhead solution for WDDL is not so easy to find as in the case of SABL.

For masking logic styles, it was not obvious in the beginning that early propagation can be a significant problem as well. However, Suzuki and Saeki additionally showed in [SS06] that early propagation also negatively influences the PA resistance of MDPL. They verified their theoretical findings with measurements of MDPL implementations on an FPGA. In parallel to these investigations, we produced a prototype chip that includes an MDPL circuit. With this chip it was possible to verify the theoretical and FPGA-based findings for MDPL also on an ASIC. The results of the prototype evaluation are presented in the next section.

6.2.1 Evaluation of MDPL on a Prototype Chip

In order to confirm the results of [SS06] for ASIC implementations, we analyzed an 8051-compatible microcontroller core that has been implemented in the logic styles CMOS (for reference) and MDPL. The comparison of the analysis results for the implementations shows that the MDPL microcontroller core can be attacked almost as easily as the CMOS core due to the early propagation effect. These findings have been published in [PKZM07]. Also two master theses have been completed in the course of analyzing the prototype chip. In the basic evaluation presented in [Zef07], it was discovered that the MDPL core has a general problem with regard to its PA resistance. The more detailed investigation presented in [Kir07] sought the origin of the problem, which finally turned out to be principally the early propagation effect.

The prototype chip has been implemented in a $0.13\,\mu m$ process technology. The general architecture of the chip is explained in more detail in the next chapter in Section 7.2 together with its successor chip, which has a very similar structure. The cell netlist of the CMOS and the MDPL core in the prototype chip is essentially the same, only the implementations of the primitive cells were done in the respective logic style. An on-chip PRNG produces the mask values for the MDPL core.

DPA Attack Results

The two logic styles have been analyzed by attacking the 8051 microcontroller of the respective core while it performs an internal MOV operation, i.e. one byte of data is moved from one internal random-access memory (IRAM) register to another one. The value in the destination register has been set to 0 before this operation. In the DPA attack, the Hamming weight (HW) of the moved byte has been used as the predicted power consumption. In the given scenario, the HW of the moved byte equals the number of bit transitions at the destination register. Besides this leakage model, the linear Pearson correlation coefficient has been used in the DPA attack to quantify the relationship between the predicted and the measured power consumption [BCO04].

The measurement setup that has been used to record the power consumption of the prototype chip consisted of three main parts: a board that holds the prototype chip and necessary external devices like power regulators and the program read-only memory (ROM), a digital oscilloscope, and a host PC that controls both the oscilloscope and the prototype chip on the board. The bandwidth of the oscilloscope has been $1\,GHz$. A suitable differential probe has been used to measure the power consumption via a $10\,\Omega$ measurement resistor in the V_{DD} line of the prototype chip.

A first investigation of the measured power traces revealed the presence of significant disturbances within some traces, which have a negative effect on the DPA attack. Highly disturbed traces have been identified by calculating the "sum of squared differences" of each trace and the mean trace of a set of measurements: first, the difference between a trace and the mean trace was calculated point-wise; these difference values were then squared and summed up. Traces for which this sum exceeded a specific threshold were considered as highly disturbed and were filtered out.

The clock frequency provided to the prototype chip has been $3.686\,MHz$. The relevant settings of the digital oscilloscope have been the same in the measurement runs for the two cores:

- **Vertical resolution:** $39\,mV/Div$
- **Input coupling:** $1\,M\Omega - AC$
- **Horizontal resolution:** $0.2\,\mu s/Div$
- **Sampling rate:** $4\,GS/s$
- **Points per power trace:** 8 000

Figure 6.5 (left) shows the result of the DPA attack for the MOV operation on the CMOS core. The correlation trace when using the correct data bytes to generate the power hypothesis is plotted in black. Additionally, 10 correlation traces are plotted in gray for which random data values have been used to generate the power hypotheses in the DPA attack. As expected, a rather high maximum correlation coefficient of 0.3068 occurs for the correct power hypothesis in the clock cycles where the MOV operation is executed. The first correlation peak occurs when the moved byte is fetched from the source register via the internal bus to the destination register. The second peak occurs when the moved byte is stored in the destination register and removed from the internal bus. In the 10 correlation traces for random data values, no significant correlation values occur.

The correlation trace for the MDPL core depicted in Figure 6.5 (right) shows a significant leakage in the MOV operation. As we will show in the next section, this leakage is mainly caused by the early propagation effect. The highest correlation peak of 0.2385 lies in the range of that one of the CMOS core. Note that the MDPL core has been operated with activated PRNG. The MDPL core precharges when the clock signal is 1 and evaluates when the clock signal is 0.

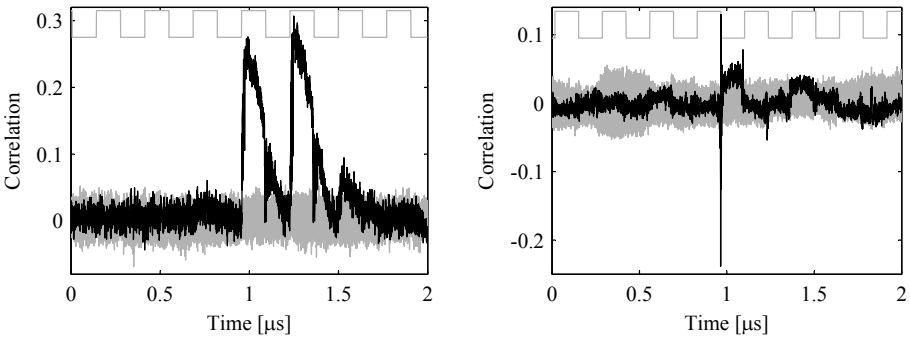


Figure 6.5: Results of the DPA attacks on the CMOS reference core (left, 5 000 samples) and the MDPL core (right, 5 000 samples): internal MOV operation in the IRAM; the correlation trace for the correct power hypothesis is plotted in black. The related clock signal is indicated in the upper part of the figures.

In Table 6.4, the results of the DPA attacks on the measured power traces of the prototype chip are summarized. The formula to calculate the number of required power traces for a successful attack from the highest correlation value is given in Section 2.2.

Table 6.4: Results of the DPA attacks on the measured power traces of the prototype chip, internal MOV operation

Logic style	Number of used power traces	Highest absolute correlation peak	Number of required power traces
CMOS	5000	0.3068	279
MDPL	5000	0.2385	471

Problem Analysis Based on Transistor-Level Simulations

In this section and the next one, the origin of the leakage of the IRAM MOV operation on the MDPL core is analyzed in detail. As shown by Suzuki and Saeki [SS06], MDPL cells may leak information due to timing differences in the input signals and the early propagation effect, which is not prevented in such cells. Suzuki and Saeki verified their theoretical results by measurements on an FPGA. We show that these effects are most probably also the cause for the DPA leakage in the MDPL core of the prototype chip.

In a first step of the problem analysis, the cells that are directly involved in the MOV operation have been analyzed with the help of transistor-level simulations. These simulations have been carried out with NanoSim from Synopsys. The transistor netlist of the MDPL core (excluding interconnect parasitics to improve simulation speed) has been simulated for two cases: moving the value $0x00$ and moving the value $0xFF$ in the IRAM for different mask values. The power

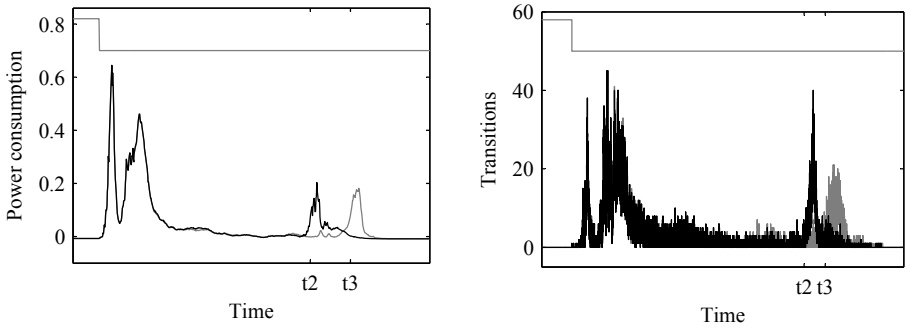


Figure 6.6: Power consumption of the MDPL core in a clock cycle of the MOV operation when moving the value $0x00$ (black) and $0xFF$ (gray), the mask is kept 0. Left: transistor-level simulation without interconnect parasitics. Right: transition count at each point in time based on logic simulations including extracted delay information.

consumption in the clock cycle of the MOV operation where the significant correlation peak (according to Figure 6.5 - right) occurs is shown in Figure 6.6 (left) for mask 0. The first two peaks of the power consumption, which are identical for the values $0x00$ and $0xFF$, occur right after the negative clock edge (start of the evaluation phase of MDPL). For the third peak of the power consumption, the time offset $t_3 - t_2$ for the two data values is clearly visible. The time offset is in the range of $1\,ns$. The NanoSim simulations for random mask values have shown that this timing difference is independent of the actual value of the mask. Thus, a correlation between the unmasked values and the power consumption occurs in the DPA attack on the MDPL core even with activated PRNG.

Next, the reason for this mask-independent time offset has been analyzed. In the simulation results, an MDPL-AND cell has been identified which switches at the beginning of the time period where the correlation peak occurs. Furthermore, the outputs of this MDPL-AND cell switch with a time difference of approximately $1\,ns$ for the two moved values in the transition from precharge phase to evaluation phase. The transistor-level simulations have also shown that the difference between the arrival times of the input signals A, B, and M of this cell is significantly larger than the propagation delay of the MDPL-AND cell, which consists of two majority (MAJ) cells (see Figure 5.3). The input signal A depends on the moved value and signal B is constantly 0. The situation is depicted in Figure 6.7.

The timing conditions for the inputs of the MDPL-AND cell are as follows: the signals M, $\overline{M}$ arrive first (time t_1), then A_M, $\overline{A_M}$ arrive (time t_2), and at last B_M, $\overline{B_M}$ arrive (time t_3). The mask signals arrive first because they are provided by a so-called mask unit (see Figure 7.6) right at the beginning of the evaluation phase and they do not need to go through combinational logic. The delay of the signals B_M, $\overline{B_M}$ is longer than that of the signals A_M, $\overline{A_M}$ because of a higher number of cells in the respective combinational paths.

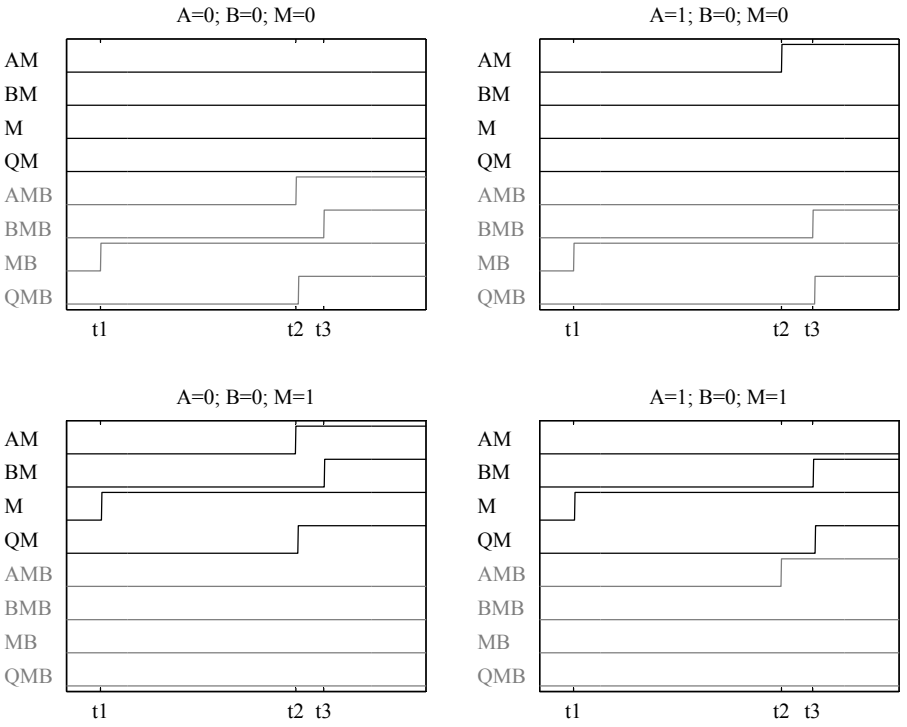


Figure 6.7: Signals of the MDPL-AND majority cells for which early propagation occurs (transistor-level simulation, black: signals of first MAJ cell, gray: signals of second MAJ cell). Signal A depends on the moved value. Signal B is constantly 0.

In the given situation, it turns out that for $A = 0$, always one majority cell switches at time t_2 (neglecting the propagation delay of the majority cell). A different mask value only changes the affected majority cell within the MDPL-AND. For $A = 1$, the majority cells always switch at time t_3 (again neglecting the propagation delay).

These results clearly show that early propagation causes the dependency between the unmasked data values and the evaluation moment of the MDPL-AND cell. In [SS06], the authors show the occurrence of leakage due to early propagation for a more general case, i.e. when the value of B is also variable. Only one cell that shows this behavior would most probably not cause such a significant correlation peak in the DPA attack on the entire chip. However, further investigations have shown that the discussed early propagation effect also occurs for the other seven bits of the moved data value and there are several other MDPL-AND cells which behave in the same way. Furthermore, the outputs of the affected cells are fed into many other MDPL cells before the data values are eventually stored in registers. Thus, also these cells are affected by the data-dependent moment of evaluation. Altogether, there are hundreds of MDPL cells

which evaluate in a data-dependent manner.

Preventing early propagation would mean that the MDPL-AND cell only evaluates when all input signals have arrived, i.e. all input signals have been set to differential values. Thus, in both cases ($A = 0$ and $A = 1$), such an improved MDPL cell would always evaluate at time t_3. The DPA leakage caused by the data-dependent evaluation moments of the MDPL-AND cell would be prevented. A proposal how to avoid early propagation for MDPL is presented in Chapter 7.

Problem Analysis Based on Logic Simulations and Transition Counts

In a last step of the problem analysis, the correlation results based on measured power traces have been reproduced by attacking simulated power traces. Transistor-level simulations were not suitable for this purpose because it would have taken too long to simulate an appropriate amount of power traces for such a big circuit as the analyzed one. Therefore, logic simulations including extracted delay information have been performed. From these results, a basic power trace has been generated by counting the number of transitions at each moment in time. Figure 6.6 shows that the result of such a simulation (see figure on the right) looks quite similar to the transistor-level simulation result (see figure on the left).

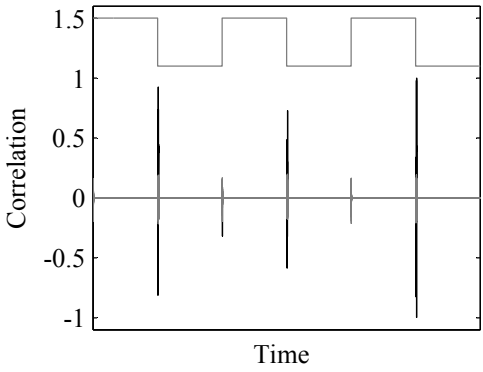


Figure 6.8: Result of the DPA attack on the MDPL core using simulated power traces: transition count based on logic simulation of internal MOV operation in the IRAM; 256 samples; the correlation trace for the correct power hypothesis is plotted in black. The clock signal is indicated at the top.

Logic simulations of the MOV operation in the MDPL core have been performed for the 256 different values of the moved byte and random mask values. A subsequent DPA attack on the simulated power traces derived from the logic simulations has led to the results shown in Figure 6.8. Correlation traces for wrong power hypotheses are plotted in gray while the correlation trace for the correct power hypothesis is plotted in black. The correlation peak in the third

clock cycle (a clock cycle starts with the positive signal edge) corresponds to the highest correlation peak shown in Figure 6.5 (right). It is also the point in time that is shown in detail in Figure 6.6. The correlation peaks in the first and second clock cycle do not appear in the DPA attack on the measured power traces. A detailed analysis has shown that these correlations are caused by very small data-dependent variations in the power consumption, which can only be exploited in the attacks based on simulations. These small data-dependent variations most probably occur because the data value that is moved is already stored in the source register before the actual MOV operation takes place. The improved version of MDPL that is presented in the next chapter is capable of removing all these correlation peaks in a DPA attack based on power traces derived from logic simulations.

6.2.2 Simulated DPA-Attack on the Demo Circuit "AES-encinit" Implemented in MDPL

In order to further investigate the particular influence of early propagation on MDPL circuits, we have also used the power trace estimation technique based on logic simulation for the demo design "AESencinit". The power simulation technique has been presented in detail in Section 3.2. The specific tool that was used to map logic simulation results to power traces is called "vcd_analyzer" and is described in detail in Section 3.3. The demo design "AESencinit" is described in detail in Section 3.3.1.

The implementation of the demo circuit in MDPL, the power simulation with vcd_analyzer, and the DPA attack have been performed similarly as described for the mCMOS implementation of the demo circuit in Section 4.4.2. The used terminology to denote the various simulation and analysis results is explained in detail in Section 3.3.1. The main difference of the mCMOS and the MDPL implementation of the demo circuit is that after synthesis, the CMOS circuit has been translated to an MDPL circuit. The **comb**-part of the MDPL demo circuit was simulated for all $256^2 * 4$ possible input value transitions (8-bit data input; 1-bit mask input).

Attack Results

The logic simulation of the MDPL implementation of the AESencinit demo circuit has been performed with a clock cycle time of $120\,ns$. The 8-bit secret key used in the simulation has been 165 (0xA5). In our detailed analysis of MDPL, we concentrated on the early propagation effect. Therefore, we only simulated and attacked the combinational part of the demo circuit.

The attack result **comb/unitdelay/equal/cc/SBin/ZV** (glitches, no early propagation) showed a DPA peak for the mCMOS implementation (see Section 4.4.2). This result demonstrated that mCMOS has a problem with glitches in the masked circuit. Looking at the corresponding result for the MDPL circuit, the attack **comb/unitdelay/equal/cc/SBin/ZV** no longer leads to a

DPA peak. Since glitches do not occur in MDPL circuits by design, they cannot cause a DPA leakage as in case of mCMOS.

If also the effects of early propagation are included in the power simulation of the MDPL demo circuit, the situation changes. Figure 6.9 shows the average power traces for different input values and random mask bits. The ZV-effect can clearly be seen: If the input value of the SubBytes block is 0, the calculation finishes much faster. Note that in the picture, the power peaks at the beginning of the precharge phase (60 ns to 119 ns) have been clipped. The highest power peak showed 253 toggles.

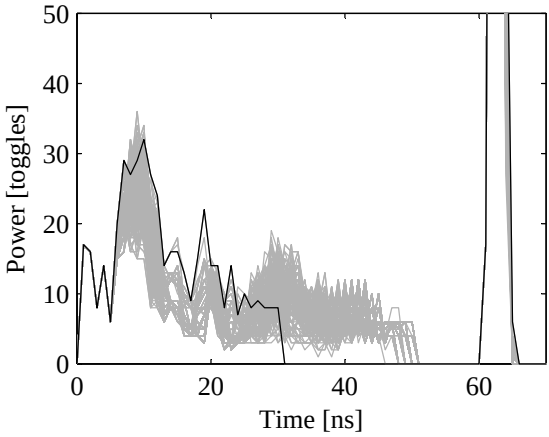


Figure 6.9: Average power traces for the simulation **comb/unitdelay/equal/tr** for different input values of the comb-part and random mask bits; black curve: input of comb-part is 165 ⇒ input of SubBytes block is 0 (unmasked).

The result of the successful attack **comb/unitdelay/equal/tr/SBin/ZV** is shown in Figure 6.10. In the evaluation phase (0 ns to 59 ns), a significant correlation peak for the correct key guess is achieved in a rather wide range around 40 ns. This corresponds to the power simulation result shown in Figure 6.9. A smaller but still visible DPA peak also occurs around 20 ns. This indicates another ZV-related leakage at this point in time. In the precharge phase there is another small, negative correlation peak.

The above results again clearly demonstrate the "early propagation problem" of combinational MDPL circuits. The power simulations for varying input data and masks always reported 924 toggles, equally divided in the precharge and the evaluation phase. The problem is thus not how many signal transitions occur but at which point in time they occur. The variance of the latter leads to successful ZV-based DPA attacks.

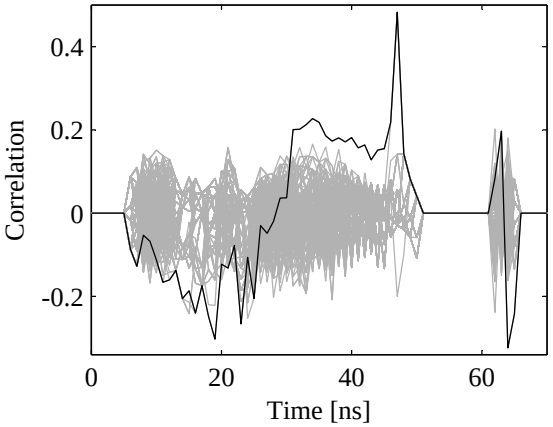


Figure 6.10: DPA attack result **comb/unitdelay/equal/tr/SBin/ZV**. The black curve shows the correlation trace for key guess = 165.

6.3 A "Secure" MDPL AES Module Despite Early Propagation

The analysis of the microcontroller in the MDPL prototype chip presented in the last section clearly showed that for the large signal-delay differences in the range of a nanosecond that occur in the controller, no increase in the PA resistance could be achieved. Still an open question in this context was how this effect scales in a circuit with significantly smaller delay differences.

In the following we show how we analyzed this issue with the help of an AES coprocessor that is attached to the microcontroller on the prototype chip. This AES module is based on a very regular architecture, which is explained in more detail in the next chapter in Section 7.2. Thus, the occurring signal-delay differences are much smaller than in the microcontroller module itself. This has been verified by analyzing logic simulations that included back-annotated signal delays. The attack results presented below indicate that MDPL might not be broken in all circumstances. So far, we have not been able to mount a successful standard, first-order DPA attack on the AES coprocessor. These findings have been published in [PKM09].

6.3.1 DPA Scenario Description

The target of the DPA attack are the movements of the SubBytes results of the first AES round through the State registers 1 to 16. According to the architecture shown in Figure 7.9, these values are moved in the AES State from top to bottom during the SubBytes operation and from right to left during the subsequent MixColumns operation. Each of the SubBytes results depends exactly on one byte of the unknown secret key, thus there are 256 possibilities

for each of them.

As an example, we track the movements of the plaintext byte that is initially loaded into the AES State cell 4. In the first clock cycle, the initial AddRoundKey operation with round key byte 4 is performed by the State cell. The result enters the first stage of the S-box until the intermediate result is stored in the pipeline registers. In the next clock cycle, the second stage of the S-box is performed and the substitution result is stored in State cell 1. Since the value we track belongs to the fourth row of the AES State, the byte is rotated three positions to the left in the third clock cycle. Thus, it gets stored in State cell 6. In the following two clock cycles, the byte moves down through cell 7 into cell 8, which is its final position after SubBytes and ShiftRows. In the next clock cycles, the byte is moved to the left because the MixColumns operation is performed.

During its way through the AES State, a SubBytes result overwrites other SubBytes results. In case of the attacks on the CMOS core, the Hamming distance of such two values is used as the power model in the DPA attack. Note that with our approach, we favor the attacker in the following way. We assume that the secret key byte which is necessary to calculate the overwritten SubBytes result is known. In doing so, we keep the number of possibilities for the secret key in one DPA attack at 256. In the attacks on the MDPL core, we only have to predict the Hamming weight of the moved SubBytes result. The reason is that before a byte is transferred over a bus, all bus lines get precharged to 0. Furthermore, since MDPL is a dual-rail logic style with more or less randomly imbalanced wire pairs, we can generally only use one bit at a time of the 8-bit SubBytes result in a meaningful leakage model. In the following, we call this the HWbit power model.

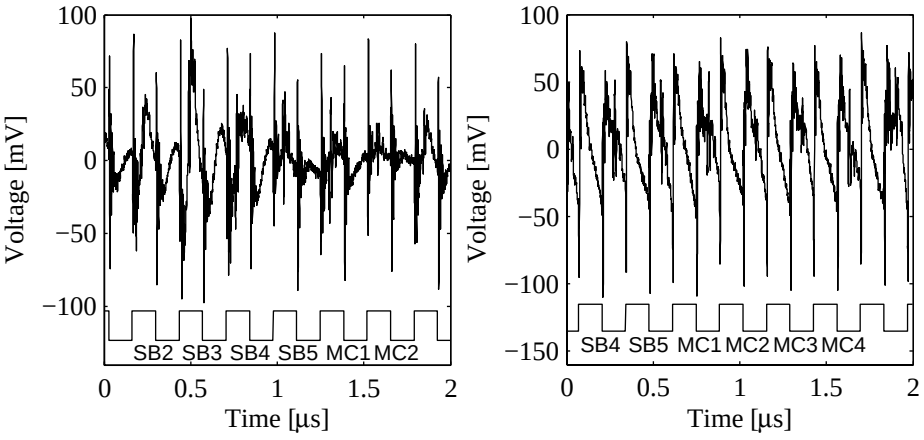


Figure 6.11: Power trace and clock signal during the execution of the attacked AES operations for the CMOS core (left) and the MDPL core (right).

Figure 6.11 shows the power trace and the clock signal of the part of the first AES round we have measured for the DPA attacks. An identifier beneath each

clock cycle indicates the step of the operation that is performed (SB = SubBytes; MC = MixColumns). A trigger signal was generated by the attacked device to easily locate these important clock cycles. The assumption that one knows when the power consumption needs to be measured also favors the attacker significantly, because the amount of data and the measurement and analysis time are in this case substantially reduced. The measurement setup was the same as that one used in the attack on the MDPL microcontroller (see Section 6.2.1 for details).

6.3.2 DPA Attack Results

In the DPA attacks, the linear Pearson correlation coefficient was used to quantify the dependency between measured power consumption and predicted power consumption [CKN01]. The height of the correlation peak in a successful attack is used to calculate the minimum number of needed power traces to get a distinct peak. For correlation peaks $\rho_{ck,ct} \leq 0.2$, the minimum number of needed power traces n is approximated as follows: $n = 28/\rho^2_{ck,ct}$ ($confidence = 99.99\,\%$, ck ... correct key, ct ... point in time where the correlation peak occurs) [MOP07]. The number n is used to quantify the DPA resistance of the attacked device for the given attack scenario.

DPA Attack on the CMOS Reference Core

The DPA attacks on the hardware AES in the CMOS reference module showed that the leakage (i.e. the height of the resulting correlation peak) of the different key bytes significantly depends on the path each key byte takes through the AES State matrix (see Figure 7.9). It turned out that State cell 4 leaks the most information because its output is connected to an S-box input, a MixColumns input and the AES module output bus. The State cells 1 to 3 leak the second most and cells 8, 12, and 16 leak the third most. The least leakage occurs for all other cells because their output is only connected to their neighboring cells to the left (circuitry for a State shift to the left) and their neighboring cells below (circuitry for a State shift to the bottom).

640 000 traces have been measured to attack the CMOS AES hardware module. As mentioned in Section 6.3.1, the HD power model has been used to map the byte transitions in the AES State cells to power consumption values. All key bytes have been successfully attacked. The lowest correlation peak value of 0.01 has been achieved for key byte 13 using the byte transition $14 \rightarrow 13$ (i.e. data related to key byte 14 is overwritten by data related to key byte 13). This byte transition only occurs in State cell 13, which has a very low leakage. A correlation peak value of 0.01 maps to a number of needed traces n of more than 276 000. This rather high number is caused by the small output load of the involved State cell 13 and the high amount of noise caused by the microcontroller working in parallel. The highest correlation peak value of 0.0382 has been achieved for key byte 8 using the byte transition $4 \rightarrow 8$. This byte transition occurs in State cell 4, which is the one with the highest leakage. A correlation

peak value of 0.0382 maps to a number of needed traces n of about 19 200. The transition also occurs one clock cycle earlier in State cell 8, which has a lower leakage (see Figure 6.12, left). For the transition in State cell 8, a correlation peak value of 0.02 has been achieved.

Figure 6.12 (left) shows the correlation traces for the attack on key byte 8. The correlation trace for the correct key guess is plotted in black. The clock cycles when the attacked byte transition occurs in State cell 8 (lower correlation peak) and one clock cycle later in State cell 4 (higher correlation peak) can clearly be seen. The time axis of the left figure matches that of Figure 6.11 (left).

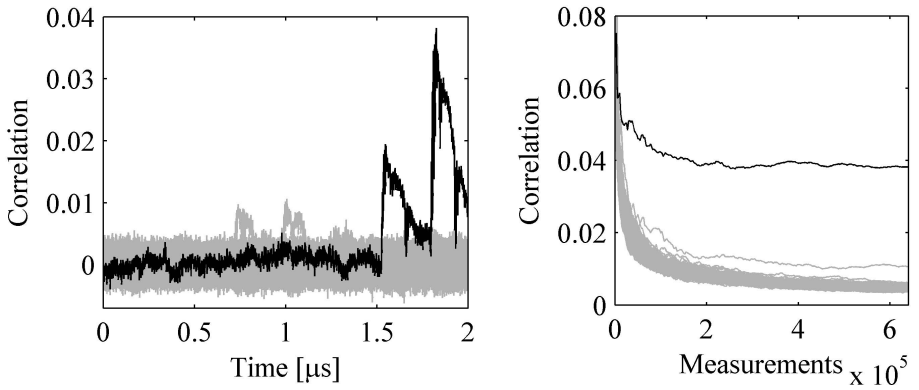


Figure 6.12: CMOS AES hardware module: 256 correlation traces for key byte 8 (left, 640 000 measurements, HD power model) and evolution of the maxima of the correlation traces over measurements (right); traces for the correct key are plotted in black.

DPA Attack on the MDPL Core with Deactivated PRNG

In order to get some experience with the MDPL AES hardware module, we first attacked it with deactivated PRNG. This gave us the chance to see if our attack scripts are correct and that we used the correct part of the power trace in the attack. We were able to attack more than half of the key bytes with the number of measurements we performed.

The DPA results in case of the MDPL AES module look quite different from those of the CMOS AES module. For MDPL, we could not clearly identify State cells with a higher and State cells with a lower leakage. Furthermore, the findings concerning the amount of leakage of the AES State cells for the CMOS core could not be reproduced for the MDPL core. The reason for it is that in a dual-rail logic style like MDPL, the amount of leakage of a cell does not depend on the absolute load connected to a cell's output but on how well the dual-rail output wires are balanced. Since no explicit balancing was done in the MDPL circuit, the actual balancing situation in the placed and routed circuit is more or less random.

For the DPA attack on the MDPL AES with deactivated PRNG we measured 1 256 000 power traces. With the arguments from the last paragraph and from Section 6.3.1 in mind, one would typically assume that the HWbit power model is the best choice for this attack. However, our analysis showed that with the HWbit model, we could find only 1 key byte (key byte 12; bit 1 used in attack; resulting correlation peak $\rho_{ck,ct,12} = 0.0056$). Much better results were yielded by using the HD model and the best results were achieved with the HW model. The reason for this behavior most likely lies in the architecture of the MDPL flip-flops (see Figure 5.5). They store the masked data in a single-rail flip-flop, which is connected to only one part of the complementary network of the input signal. This leads to a more or less uniform direction of the imbalance of the dual-rail signals going into an MDPL flip-flop. Therefore, the HW power model leads to better results than the HWbit model in cases where such signals are attacked.

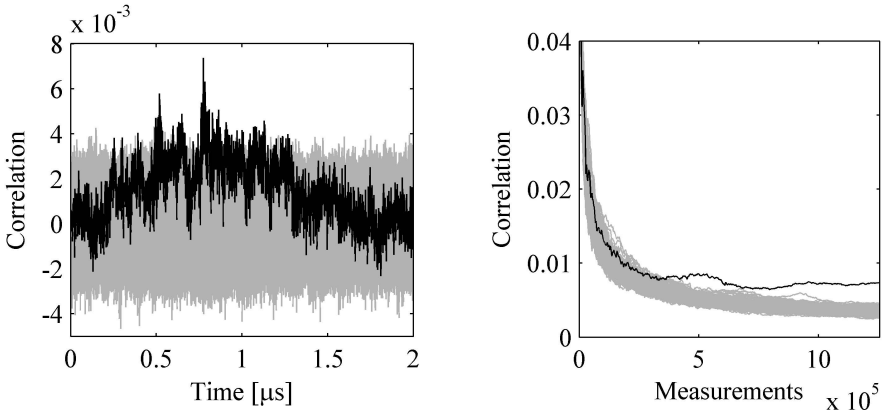


Figure 6.13: MDPL AES hardware module with deactivated PRNG: 256 correlation traces for key byte 14 (left, 1 256 000 measurements, HW power model) and evolution of the maxima of the correlation traces over measurements (right); traces for the correct key are plotted in black.

When using the HD power model as in the DPA attack on the CMOS AES module, the highest correlation peak we got had a value of 0.0054, which maps to a number of required traces n of around 960 000. We could unambiguously identify 6 key bytes. In case of the HW power model, we got more and higher correlation peaks for the different key bytes (due to the precharging, which also happens in the input stages of MDPL flip-flops). However, we could not find all key bytes with the 1 256 000 measured power traces. The four highest correlation peaks occurred for key bytes 5 ($\rho_{ck,ct,5} = 0.0067$), 13 ($\rho_{ck,ct,13} = 0.0061$), 14 ($\rho_{ck,ct,14} = 0.0074$), and 15 ($\rho_{ck,ct,15} = 0.0061$). This maps to a number of needed power traces of approximately: $n_5 = 624\,000$, $n_{13} = n_{15} = 752\,000$, and $n_{14} = 511\,000$. More than half of the key bytes could be successfully attacked, for the others there were either ambiguous peaks for incorrect key guesses or no

peaks at all in the correlation traces. Figure 6.13 (left) shows the correlation traces for the HW attack on key byte 14. The correlation trace for the correct key guess is plotted in black and the time axis matches the one of Figure 6.11 (right). In Figure 6.13 (right), the evolution of the maxima of the correlation traces over the measurements can be seen. It clearly shows that the number of needed power traces $n_{14} = 511\,000$, which has been calculated above, is reasonable.

DPA Attack on the MDPL Core with Activated PRNG

Due to the findings described in the last section, we attacked the MDPL core with activated PRNG also by using the power models HWbit, HD, and HW. The following behavior was observed: No at least "almost" or "weak" correlation peaks could be achieved with the HWbit and the HD power model. With the HW power model, we were also not able to identify a single key byte unambiguously. However, there were 4 key bytes (9, 10, 13, 14) for which one could get the impression that not too many more measurements would be necessary to get a distinct correlation peak. But this is only a speculation.

For the attack on the MDPL core with activated PRNG, we measured three sets of power traces to improve our statistical sampling accuracy. For the biggest set, we measured 3 511 000 power traces, because we expected a significant increase in the number of needed measurements due to the random mask that is provided to the MDPL AES core.

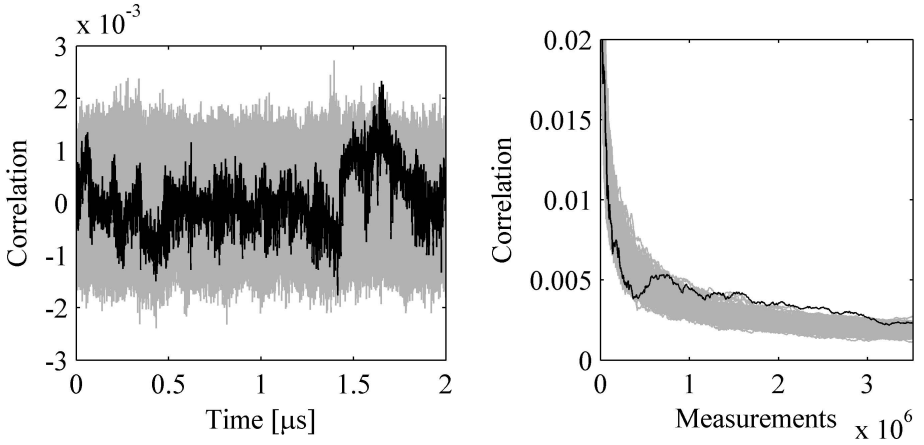


Figure 6.14: MDPL AES hardware module with activated PRNG: 256 correlation traces for key byte 14 (left, 3 511 000 measurements, HW power model) and evolution of the maxima of the correlation traces over measurements (right); traces for the correct key are plotted in black.

Figure 6.14 shows the correlation results for key byte 14 (HW attack), which showed the highest correlation peak in the HW attack on the MDPL core with deactivated PRNG. One can clearly see that the maxima of the correlation trace over measurements for the correct key (right figure, black trace) stay for long

periods at the top border of the gray correlation-maxima band. However, its value always declines with more measurements, which is not the case in a successful attack (see Figures 6.12 and 6.13). The correlation traces for the other three suspicious key bytes showed a similar behavior. Therefore, we regard the MDPL AES module in the prototype chip (with activated PRNG) as secure for at least 3 511 000 measurements under the given attack scenario, i.e. if the attacker applies a standard, first-order DPA attack.

6.4 Other Issues of MDPL

Also some more general vulnerabilities of masked logic styles have been discussed lately. The authors of [TS07] and [ST07] presented an attack which aims at undoing the effect of masking in a clock cycle by analyzing the mean of the power consumption during that clock cycle. With this method, which is called "PDF-attack" (PDF ... probability density function) or "folding attack", it should be possible to remove the effects of the mask completely or at least to introduce a significant bias in the mask. With the help of the prototype chip, we analyzed the PDF-attack in practice. It turned out that, at least for the settings we chose, the PDF-attack does not work offhand, i.e. no direct improvements in the DPA attack results could be achieved. We published these practical results for the straight-forward application of the PDF-attack in [PKM09]. The results are presented in the following section.

The very recent publications [MGPV09a] and [MGPV09b], in which the authors investigated MDPL and the PDF-attack in more detail, indicate that by further tweaking the attack, the AES coprocessor of the MDPL prototype chip can be successfully attacked in the end. However, they also stated that applying the PDF-attack as it is originally proposed (analyzing the power consumption over a full clock cycle, "folding" the power values for different mask values) apparently does not lead to successful PA attacks in practice.

Another issue we discussed in [PKM09] is the question whether MDPL has special requirements for the generation of the mask bits or not. This question has been raised in [Gie07]. We elaborated on that issue in a theoretical manner and came to the conclusion that mask generation is not particularly difficult for MDPL, i.e. as for all masked logic styles, there must not be a statistical bias in the values of the masks for MDPL circuits.

6.4.1 The PDF-Attack in Practice

In [TS07] and [ST07], the authors present a new attack methodology against masked logic styles like MDPL. In an experiment with the MDPL core on our prototype chip, we tested whether we can practically reproduce this PDF-attack. The outcome of the experiment was that we could not mount a successful attack on the MDPL core in our environment.

In Section 6.3.2, we described that key byte 14 leaks the most information in an attack on the AES hardware module in the MDPL core when the PRNG is

deactivated. In case of an activated PRNG, the correlation result for key byte 14 was ambiguous. In order to test the effectiveness of the PDF-attack, we chose the following approach. If the PDF-attack works (i.e. if the mask can be biased), it should be possible to turn the ambiguous result for key byte 14 (DPA attack with activated PRNG) into an unambiguous result.

The first step in the PDF-attack was to build power profiles of the clock cycles where we expect a significant leakage and where we thus want to bias the mask. We selected two clock cycles for this purpose. The first clock cycle was the one where the correlation peak occurred in the attack on key byte 14 on the MDPL core with deactivated PRNG: clock cycle $MC1$ around $0.75\,\mu s$ (see Figure 6.13 - left - and Figure 6.11 - right). The other clock cycle was the one where there might be the beginning of a correlation peak in the DPA attack on the MDPL core with activated PRNG: clock cycle $MC4$ around $1.6\,\mu s$ (see Figure 6.14 - left - and Figure 6.11 - right).

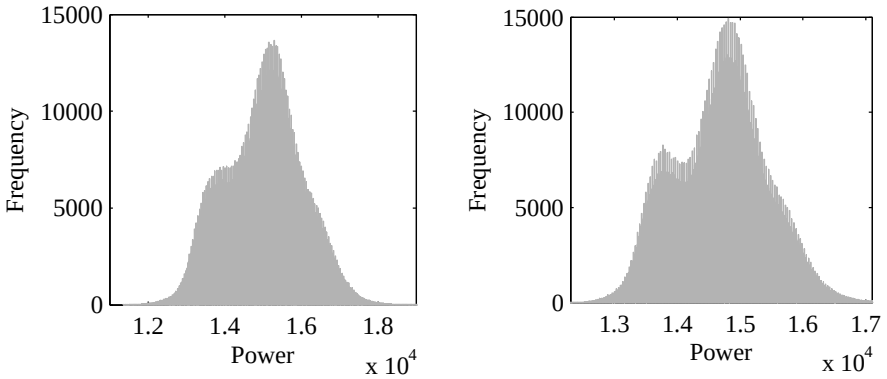


Figure 6.15: Power histograms of the PDF-attack on the MDPL core with activated PRNG: clock cycle $MC1$ (left), clock cycle $MC4$ (right). Each power value in the histogram is the sum of the absolute values of the power consumption in the evaluation phase of the respective clock cycle.

For the power profiles, we took the power values during the evaluation phase of the clock cycles $MC1$ and $MC4$ for each measurement sample of the MDPL core with activated PRNG. Then, we calculated the sum of the absolute values of the selected points of each clock cycle. This was done in accordance to [ST07] where the analysis is based on toggle counts of clock cycles. As a result, we got two power values for every measurement sample, one for clock cycle $MC1$ and one for clock cycle $MC4$. Figure 6.15 shows the two histograms of the two groups of power values. The shapes of the histograms were rather unexpected, because according to the PDF-attack, there should be two equally high peaks (we verified that the PRNG does not have such a significant bias via simulations, which could also cause such a shape). For comparison, Figure 6.16 shows the power histograms for the same clock cycles $MC1$ and $MC4$ in case of a deactivated PRNG. Here, the histograms show single peaks as expected.

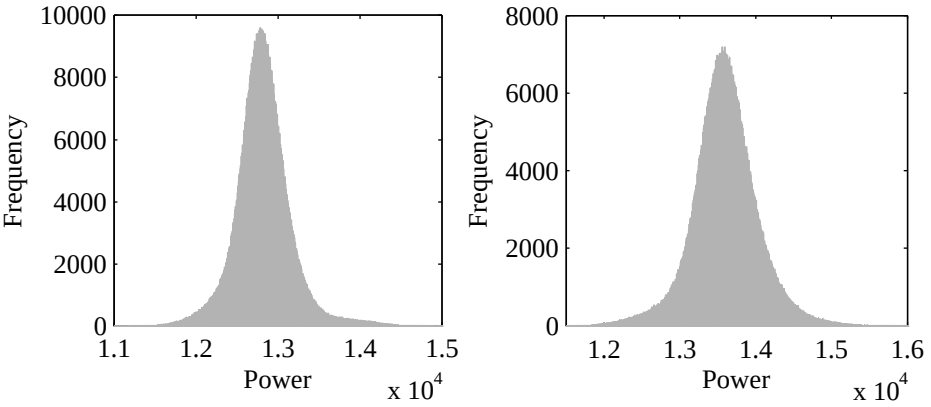


Figure 6.16: Power histograms for the MDPL core with deactivated PRNG: clock cycle $MC1$ (left), clock cycle $MC4$ (right). Each power value in the histogram is the sum of the absolute values of the power consumption in the evaluation phase of the respective clock cycle.

According to the theory of the PDF-attack, it should now be possible to bias the mask by using just the power traces in a DPA attack that have a power consumption in the profiled clock cycles that is above (below) the mean power consumption in the respective clock cycle. Since we were not sure where the different peaks in the power profile histograms originated from, we selected different values as discerning points: the overall mean of each histogram, the mean of each peak in the histograms, and the lowest value in the valley between two peaks of a histogram (we also tried various other clock cycles and in some of them, we got more separated peaks with a valley between them).

The original number of power traces we had at hand for the PDF-attack were 3 511 000. After the selection process according to the chosen discerning point, we always had around 1.5 million traces left for the PDF-attack. According to the needed power traces to successfully attack the MDPL core with deactivated PRNG, this amount of traces should have been enough. However, in none of the settings for the discerning point, we could turn the DPA attack against key byte 14 into a successful one. Finally, we targeted some other key bytes in PDF-attacks and we used the hypotheses for the HD power model instead of the hypotheses for the HW power model. Also these attacks were not successful.

Therefore, we concluded that the PDF-attack as initially proposed does not work as easy in practice as the theoretical results, which were based on power simulations, might suggest. This is also explainable in the following way. With the settings chosen for the PDF-attack in theory (no electronic and other forms of noise, no perfect balancing of dual-rail wire pairs, ...), it can easily be shown that all PA-resistant logic styles that try to achieve a constant power consumption are also completely broken. However, various publications based on practical results obviously draw opposite conclusions, e.g. [HTH+06].

7

Tackling the Early Propagation Problem - iMDPL

Looking at the evaluation results for MDPL in the last chapter, a main conclusion can be drawn very clearly: Masked logic styles of this type which want to resist DPA attacks must avoid the early propagation effect. Otherwise, a power consumption occurs that depends on the unmasked data values due to data-dependent evaluation moments of the combinational cells. In many cases, this data-dependent power consumption almost nullifies the PA-resistance improvement of MDPL.

Therefore, a logical next step in our research was to get rid of the early propagation problem in the MDPL cells. The proposal we finally came up with is called Improved MDPL (iMDPL) [PKZM07]. The main idea is to delay the evaluation of a combinational cell until the last input signal has arrived, i.e. until all input signals have been set to complementary values. iMDPL has very high area requirements but it is still based on standard CMOS cells. Besides iMDPL, also other proposals for masked logic styles that take the issue of early propagation into account have been made, e.g. precharged masked Reed-Muller logic (PMRML) [LFYL07].

In this chapter, we first introduce iMDPL, describe its working principle, and discuss basic evaluation results. In the second section, we describe the architecture of a test chip that contains iMDPL circuits and present practical evaluation results. The results show that iMDPL indeed improves the PA resistance of a circuit, i.e. the early propagation effect is significantly reduced. However, the increase in the number of samples indicated by the test chips when comparing unprotected (CMOS) and protected (iMDPL) circuits is lower than expected, especially when the considerable overhead of iMDPL is taken into account. As

a consequence, iMDPL seems to be practically useable at most in specific, sufficiently small parts of the overall circuit.

7.1 Improved Masked Dual-Rail Precharge Logic (iMDPL)

The primary aim when developing iMDPL was to get rid of the early propagation problem of the combinational MDPL cells while still basing the new logic style on standard CMOS cells that are commonly available. This significant advantage (e.g. when it comes to a change of the underlying process technology) should justify at least to some degree the additional overheads introduced by iMDPL.

The differential encoding of the signals in MDPL circuits allows detecting the point in time in the evaluation phase when all input signals of a cell are in a valid differential state. A cell that avoids early propagation must delay the evaluation moment until this point in time. In [CZ06], the logic style Dual-Rail Random-Switching Logic (DRSL) has been presented, which implements such a behavior in the evaluation phase.

It has also been shown in [SS06] that it is necessary to avoid an early propagation effect in the precharge phase as well. Our DPA attack results of the MDPL microcontroller core presented in the last chapter in Figure 6.5 (right) of Section 6.2.1 confirmed this practically. After the high correlation peak at the beginning of the evaluation phase, there occurs a smaller but still clearly recognizable correlation peak at the beginning of the subsequent precharge phase (around 1.1 μs).

According to our analysis, DRSL does not completely avoid an early propagation effect in the precharge phase. The reason is that the input signals, which arrive at different moments, can still directly precharge the DRSL cells. The propagation delay of the evaluation-precharge detection unit (EPDU) used in the DRSL cells leads to a time frame in which this can happen. Only after that time frame, the EPDU unconditionally precharges the DRSL cells. Our simulations with an intermediate version of an improved MDPL cell (i.e. a version which also did not correctly handle precharging with respect to early propagation) confirmed this - there still occurred correlation peaks in the precharge phase. Thus, the input signals of a cell must be maintained until the EPDU generates the signal to precharge the cell.

In summary, we chose the following behavior for iMDPL: a cell evaluates when the *last* complementary input signal has arrived, and it is precharged when the *first* input signal has switched to the precharge state. It is important to notice that with this approach, only the delay differences between different pairs of complementary signal wires can be equalized. The much smaller delay differences within a complementary wire pair might still influence the time of evaluation or precharging of an iMDPL cell. As noted in [SS08] "there is a possibility of leakage if a sufficient difference in delays exists". It is an open question whether the delay differences in iMDPL due to the unbalanced routing

approach are usually sufficient or not and what degree of balanced routing would be necessary to make this kind of leakage undetectable.

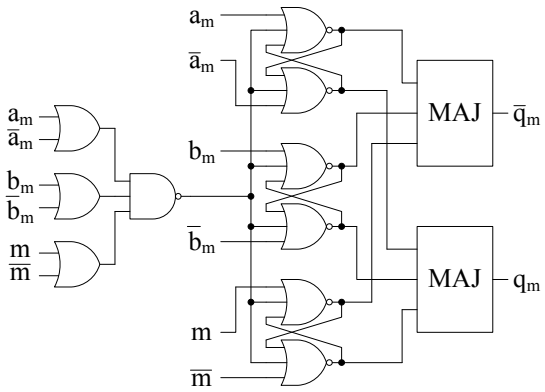


Figure 7.1: An iMDPL-AND cell. The original MDPL-AND cell only consists of the two CMOS majority cells MAJ.

Figure 7.1 shows the schematic of an iMDPL AND cell. The three OR and the NAND cell on the left side implement the EPDU, which generates 0 at its output only if all input signals a_m, b_m, and m are in a differential state. The following three set-reset latches, each consisting of two cross-coupled 3-input NORs, work as gate elements. As long as the EPDU provides a 1, each NOR produces a 0 at its output. Thus, the outputs of both MAJ cells are 0 and the iMDPL cell is in the precharge state.

When the EPDU provides a 0 because all input signals have been set to a differential state, the set-reset latches evaluate accordingly and the MAJ cells produce the intended output according to the masked AND function. Note that this evaluation only happens after all input signals have arrived differentially, i.e. no early propagation occurs in the evaluation phase.

Finally, if the first input signal is set back to the precharge value, the EPDU again produces a 1 and all six outputs of the set-reset latches switch to 0. Note that the set-reset latches are only set to this state by the EPDU and not by an input signal that switches back to the precharge value. Thus, also an early propagation effect at the onset of the precharge phase is prevented. An iMDPL-OR cell can be constructed from an iMDPL-AND cell by simply swapping (i.e. inverting) the mask signals m and $\overline{m}$.

Figure 7.2 shows the cell schematic of an iMDPL D-flip-flop (iMDPL-DFF). In principle, the functionality is the same as the one of the original MDPL-DFF shown in Figure 5.5. The additional cells just control the start of the evaluation and the precharge moments as described for the iMDPL-AND cell. Note that the iMDPL-AND cell that is part of the iMDPL-DFF is actually used as an iMDPL-NAND cell. The unnecessary MAJ cell in the iMDPL-AND cell, which produces the output signal q_m, can be omitted.

In Section 6.2.1 of the last chapter, we analyzed the early propagation prob-

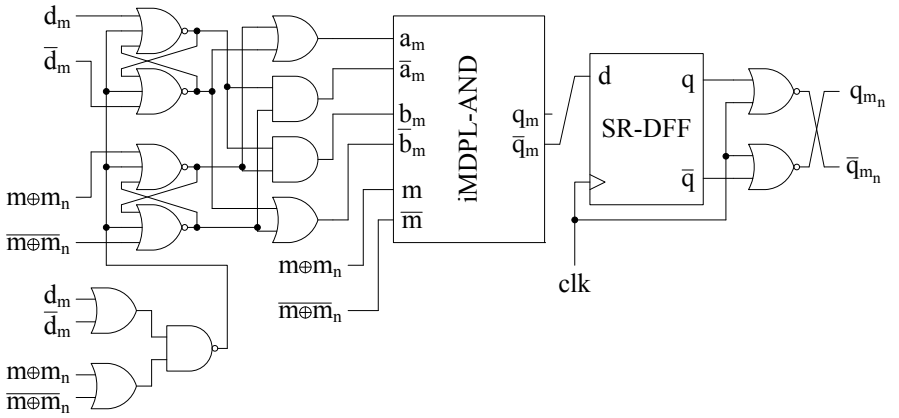


Figure 7.2: An iMDPL-DFF. The original MDPL-DFF does not have the two input latches and the EPDU.

lem of an MDPL microcontroller chip amongst others with the help of logic simulations and transition counting. Figure 6.8 shows the resulting correlation traces with peaks for the correct key for this analysis of MDPL. In a first effort to analyze the effectiveness of iMDPL, we performed the same analysis for the microcontroller implemented in iMDPL. To do this, the MDPL cells in the circuit netlist of the microcontroller core have been replaced by the corresponding iMDPL cells. Subsequently, the necessary logic simulations have been performed.

In Figure 7.3, the correlation traces when attacking simulated power traces of the core implemented in iMDPL are shown. The correlation traces for both the correct and the wrong power hypotheses show an ideal flat line for the attacked MOV operation. This indicates that the DPA leakage due to the early propagation effect is removed successfully within this simulation and analysis model.

Obviously, the price that has to be paid for the improvements in terms of early propagation is a further significant increase of the area requirements of iMDPL cells compared to MDPL. Since the iMDPL cells are already quite complex, exact figures for the area increase cannot be given in general because it depends significantly on the particular standard-cell library that is used to implement an iMDPL circuit. For example, there might be a standard cell available that implements the complete EPDU - such a cell is usually called OAI222. However, one can expect an increase of the area by a factor of around 3-4 compared to original MDPL, which is already approximately 5 times larger than CMOS. This makes it clear that carefully finding out which parts of a design really need to be implemented in DPA-resistant logic is essential to save chip area. More concrete figures for area requirements and circuit speed of iMDPL circuits can be found in Section 7.2 where a test chip containing iMDPL circuits is described.

A significant reduction of the cell size can be achieved by designing new standard cells that implement the functionality of iMDPL. Of course, that has

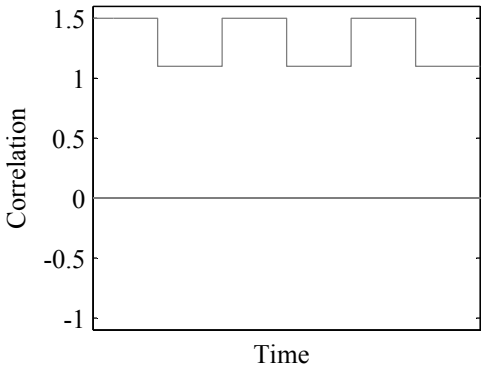


Figure 7.3: Result of the DPA attack on the iMDPL microcontroller core: transition count based on logic simulation of internal MOV operation in the IRAM; 256 samples; the correlation trace for the correct power hypothesis is plotted in black. The clock signal is indicated at the top.

the well known disadvantages of greatly increased design and verification costs. Furthermore, a change of the process technology would then mean spending all the effort to design the complete iMDPL standard-cell library again.

7.1.1 General Cells Required for MDPL, iMDPL, and Similar Circuits

In the course of the work for this thesis, two prototype chips have been designed and fabricated which contain MDPL circuits (see Section 6.2.1) and iMDPL circuits (see Section 7.2). The schematics of some important auxiliary cells for these masked circuits have never been explicitly published before. We make up for this in the current section about iMDPL because the functionality of these cells is the same for both logic styles. The set of auxiliary cells includes: flip-flops with asynchronous reset and preset inputs; tie-high and tie-low cells; interfacing cells in and out of the masked circuits; and a so-called mask preparation unit.

D-Flip-Flops with Asynchronous Reset/Preset Inputs

The standard design of MDPL-DFFs (see Figure 5.5) and iMDPL-DFFs (see Figure 7.2) makes it very easy to add asynchronous control inputs. The single-rail DFF within the masked cells only needs to be replaced by a DFF with the required asynchronous inputs. If a DFF in the original single-rail netlist has no asynchronous input, also the masked DFF does not have it. This advantage can get lost if more complex designs are used for the MDPL/iMDPL-DFFs. For example, the MDPL-DFF shown in Figure 5.6, which contains four single-rail DFFs instead of one, requires a connection to the asynchronous reset network in all circumstances. The reason is that the single-rail DFFs need to be initialized

correctly: two of them to the precharge value; the other two at least to a complementary value pair. The netlist conversion from single-rail to masked circuit needs to take care of this.

Tie-High and Tie-Low Cells

It is often the case that synthesized netlists contain some signals that are fixed to a logic value. These signals are usually provided by so-called tie-high and tie-low cells. In masked circuits, these signals are no longer constant. Instead, their values depend on the current state of the mask. Tie-high and tie-low cells in masked circuits are realized by direct connections to the complementary mask signal (m, $\overline{m}$). If we denote the dual-rail output of the masked tie-cell with (q_m, $\overline{q_m}$), the following relations must hold for a tie-low cell: $q_m = m$ and $\overline{q_m} = \overline{m}$. The opposite functionality is required for a masked tie-high cell.

Interfacing Cells In and Out of Masked Circuits

In the following, the design considerations made for the interfacing cells of masked circuits are summarized.

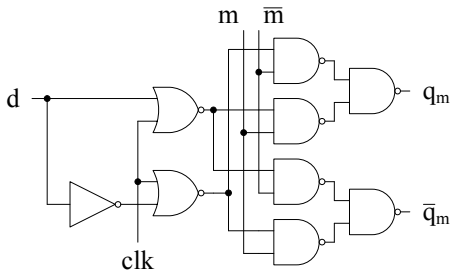


Figure 7.4: Interface cell for signals traversing into the masked circuit.

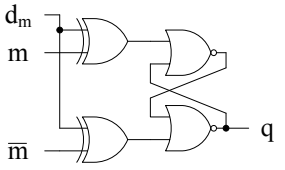


Figure 7.5: Interface cell for signals traversing out of the masked circuit.

Figure 7.4 shows the schematic of the interface cell for incoming signals of masked circuits. The inverter produces a dual-rail signal pair out of the single-rail input value d. The following two NOR cells add the precharge behavior to the dual-rail signal. Finally, two XORs implemented with three NANDs each are used to apply the mask to the dual-rail precharged signal.

A main design goal for this interface cell was that incoming signals are not delayed by one or more clock cycles. Such a behavior would significantly change the timing behavior of the circuit and could cause severe troubles especially after the single-rail to masked netlist conversion. A drawback of this approach is that glitches can occur in principle on the masked side and they might propagate into critical areas. The possibility of glitches is reduced by the design of the interface cell. If the single-rail input signal becomes stable already in the first half of a clock cycle (precharge phase), no glitches occur due to the blocking

behavior of the two "precharging-NORs" during that time. In order to eliminate any possibility of negative effects due to glitches, the high-level designer of the masked circuit has to make sure that the (per definition) uncritical input signal is fed into a masked DFF before it is combined with critical signals. The output signal of a masked DFF is always free of glitches.

The construction of the XORs out of the NAND cells is done to prevent a generation of glitches in this part of the interface circuitry during the second halve of a clock cycle (evaluation phase). The imperfect masking of the intermediate results calculated by the NANDs of the first logic level is uncritical because the underlying signal comes from the unprotected single-rail domain anyway. A general feature of the interface cell is that the complementary mask signals are symmetrically loaded by two identical NAND inputs.

The schematic of the interface cell for outgoing signals of masked circuits is depicted in Figure 7.5. The two XORs remove the mask from the signal d_m and the two subsequent NORs, connected to work as a set-reset latch, remove the precharge phase.

The interfacing cell does not introduce a delay of one clock cycle or more, which eases the overall circuit design significantly (see identical argumentation above in case of the interface cell for incoming signals). The glitches generated by the two XORs are uncritical since the outgoing signal will be used in the unprotected single-rail domain anyway. The mask removal is performed before the removal of the precharge phase in order to "naturally" achieve a symmetrical loading of the complementary mask signals. The possible area reduction by removing the mask after removing the precharge phase with only one XOR cell would not have this advantage while only saving an insignificant amount of circuit area.

Mask Preparation Unit

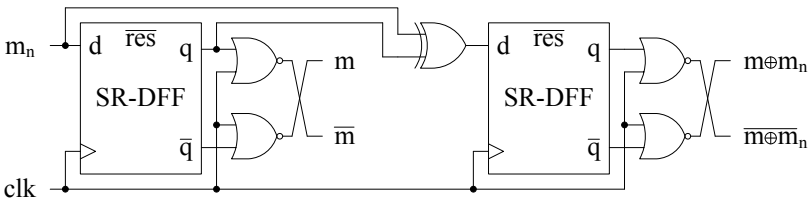


Figure 7.6: Mask unit to prepare the necessary mask signals.

Figure 7.6 shows the schematic of a so-called mask unit. It is used to synchronize the mask signal with the clock of the masked circuit. Furthermore, it takes care of generating the required mask signals in a dual-rail and precharged manner. The required mask signals are the mask of the current clock cycle m, the mask m xor'ed with the mask of the next clock cycle m_n: $m \oplus m_n$, and their complementary values $\overline{m}$ and $\overline{m \oplus m_n}$.

7.1.2 Simulated DPA-Attack on the Demo Circuit "AESencinit" Implemented in iMDPL

In order to show that the functionality of iMDPL improves the situation with respect to the early propagation problem, we have used the power trace estimation technique based on logic simulation. This technique has been presented in detail in Section 3.2. The specific tool that was used to map logic simulation results to power traces is called "vcd_analyzer" and is described in detail in Section 3.3. As demo circuit, the design "AESencinit" has been used, which is described in Section 3.3.1.

The implementation of the demo circuit in iMDPL, the power simulation with vcd_analyzer, and the DPA attack have been performed similarly as described for the MDPL implementation of the demo circuit in Section 6.2.2. The used terminology to denote the various simulation and analysis results is explained in detail in Section 3.3.1. The main difference of the MDPL and the iMDPL implementation of the demo circuit is that after synthesis, the CMOS circuit has been translated to an iMDPL circuit. The **comb**-part of the iMDPL demo circuit was simulated for all $256^2 * 4$ possible input value transitions (8-bit data input; 1-bit mask input).

Attack Results

The logic simulation of the iMDPL implementation of the AESencinit demo circuit has been performed with a clock cycle time of $540\,ns$. The 8-bit secret key used in the simulation has been 165 (0xA5). In our detailed analysis of iMDPL, we concentrated on the early propagation effect like we did for MDPL in Section 6.2.2. Therefore, we only simulated and attacked the combinational part of the demo circuit.

The power simulation results **comb/unitdelay/equal/tr** (effects of glitches and early propagation considered) show that the power traces of the combinational part of the iMDPL demo circuit are on average completely independent of the input data and the mask values. "On average" means over all possible input transitions for a particular 8-bit input value x ($0 \rightarrow x$, $1 \rightarrow x$, $2 \rightarrow x$, $\ldots$, $255 \rightarrow x$) and over all 4 possible mask bit transitions. Figure 7.7 depicts these 256 identical average-traces, which exactly lie on top of each other in the chosen power simulation model. Note that the power peak at the beginning of the precharge phase ($270\,ns$ to $539\,ns$) has been clipped. Its maximum height is 1314 toggles. As a result of such invariant power traces, the attack **comb/unitdelay/equal/tr/SBin/ZV**, which would exploit an early propagation leakage as it did in case of MDPL (see corresponding result in Section 6.2.2), does not succeed for the combinational iMDPL circuit. In fact, the DPA attack result is a perfect flat line in the given power simulation and attack model.

It was also verified that the corresponding attack **comb/unitdelay/random /tr/SBin/ZV** using random transition weights still does not lead to a successful attack. In this case, the average power traces are still completely independent of the input data and the mask values. Only the toggle values (y-axis values)

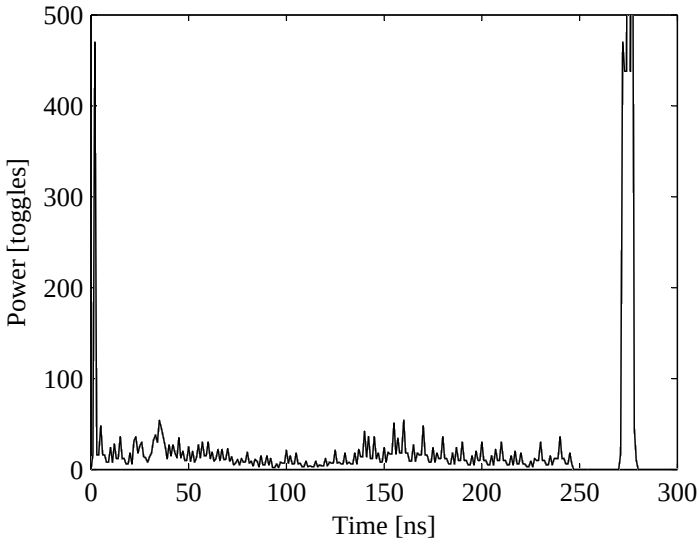


Figure 7.7: Average power traces for the simulation **comb/unitdelay/equal/tr** for different input values of the comb-part and random mask bits; black curve: input of comb-part is 165 ⇒ input of SubBytes block is 0 (unmasked).

of the simulated power traces approximately doubled since the random transition weights have been taken from the interval $[1.5, 2.5]$. The attack outcome **comb/unitdelay/random/tr/SBin/ZV** resulting from these power simulations is not a perfect flat line any more but there is still no distinct DPA peak for the correct key guess.

The above results show that in the chosen power simulation and attack model, iMDPL shows no early propagation problem any more. However, the practical results presented in the following Section 7.2 will show that the power simulation model still misses some effects that cause a small but exploitable DPA leakage for real devices implemented in iMDPL. This means that still more accuracy is required in the power simulation to cover all significant effects in terms of power leakage.

7.2 Analysis of the iMDPL Prototype Chip

For an evaluation of the masked logic style iMDPL in practice, a prototype chip containing a microcontroller and cryptographic coprocessors has been designed and fabricated. Its architecture is described in the next section. Subsequently, analysis results of the iMDPL prototype chip concerning its PA resistance are presented. The results show that iMDPL succeeds in avoiding a significant leakage due to early propagation, which troubled the predecessor logic style MDPL

(see Section 6.2.1). iMDPL increases the number of required measurements for a successful DPA attack by a factor of around 100 compared to the corresponding unprotected CMOS reference implementation.

7.2.1 iMDPL (and MDPL) Prototype Chip Architecture

The general architecture of the iMDPL prototype chip is shown in Figure 7.8. The system that has been implemented consists of the following main parts. An Intel 8051-compatible microcontroller and an AES cryptographic module that is connected to the microcontroller as a coprocessor form the so-called "μPCore" module. The microcontroller features 128 bytes of internal random-access memory (IRAM), a serial interface (RS-232), and an 8-bit parallel input/output (I/O) port. The program that is executed resides in an external read-only memory (ROM) chip. Additionally, an external RAM (XRAM) chip can also be attached. The second part is a stand-alone AES module and the associated I/O and operation controller. Both blocks are called the "AESCore". The internal structures of the AES coprocessor of the μPCore and the AES stand-alone module are identical. Details about it are given further below in this section. The AESCore shares the parallel I/O port and the RS-232 interface with the μPCore.

The core control logic is used to activate the currently selected core, i.e. supplying it with the clock signal and connecting its input and output signals to the corresponding chip pins. Part of the core control logic is a pseudo-random number generator (PRNG), which produces the mask values. The PRNG is controlled either by the μPCore microcontroller or by the AESCore controller depending on what is currently selected. The controlling is done via additional parallel ports that are connected on-chip to the PRNG. The main operations of the PRNG are: load a seed value, generate one random bit per clock cycle, provide a constant mask value, and stop operating.

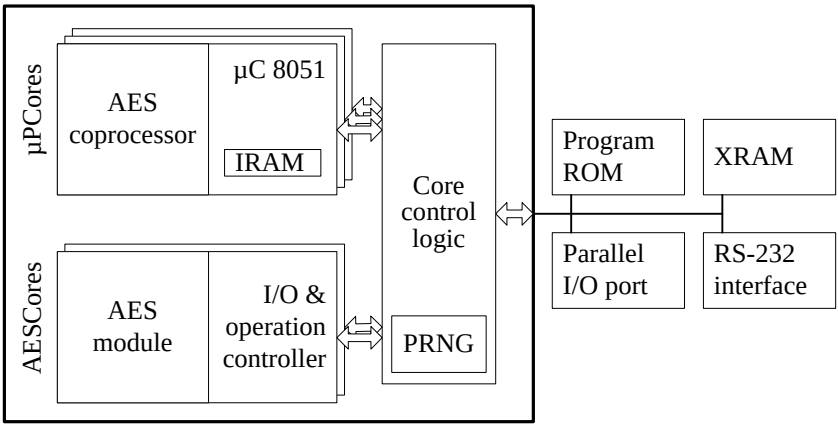


Figure 7.8: General architecture of the prototype chip.

The main reason for implementing also the AESCore was to have a possibility to analyze the AES module in an environment with a low amount of noise. Earlier experiences with the MDPL prototype chip showed that when analyzing the AES coprocessor, the microcontroller which controls the AES coprocessor acts as a huge noise generator. This significantly influences the DPA results of the AES coprocessor, i.e. much more measurements are required to detect a PA leakage.

On the iMDPL prototype chip, the system has been implemented in CMOS (for reference measurements) and in iMDPL. Thus, four cores have been implemented on this prototype chip: μPCore_CMOS, μPCore_iMDPL, AESCore_CMOS, and AESCore_iMDPL. The predecessor MDPL prototype chip mentioned in Sections 6.2.1 and 6.3 has a very similar architecture. The main difference is that it does not feature an AESCore module like the iMDPL prototype chip. Thus, the MDPL prototype chip contains the following two relevant cores: μPCore_CMOS and μPCore_MDPL.

The MDPL prototype chip has been implemented in a $0.13\,\mu m$ CMOS process technology from a well-known semiconductor company while the iMDPL prototype chip has been implemented in a $0.18\,\mu m$ CMOS process technology from United Microelectronics Corporation (UMC) [Unia]. The main attributes of the circuits of the prototype chips with regard to area requirements and speed are summarized in Table 7.1. The figures in the table indicate that MDPL increases the circuit area approximately by a factor of 5 while for iMDPL this factor is around $19-20$. The speed of the MDPL circuit is about 40 % of that of the corresponding CMOS circuit. In case of the iMDPL circuit, speed is reduced to approximately 20 %.

Table 7.1: Area and speed attributes of the μPCore and AESCore circuits on the MDPL prototype chip ($0.13\,\mu m$ process technology) and on the iMDPL prototype chip ($0.18\,\mu m$ process technology).

	CMOS 0.13	MDPL 0.13	CMOS 0.18	iMDPL 0.18
Area μPCore	$0.2\,mm^2$	$1\,mm^2$	$0.3\,mm^2$	$5.4\,mm^2$
Area AESCore	-	-	$0.1\,mm^2$	$1.9\,mm^2$
Speed μPCore	$18\,MHz$	$7\,MHz$	$15\,MHz$	$2.7\,MHz$
Speed AESCore	-	-	$> 15\,MHz$	$7.4\,MHz$

The cryptographic module included in the μPCore and in the AESCore is a hardware implementation of AES-128 [Nat01]. Its architecture follows the standard version of the regular and scalable AES hardware architecture presented in [MAD03], which reproduces the logical $4x4$ State layout of AES in hardware. The two main differences are the missing cipher block chaining (CBC) unit and the key storage in the key unit. Thus, the AES module is only capable of the electronic code book (ECB) mode of operation and the secret key has to be loaded into the AES module before each operation.

The datapath of the AES coprocessor is shown in Figure 7.9. The third and last main difference to the original design is the single MixColumns mod-

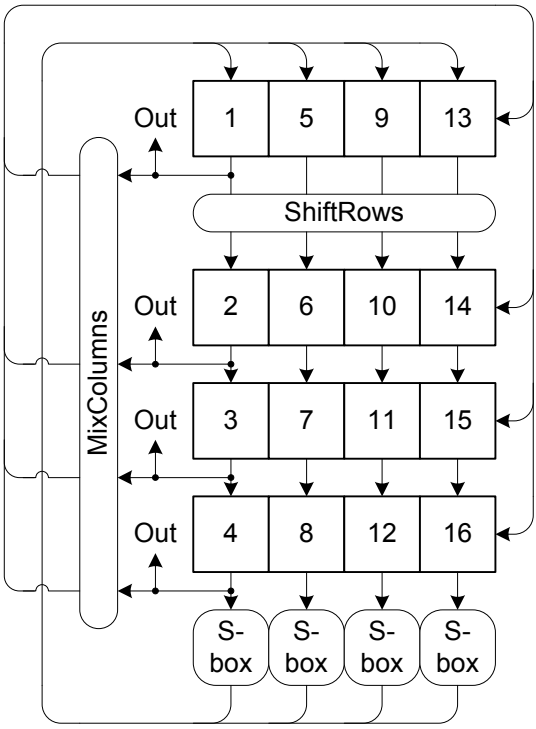


Figure 7.9: Architecture of the datapath of the AES hardware module.

ule attached to the left column of the AES State. The ShiftRows operation is implemented as a Barrel shifter. The four S-boxes performing the SubBytes operation are combinational and pipelined (one stage) implementations as described in [WOL02]. The initial State values are loaded column-wise from the right (not shown in figure). One round of AES takes 9 clock cycles in this architecture. First, each of the 16 AES State cells applies the AddRoundKey operation to its stored value. Then, the State is shifted row-wise down through the S-boxes and afterwards column-wise left through the MixColumns module. Calculation of the next AES round key happens in parallel to the MixColumns operation. When the AES operations have finished, the final State values are read out column-wise to the left.

7.2.2 iMDPL Prototype Chip Evaluation Results

We summarize the main results of the DPA attacks on the iMDPL prototype chip in the following. The full details of the analysis have been published recently in [KP09].

The settings of the measurement setup have been similar to them used for the MDPL prototype chip measurements as reported in Section 6.2.1. The main

difference has been that instead of a 10 Ω resistor, a BAT41 Schottky diode has been inserted in the V_{DD} line in forward direction to measure the current consumption of the iMDPL chip. The main idea behind this approach is the reduction of huge but unimportant spikes in the current trace, e.g. the spikes at the clock edges. Because of operating the diode in forward direction, huge current spikes should only cause a relatively small voltage drop. As a result, the oscilloscope input could be set to a higher sensitivity (y-axis resolution) while still recording the full current signal. Various experiments have indicated so far that the DPA results generally improve with this method. The oscilloscope sampling rate used for the iMDPL power measurements has been $2\,GS/s$.

Standard, first-order DPA attacks based on the linear Pearson correlation coefficient have been performed against the iMDPL prototype chip. Three targets have been chosen: the movement of a byte in the IRAM of the microcontroller (μPCore MOV) and an AES encryption operation both of the AES coprocessor (μPCore AES) and of the AES stand-alone module (AESCore AES). In case of the attacks on AES, the 16 output bytes of the SubBytes operation in the first round of the encryption process have been targeted. All three attacks have been similar to those conducted against the MDPL prototype chip, which are described in the Sections 6.2.1 and 6.3.

In case of the DPA attacks on the AES module we defined as success criterion that at least 9 of the 16 key bytes can be revealed by the DPA attack. The remaining 7 key bytes can then be revealed by a brute force attack with a reasonable effort. Hence we used the ninth-highest correlation peak to calculate the number of required power traces to perform a successful DPA attack. The formula to calculate the minimum number of required power measurements n_{min} for a successful attack from the correlation peak for the correct key guess $\rho_{ck,ct}$ has been presented in Section 2.2.

Table 7.2: Correlation peak heights $\rho_{ck,ct}$ and the corresponding minimal number of required measurements n_{min} of the DPA attacks on the iMDPL prototype chip. Targets have been a MOV operation executed by the μPCore microcontroller, the AES coprocessor of the μPCore, and the AES module of the AESCore.

Target	CMOS	iMDPL (PRNG active)	$\frac{iMDPL}{CMOS}$
μPCore MOV	$\rho_{ck,ct} = 0.334$ $n_{min} = 230$	0.0376 20 000	 87
μPCore AES	0.0331 25 600	not successful $> 6\,200\,000$	 > 242
AESCore AES	0.0334 25 100	0.00305 3 000 000	 120

Table 7.2 shows the attack results for the iMDPL prototype chip. The attack result for the MOV operation in the μPCore shows that iMDPL increases the number of required power traces by a factor of almost 90. For MDPL, this factor has been lower than 2 (see Table 6.4) due to the massive early propagation

problem of that logic style. The attack on the μPCore AES coprocessor implemented in iMDPL has not been successful with up to 6.2 million power traces. This translates to an improvement factor of iMDPL in this case of more than 240. The reason for this very high factor is most likely the significant amount of noise generated by the iMDPL microcontroller that controls the AES coprocessor. The same attack on the MDPL prototype chip was also not successful with up to 3.511 million traces (see Section 6.3.2). Finally, the attack on the AES operation of the stand-alone core yields an improvement in the number of required measurements by a factor of 120 in our attack scenario. In relation to the formidable area increase and speed penalty incurred by iMDPL, we believe this improvement is unfortunately too small to justify the utilization of iMDPL as a general PA countermeasure.

8 Conclusions

In this thesis, we have presented various proposals for PA-resistant logic styles that we have developed. The starting point of our research was the perception that most of the existing proposals for such logic styles so far are burdened by tough implementation constraints. An example of such a constraint is the balancing of the capacitive loads of wire pairs, which cannot be fulfilled perfectly in practice. Therefore, we aimed at PA-resistant logic styles that do not require such tough constraints. Our initial assumption was that with PA-resistant logic styles based on the concept of masking we could reach this goal. However, this turned out to be not really true.

Before we focused on our masked logic style proposals beginning with Chapter 4, we gave a general introduction to our thesis and to implementation and PA attacks in Chapters 1 and 2. The examples of successful implementation and PA attacks against commercial cryptographic products clearly show that these attacks are practical and severe threats to the security of such products.

In Chapter 3, we presented the simulation methodology we propose to estimate the PA resistance of cryptographic circuits. The methodology is based on simulating the activities of circuits at the logic level and mapping them to power consumption values. Our evaluation of the methodology with practically implemented tools showed that this approach is a good compromise between accuracy and resource requirements (time, memory). All important effects influencing the PA resistance can be considered and enough simulation runs can be performed to identify vulnerabilities in cryptographic circuits.

Chapter 4 started with a general introduction to masking at the cell level. Subsequently, the masked logic style "masked CMOS" (mCMOS) has been presented. mCMOS is based on standard CMOS cells, only moderately increases area and power consumption of circuits, and does not require special constraints

for the implementation process. However, during the development of mCMOS it turned out that signal glitches significantly reduce the PA resistance of masked circuits. This was verified in theory and practically based on simulations. Unfortunately, glitches occur in mCMOS circuits. Thus, mCMOS cannot be considered as a working PA-resistant logic style.

In Chapter 5, the "Masked Dual-Rail Precharge Logic" (MDPL) style has been presented. It avoids glitches by design but has increased requirements concerning area and power consumption compared to mCMOS. If MDPL would have shown very good results concerning its PA resistance, it would have been a perfect candidate for semi-custom circuit design because no special, tough implementation constraints would have been necessary. However, it has been shown in Chapter 6 that the PA resistance of MDPL suffers from another timing effect called early propagation. Results from an MDPL prototype chip have been used to practically verify these findings. However, it has also been shown for the AES crypto coprocessor on the prototype chip that early propagation does not completely nullify the resistance of MDPL against standard, first-order DPA attacks. The results for more advanced attack techniques also discussed in this chapter like the PDF-attack indicate that the use of a single mask bit for all signals in the circuit is another significant weakness of MDPL besides the early-propagation vulnerability.

An improved version of MDPL (iMDPL) has been presented in Chapter 7. Its goal is to remove the early propagation behavior from the MDPL cells. Various evaluation results including those from a prototype chip have shown that iMDPL indeed improves the PA resistance of cryptographic circuits. However, the increase in the number of required measurements compared to the unprotected CMOS circuits is most likely too low to justify the considerable overhead of iMDPL in terms of area, speed, and power. As a consequence, iMDPL does not seem to be practically useable as a general PA countermeasure but at most to protect specific, sufficiently small parts of cryptographic circuits.

In general, we conclude that masking does not seem to lead to PA-resistant logic styles that have "simpler" implementation requirements than the hiding approach. The main reason is that for masked logic styles also the timing behavior of the circuits (glitches, early propagation) is of great importance. Its negative effects on the PA resistance must not be underestimated. Hiding logic styles with the aim of an equal power consumption "naturally" have the tendency to minimize all effects that cause a data-dependent power consumption. This is not the case for PA-resistant logic styles that are straight-forwardly based on the masking principle.

Another important issue is the approach to base PA-resistant logic styles on standard CMOS cells. This is very helpful because it makes the implementation process much easier. On the other hand however, it typically leads to very big cells and circuits. Furthermore, it sets limits to the achievable PA resistance because real fine-tuning is only possible with full-custom cells. A future field of research could be the combination of the masked (with several mask bits) and hiding (with equal power consumption) approaches in a full-custom logic style.

Bibliography

[AE01] Mohamed W. Allam and Mohamed I. Elmasry. Dynamic Current Mode Logic (DyCML): A New Low-Power High-Performance Logic Style. *IEEE Journal of Solid-State Circuits*, 36(3):550–558, March 2001. ISSN 0018-9200.

[AMM+05] Manfred Aigner, Stefan Mangard, Renato Menicocci, Mauro Olivieri, Giuseppe Scotti, and Alessandro Trifiletti. A Novel CMOS Logic Style with Data Independent Power Consumption. In *International Symposium on Circuits and Systems (ISCAS 2005), Kobe, Japan, May 23-26, 2005, Proceedings*, volume 2, pages 1066–1069. IEEE, 2005. ISBN 0-7803-8834-8.

[ARR03] Dakshi Agrawal, Josyula R. Rao, and Pankaj Rohatgi. Multi-channel Attacks. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 2–16. Springer, 2003.

[AT03] Toru Akishita and Tsuyoshi Takagi. Zero-Value Point Attacks on Elliptic Curve Cryptosystem. In Colin Boyd and Wenbo Mao, editors, *Information Security, 6th International Conference, ISC 2003, Bristol, UK, October 1-3, 2003, Proceedings*, volume 2851 of *Lecture Notes in Computer Science*, pages 218–233. Springer, 2003.

[aus] austriamicrosystems. The austriamicrosystems Website. http://www.austriamicrosystems.com/.

[BBL04] Holger Bock, Marco Bucci, and Raimondo Luzzi. An Offset-Compensated Oscillator-Based Random Bit Source for Security Applications. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 268–281. Springer, 2004.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embed-*

ded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BGLT04] Marco Bucci, Michele Guglielmo, Raimondo Luzzi, and Alessandro Trifiletti. A Power Consumption Randomization Countermeasure for DPA-Resistant Cryptographic Processors. In Enrico Macii, Odysseas G. Koufopavlou, and Vassilis Paliouras, editors, *14th International Workshop on Integrated Circuit and System Design, Power and Timing Modeling, Optimization and Simulation, PATMOS 2004, Santorini, Greece, September 15-17, 2004, Proceedings*, volume 3254 of *Lecture Notes in Computer Science*, pages 481–490. Springer, 2004.

[BGLT06] Marco Bucci, Luca Giancane, Raimondo Luzzi, and Alessandro Trifiletti. Three-Phase Dual-Rail Pre-Charge Logic. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 232–241. Springer, 2006.

[BGM+03] Luca Benini, Angelo Galati, Alberto Macii, Enrico Macii, and Massimo Poncino. Energy-Efficient Data Scrambling on Memory-Processor Interfaces. In Ingrid Verbauwhede and Hyung Roh, editors, *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED), 2003, Seoul, Korea, August 25-27, 2003*, pages 26–29. ACM Press, 2003.

[BLGT05] Marco Bucci, Raimondo Luzzi, Michele Guglielmo, and Alessandro Trifiletti. A Countermeasure against Differential Power Analysis based on Random Delay Insertion. In *International Symposium on Circuits and Systems (ISCAS 2005), Kobe, Japan, May 23-26, 2005, Proceedings*, volume 4, pages 3547–3550. IEEE, 2005. ISBN 0-7803-8834-8.

[BRDG05] G. Fraidy Bouesse, Marc Renaudin, Sophie Dumont, and Fabien Germain. DPA on Quasi Delay Insensitive Asynchronous Circuits: Formalization and Improvement. In *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, pages 424–429. IEEE Computer Society, 2005. ISBN 0-7695-2288-2.

[BRG04] G. Fraidy Bouesse, Marc Renaudin, and Fabien Germain. Asynchronous AES Crypto-Processor Including Secured and Optimized Blocks. *Journal of Integrated Circuits and Systems (JICS)*, 1(1):5–13, March 2004. ISSN 1807-1953.

[BRR+04] G. Fraidy Bouesse, Marc Renaudin, Bruno Robisson, Edith Beigne, Pierre-Yvan Liardet, Solenn Prevosto, and J. Sonzogni. DPA on

Quasi Delay Insensitive Asynchronous Circuits: Concrete Results. In Pascal Fouillat, María Luisa López Vallejo, and Jean Tomas, editors, *XIX Conference on Design of Circuits and Integrated Systems (DCIS 2004), Bordeaux, France, November 24-26, 2004, Proceedings*, 2004.

[BS93] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1st edition, May 1993. ISBN 978-0387979304.

[BSI] BSIM Research Group. BSIM Homepage. http://www-device.eecs.berkeley.edu/~bsim3/.

[BSR06] G. Fraidy Bouesse, Gilles Sicard, and Marc Renaudin. Path Swapping Method to Improve DPA Resistance of Quasi Delay Insensitive Asynchronous Circuits. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 384–398. Springer, 2006. ISBN 3-540-46559-6.

[BSYK03] Alex Bystrov, Danil Sokolov, Alex Yakovlev, and Albert Koelmans. Balancing Power Signature in Secure Systems. In *14th UK Asynchronous Forum, Newcastle, June 2003*, 2003. Available online at http://www.staff.ncl.ac.uk/i.g.clark/async/ukasyncforum14/forum14-papers/forum14-bystrov.pdf.

[Cad] Cadence Design Systems. The Cadence Design Systems Website. http://www.cadence.com/.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CKN01] Jean-Sébastien Coron, Paul C. Kocher, and David Naccache. Statistics and Secret Leakage. In Yair Frankel, editor, *Finan-*

cial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings, volume 1962 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2001.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2003.

[CZ06] Zhimin Chen and Yujie Zhou. Dual-Rail Random Switching Logic: A Countermeasure to Reduce Side Channel Leakage. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 242–254. Springer, 2006.

[EKM+08] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KEELOQ Code Hopping Scheme. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008, Proceedings*, number 5157 in Lecture Notes in Computer Science, pages 203–220. Springer, 2008. ISBN 978-3-540-85173-8.

[ETS+05] Reouven Elbaz, Lionel Torres, Gilles Sassatelli, Pierre Guillemin, C. Anguille, M. Bardouillet, Christian Buatois, and Jean-Baptiste Rigaud. Hardware Engines for Bus Encryption: A Survey of Existing Techniques. In *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, pages 40–45. IEEE Computer Society, 2005.

[FDW04] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, August 2004.

[FG05] Wieland Fischer and Berndt M. Gammel. Masking at Gate Level in the Presence of Glitches. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES*

2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings, volume 3659 of *Lecture Notes in Computer Science*, pages 187–200. Springer, 2005.

[FML+03] Jacques J. A. Fournier, Simon Moore, Huiyun Li, Robert D. Mullins, and George S. Taylor. Security Evaluation of Asynchronous Circuits. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 2003.

[FP08] Martin Feldhofer and Thomas Popp. Power Analysis Resistant AES Implementation for Passive RFID Tags. In Christiopher Lackner, Timm Ostermann, Michael Sams, and Ronal Spilka, editors, *Proceedings of Austrochip 2008, October 8, 2008, Linz, Austria*, pages 1–6, October 2008. ISBN 978-3-200-01330-8.

[FS03] Wieland Fischer and Jean-Pierre Seifert. Unfolded Modular Multiplication. In Toshihide Ibaraki, Naoki Katoh, and Hirotaka Ono, editors, *Algorithms and Computation, 14th International Symposium, ISAAC 2003, Kyoto, Japan, December 15-17, 2003, Proceedings*, volume 2906 of *Lecture Notes in Computer Science*, pages 726–735. Springer, 2003. ISBN 3-540-20695-7.

[FWR05] Martin Feldhofer, Johannes Wolkerstorfer, and Vincent Rijmen. AES Implementation on a Grain of Sand. *IEE Proceedings on Information Security*, 152(1):13–20, October 2005.

[GHM+04] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu, Renaud Pacalet, and Jean Provost. CMOS Structures Suitable for Secured Hardware. In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, volume 2, pages 1414–1415. IEEE Computer Society, February 2004. ISBN 0-7695-2085-5.

[GHMP05] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu, and Renaud Pacalet. The "Backend Duplication" Method. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 383–397. Springer, 2005.

[Gie07] Benedikt Gierlichs. DPA-Resistance Without Routing Constraints? A Cautionary Note About MDPL Security. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna,*

Austria, September 10-13, 2007, Proceedings, volume 4727 of *Lecture Notes in Computer Science*, pages 107–120. Springer, September 2007.

[GM04] Jovan D. Golić and Renato Menicocci. Universal Masking on Logic Gate Level. *IEE Electronic Letters*, 40(9):526–527, April 2004. ISSN 0013-5194.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[GOK+05] Frank K. Gürkaynak, Stephan Oetiker, Hubert Kaeslin, Norbert Felber, and Wolfgang Fichtner. Improving DPA Security by Using Globally-Asynchronous Locally-Synchronous Systems. In *31th European Solid-State Circuits Conference - ESSCIRC 2005, Grenoble, France, September 12-16, 2005, Proceedings*, pages 407–410. IEEE, September 2005.

[Gol03] Jovan D. Golić. DeKaRT: A New Paradigm for Key-Dependent Reversible Circuits. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 98–112. Springer, 2003.

[Gou03] Louis Goubin. A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*, pages 199–210. Springer, 2003.

[GP99] Louis Goubin and Jacques Patarin. DES and Differential Power Analysis – The Duplication Method. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99, First International Workshop, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[GT03] Jovan D. Golić and Christophe Tymen. Multiplicative Masking and Power Analysis of AES. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, vol-

ume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2003.

[GvRVS09] Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. Wirelessly Pickpocketing a Mifare Classic Card. In *30th IEEE Symposium on Security and Privacy (S&P 2009), Oakland, California, USA, 17-20 May, 2009, Proceedings*, pages 3–15. IEEE Computer Society, May 2009.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2006*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 2006.

[HTH+06] David D. Hwang, Kris Tiri, Alireza Hodjat, Bo-Cheng Lai, Shenglin Yang, Patrick Schaumont, and Ingrid Verbauwhede. AES-Based Security Coprocessor IC in 0.18-*muhboxm*CMOS With Resistance to Differential Power Analysis Side-Channel Attacks. *IEEE Journal of Solid-State Circuits*, 41(4):781–792, April 2006.

[IEE01] IEEE. IEEE Standard 1364-2001: Verilog hardware description language. Available online at http://ieeexplore.ieee.org/servlet/opac?punumber=7578, March 2001. ISBN 0-7381-2826-0.

[IKV01] Mary J. Irwin, Mahmut T. Kandemir, and Narayanan Vijaykrishnan. SimplePower: A Cycle-Accurate Energy Simulator. IEEE TCCA Newsletter, January 2001.

[Ins] Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology. IAIK's Side-Channel Analysis Toolbox for MATLAB. http://www.iaik.tugraz.at/content/research/implementation_attacks/impa_lab_infrastructure/index.php#sca_toolbox.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[Kae08] Hubert Kaeslin. *Digital Integrated Circuit Design – From VLSI Architectures to CMOS Fabrication.* Cambridge University Press, 2008. ISBN 978-0-521-88267-5.

[KDFM95] Alan Kramer, John S. Denker, B. Flower, and J. Moroney. 2nd Order Adiabatic Computation with 2N-2P and 2N-2N2P Logic

Circuits. In Massoud Pedram, Robert W. Brodersen, and Kurt Keutzer, editors, *Proceedings of the 1995 International Symposium on Low Power Design (ISLPD), 1995, Dana Point, California, USA, April 23-26, 1995*, pages 191–196. ACM, 1995. ISBN 0-89791-744-8.

[Ker83a] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, January 1883.

[Ker83b] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:161–191, February 1883.

[Kir07] Mario Kirschbaum. Investigation of DPA-Resistant Logic Styles. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, May 2007.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KKG02] Franz Klug, Oliver Kniffler, and Berndt M. Gammel. Rechenwerk, Verfahren zum Ausführen einer Operation mit einem verschlüsselten Operanden, Carry-Select-Addierer und Kryptographieprozessor. German Patent DE 10201449 C1, January 2002. Available online at http://www.depatisnet.de/.

[KKMP09] Markus Kasper, Timo Kasper, Amir Moradi, and Christof Paar. Breaking KeeLoq in a Flash: On Extracting Keys at Lightning Speed. In Bart Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, volume 5580 of *Lecture Notes in Computer Science*, pages 403–420. Springer, June 2009. ISBN 978-3-642-02383-5.

[KKT06a] Konrad J. Kulikowski, Mark G. Karpovsky, and Alexander Taubin. DPA on Faulty Cryptographic Hardware and Countermeasures. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography, Third International Workshop, FDTC 2006, Yokohama, Japan, October 10, 2006, Proceedings*, volume 4236 of *Lecture Notes in Computer Science*, pages 211–222. Springer, October 2006.

[KKT06b] Konrad J. Kulikowski, Mark G. Karpovsky, and Alexander Taubin. Power Attacks on Secure Hardware Based on Early Propagation of

Data. In *12th IEEE International On-Line Testing Symposium (IOLTS 2006), July 10-12, 2006*, pages 131–138. IEEE Computer Society, July 2006.

[KME+08] Mehrdad Khatir, Amir Moradi, Alireza Ejlali, Mohammad T. Manzuri Shalmani, and Mahmoud Salmasizadeh. A Secure and Low-Energy Logic Style Using Charge Recovery Approach. In Vijaykrishnan Narayanan, C. P. Ravikumar, Jörg Henkel, Ali Keshavarzi, Vojin G. Oklobdzija, and Barry M. Pangrle, editors, *Proceedings of the 2008 International Symposium on Low Power Electronics and Design (ISLPED), 2008, Bangalore, India, August 11-13, 2008*, pages 259–264. ACM, 2008. ISBN 978-1-60558-109-5.

[KP07] Mario Kirschbaum and Thomas Popp. Evaluation of Power Estimation Methods Based on Logic Simulations. In Karl Christian Posch and Johannes Wolkerstorfer, editors, *Proceedings of Austrochip 2007, October 11, 2007, Graz, Austria*, pages 45–51. Verlag der Technischen Universität Graz, October 2007. ISBN 978-3-902465-87-0.

[KP09] Mario Kirschbaum and Thomas Popp. Evaluation of a DPA-Resistant Prototype Chip. In *25th Annual Computer Security Applications Conference (ACSAC 2009), 7-11 December 2009, Honolulu, Hawaii, USA*, 2009.

[KSS+05] Konrad J. Kulikowski, Ming Su, Alexander B. Smirnov, Alexander Taubin, Mark G. Karpovsky, and Daniel MacDonald. Delay Insensitive Encoding and Power Analysis: A Balancing Act. In *11th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC 2005), 14-16 March 2005, New York, NY, USA*, pages 116–125. IEEE Computer Society, March 2005. ISBN 0-7695-2305-6.

[KST06] Konrad J. Kulikowski, Alexander B. Smirnov, and Alexander Taubin. Automated Design of Cryptographic Devices Resistant to Multiple Side-Channel Attacks. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 399–413. Springer, 2006.

[KVW+08] Konrad J. Kulikowski, Vyas Venkataraman, Zhen Wang, Alexander Taubin, and Mark G. Karpovsky. Asynchronous Balanced Gates Tolerant to Interconnect Variability. In *International Symposium on Circuits and Systems (ISCAS 2008), 18-21 May 2008, Seattle, Washington, USA*, pages 3190–3193. IEEE, May 2008. ISBN 978-1-4244-1683-7.

[KVWT08] Konrad J. Kulikowski, Vyas Venkataraman, Zhen Wang, and Alexander Taubin. Power Balanced Gates Insensitive to Routing Capacitance Mismatch. In *Design, Automation and Test in Europe, DATE 2008, Munich, Germany, March 10-14, 2008*, pages 1280–1285. IEEE, 2008. ISBN 978-3-9810801-3-1.

[LFYL07] Kuan Jen Lin, Shan Chien Fang, Shih Hsien Yang, and Cheng Chia Lo. Overcoming Glitches and Dissipation Timing Skews in Design of DPA-Resistant Cryptographic Hardware. In Rudy Lauwereins and Jan Madsen, editors, *2007 Design, Automation and Test in Europe Conference and Exposition (DATE 2007), April 16-20, 2007, Nice, France*, pages 1265–1270. ACM Press, April 2007. ISBN 978-3-9810801-2-4.

[MAC+02] Simon Moore, Ross J. Anderson, Paul Cunningham, Robert D. Mullins, and George S. Taylor. Improving Smart Card Security using Self-timed Circuits. In *Eighth International Symposium on Asynchronous Circuits and Systems (ASYNC 2002), Proceedings*, pages 211–218. IEEE Computer Society, 2002.

[MAD03] Stefan Mangard, Manfred Aigner, and Sandra Dominikus. A Highly Regular and Scalable AES Hardware Architecture. *IEEE Transactions on Computers*, 52(4):483–491, April 2003.

[Man04a] Stefan Mangard. Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[Man04b] Stefan Mangard. *Securing Implementations of Block Ciphers against Side-Channel Attacks*. PhD thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, August 2004.

[MDP01] Thomas S. Messerges, Ezzy A. Dabbish, and Larry Puhl. Method and Apparatus for Preventing Information Leakage Attacks on a Microelectronic Assembly. US Patent 6,295,606 B1, September 2001. Filed: Jul. 26, 1999, Available online at http://www.uspto.gov/.

[MDS02] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Examining Smart-Card Security under the Threat of Power Analysis Attacks. *IEEE Transactions on Computers*, 51(5):541–552, January 2002.

[Men] Mentor Graphics. The Mentor Graphics Website. http://www.mentor.com/.

[Mes00] Thomas S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 2000.

[MGPV09a] Elke De Mulder, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede. Practical DPA Attacks on MDPL. Cryptology ePrint Archive (http://eprint.iacr.org/), Report 2009/231, 2009.

[MGPV09b] Elke De Mulder, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede. Practical DPA Attacks on MDPL. In *1st IEEE International Workshop on Information Forensics and Security (WIFS 2009), December 6-9, 2009, London, United Kingdom*. IEEE, 2009.

[MKSS09] Amir Moradi, Mehrdad Khatir, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. Charge Recovery Logic as a Side Channel Attack Countermeasure. In *10th International Symposium on Quality of Electronic Design (ISQED 2009), 16-18 March 2009, San Jose, CA, USA*, pages 686–691. IEEE, 2009. ISBN 978-1-4244-2952-3.

[MM08] Eric Menendez and Ken Mai. A High-Performance, Low-Overhead, Power-Analysis-Resistant, Single-Rail Logic Style. In Mohammad Tehranipoor and Jim Plusquellic, editors, *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST 2008), Anaheim, CA, USA, June 9, 2008, Proceedings*, pages 33–36. IEEE Computer Society, 2008. ISBN 978-1-4244-2401-6.

[MMM09] Robert P. McEvoy, Colin C. Murphy, and William P. Marnane. Isolated WDDL: A Hiding Countermeasure for Differential Power Analysis on FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2(3):23, March 2009.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer, 2007. ISBN 978-0-387-30857-9.

[MP03] Renato Menicocci and Johan Pascal. Elaborazione Crittografica di Dati Digitali Mascherati. Italian Patent IT MI0020031375 A, July 2003. Available online at http://www.depatisnet.de/.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-Channel Leakage of Masked CMOS Gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers'*

Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, February 2005.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully Attacking Masked AES Hardware Implementations. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[MSH+04] François Mace, François-Xavier Standaert, Ilham Hassoune, Jean-Jacques Quisquater, and Jean-Didier Legat. A Dynamic Current Mode Logic to Counteract Power Analysis Attacks. In *19th Conference on Design of Circuits and Integrated Systems (DCIS 2004), Bordeaux, France, November 2004, Proceedings*, pages 186–191, 2004. ISBN 2-9522971-0-X.

[MSSE09] Amir Moradi, Mahmoud Salmasizadeh, Mohammad T. Manzuri Shalmani, and Thomas Eisenbarth. Vulnerability Modeling of Cryptographic Hardware to Power Analysis Attacks. In *Integration, the VLSI Journal*, volume 42, pages 468–478. Elsevier, September 2009.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. Series on Discrete Mathematics and its Applications. CRC Press, 1997. ISBN 0-8493-8523-7, Available online at http://www.cacr.math.uwaterloo.ca/hac/.

[Nat01] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November 2001. Available online at http://www.itl.nist.gov/fipspubs/.

[Nat08] National Institute of Standards and Technology (NIST). FIPS-180-3: Secure Hash Standard, October 2008. Available online at http://www.itl.nist.gov/fipspubs/.

[NESP08] Karsten Nohl, David Evans, Starbug, and Henryk Plötz. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX Security Symposium, San Jose, CA, USA, 31 July, 2008, Proceedings*, pages 1–9. USENIX, 2008.

[OMPR05] Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A Side-Channel Analysis Resistant Description of the AES S-box. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption, 12th International Workshop, FSE*

2005, Paris, France, February 21-23, 2005, Revised Selected Papers, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.

[Pes] David Pescovitz. 1972: The release of SPICE, still the industry standard tool for integrated circuit design. http://www.coe.berkeley.edu/labnotes/0502/history.html.

[PGH+04] Norbert Pramstaller, Frank K. Gürkaynak, Simon Haene, Hubert Kaeslin, Norbert Felber, and Wolfgang Fichtner. Towards an AES Crypto-chip Resistant to Differential Power Analysis. In *30th European Solid-State Circuits Conference - ESSCIRC 2004, Leuven, Belgium, September 21-23, 2004, Proceedings*, pages 307–310. IEEE, September 2004.

[PKM09] Thomas Popp, Mario Kirschbaum, and Stefan Mangard. Practical Attacks on Masked Hardware. In Marc Fischlin, editor, *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009, Proceedings*, volume 5473 of *Lecture Notes in Computer Science*, pages 211–225. Springer, April 2009. ISBN 978-3-642-00861-0.

[PKZM07] Thomas Popp, Mario Kirschbaum, Thomas Zefferer, and Stefan Mangard. Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 81–94. Springer, September 2007. ISBN 978-3-540-74734-5.

[Plo09] Thomas Plos. Evaluation of the Detached Power Supply as Side-Channel Analysis Countermeasure for Passive UHF RFID Tags. In Marc Fischlin, editor, *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009, Proceedings*, volume 5473 of *Lecture Notes in Computer Science*, pages 444–458. Springer, April 2009.

[PM05] Thomas Popp and Stefan Mangard. Masked Dual-Rail Pre-Charge Logic: DPA-Resistance without Routing Constraints. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2005.

[PM06] Thomas Popp and Stefan Mangard. Implementation Aspects of the DPA-Resistant Logic Style MDPL. In *International Symposium on Circuits and Systems (ISCAS 2006), Island of Kos, Greece, May 21-24, 2006, Proceedings*, pages 2913–2916. IEEE, May 2006. ISBN 0-7803-9390-2.

[PMO07] Thomas Popp, Stefan Mangard, and Elisabeth Oswald. Power Analysis Attacks and Countermeasures. *IEEE Design & Test of Computers - Design and Test of ICs for Secure Embedded Computing*, 24(6):535–543, November-December 2007. ISSN 0740-7475.

[POM+04] Norbert Pramstaller, Elisabeth Oswald, Stefan Mangard, Frank K. Gürkaynak, and Simon Haene. A Masked AES ASIC Implementation. In Erwin Ofner and Manfred Ley, editors, *Proceedings of Austrochip 2004, October 8, 2004, Villach, Austria*, pages 77–82, October 2004. ISBN 3-200-00211-5.

[Pop04] Thomas Popp. Semi-Custom Design Flow for a DPA-Resistant Logic Style. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, January 2004.

[Pop09] Thomas Popp. An Introduction to Implementation Attacks and Countermeasures. In *7th ACM-IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE 2009), July 13-15, 2009, Cambridge, MA, USA, Proceedings*, pages 108–115. IEEE Computer Society, July 2009. ISBN 978-1-4244-4807-4.

[Rab] Jan M. Rabaey. The SPICE Home Page. http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/.

[RBE+07] Francesco Regazzoni, Stéphane Badel, Thomas Eisenbarth, Johann Großschädl, Axel Poschmann, Zeynep Toprak, Marco Macchetti, Laura Pozzi, Christof Paar, Yusuf Leblebici, and Paolo Ienne. Simulation-based Methodology for Evaluating DPA-Resistance of Cryptographic Functional Units with Application to CMOS and MCML Technologies. In Holger Blume, Georgi Gaydadjiev, C. John Glossner, and Peter M. W. Knijnenburg, editors, *7th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (IC-SAMOS 2007), Samos, Greece, July 16-19, 2007, Proceedings*, pages 209–214. IEEE, July 2007. ISBN 1-4244-1058-4.

[RCN03] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolić. *Digital Integrated Circuits – A Design Perspective*. Electronics and VLSI Series. Prentice Hall, 2nd edition, 2003. ISBN 0-13-090996-3.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. ISSN 0001-0782.

[SA05] Timmy Sundström and Atila Alvandpour. A comparative analysis of logic styles for secure IC's against DPA attacks. In *23rd*

NORCHIP Conference, November 21-22, 2005, pages 297–300, November 2005.

[SC01] Amit Sinha and Anantha Chandrakasan. JouleTrack – A Web Based Tool for Software Energy Profiling. In *38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001, Proceedings*, pages 220–225. ACM Press, June 2001.

[Sch03] Hermann Schneider. Analysis of the Resistance of Different Logic Styles Against SPA & DPA Attacks. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, July 2003.

[Sch09] Jörn-Marc Schmidt. *Implementation Attacks - Manipulating Devices to Reveal Their Secrets*. PhD thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, September 2009.

[Sha00] Adi Shamir. Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 71–77. Springer, 2000.

[Sko05] Sergei P. Skorobogatov. *Semi-invasive attacks - A new approach to hardware security analysis*. PhD thesis, University of Cambridge - Computer Laboratory, 2005. Available online at http://www.cl.cam.ac.uk/TechReports/.

[SMBY04] Danil Sokolov, Julian Murphy, Alex Bystrov, and Alex Yakovlev. Improving the Security of Dual-Rail Circuits. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 282–297. Springer, 2004. Extended version of paper at http://www.staff.ncl.ac.uk/i.g.clark/async/tech-reports/NCL-EECE-MSD-TR-2004-101.pdf.

[SMBY05] Danil Sokolov, Julian Murphy, Alex Bystrov, and Alex Yakovlev. Design and Analysis of Dual-Rail Circuits for Security Applications. *IEEE Transactions on Computers*, 54(4):449–460, April 2005. ISSN 0018-9340.

[SMY09] François-Xavier Standaert, Tal G. Makin, and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009, Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009. ISBN 978-3-642-01000-2.

[SS06] Daisuke Suzuki and Minoru Saeki. Security Evaluation of DPA Countermeasures Using Dual-Rail Pre-charge Logic Style. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2006.

[SS08] Minoru Saeki and Daisuke Suzuki. Security Evaluations of MRSL and DRSL Considering Signal Delays. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E91-A(1):176–183, January 2008. ISSN 0916-8508.

[SSI04] Daisuke Suzuki, Minoru Saeki, and Tetsuya Ichikawa. Random Switching Logic: A Countermeasure against DPA based on Transition Probability. Cryptology ePrint Archive (http://eprint.iacr.org/), Report 2004/346, December 2004.

[SSI05] Daisuke Suzuki, Minoru Saeki, and Tetsuya Ichikawa. DPA Leakage Models for CMOS Logic Circuits. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 366–382. Springer, 2005.

[SSI07a] Minoru Saeki, Daisuke Suzuki, and Tetsuya Ichikawa. Leakage Analysis of DPA Countermeasures at the Logic Level. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E90-A(1):169–178, 2007. ISSN 0916-8508.

[SSI07b] Daisuke Suzuki, Minoru Saeki, and Tetsuya Ichikawa. Random Switching Logic: A New Countermeasure against DPA and Second-Order DPA at the Logic Level. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E90-A(1):160–168, 2007. ISSN 0916-8508.

[SSSS09] Minoru Saeki, Daisuke Suzuki, Koichi Shimizu, and Akashi Satoh. A Design Methodology for a DPA-Resistant Cryptographic LSI

with RSL Techniques. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2009. ISBN 978-3-642-04137-2.

[ST07] Patrick Schaumont and Kris Tiri. Masking and Dual-Rail Logic Dont Add Up. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 95–106. Springer, September 2007.

[Ste06] Michael Steinkogler. The Power Consumption of Integrated Circuits in Simulation and Reality. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, May 2006.

[Syn] Synopsys. The Synopsys Website. http://www.synopsys.com/.

[SYO09] Kazuo Sakiyama, Tatsuya Yagi, and Kazuo Ohta. Fault Analysis Attack against an AES Prototype Chip Using RSL. In Marc Fischlin, editor, *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009, Proceedings*, volume 5473 of *Lecture Notes in Computer Science*, pages 429–443. Springer, April 2009.

[TAV02] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In *28th European Solid-State Circuits Conference - ESSCIRC 2002, Florence, Italy, September 24-26, 2002, Proceedings*, pages 403–406. IEEE, September 2002.

[The] The Mathworks. MATLAB - The Language of Technical Computing. http://www.mathworks.com/products/matlab/.

[THH+05] Kris Tiri, David D. Hwang, Alireza Hodjat, Bo-Cheng Lai, Shenglin Yang, Patrick Schaumont, and Ingrid Verbauwhede. Prototype IC with WDDL and Differential Routing - DPA Resistance Assessment. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 354–365. Springer, 2005.

[THM07] Stefan Tillich, Christoph Herbst, and Stefan Mangard. Protecting AES Software Implementations on 32-bit Processors against Power

Analysis. In Jonathan Katz and Moti Yung, editors, *Proceedings of the 5th International Conference on Applied Cryptography and Network Security (ACNS 2007)*, volume 4521 of *Lecture Notes in Computer Science*, pages 141–157. Springer, June 2007.

[Tho08] Herbert Hugh Thompson. How I Stole Someone's Identity. Scientific American http://www.scientificamerican.com/article.cfm?id=anatomy-of-a-social-hack, August 2008. Last visited: 25th of November 2009.

[TKL05] Elena Trichina, Tymur Korkishko, and Kyung-Hee Lee. Small Size, Low Power, Side Channel-Immune AES Coprocessor: Design and Synthesis Results. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers*, volume 3373 of *Lecture Notes in Computer Science*, pages 113–127. Springer, 2005.

[TL05] Zeynep Toprak and Yusuf Leblebici. Low-Power Current Mode Logic for Improved DPA-Resistance in Embedded Systems. In *International Symposium on Circuits and Systems (ISCAS 2005), Kobe, Japan, May 23-26, 2005, Proceedings*, volume 2, pages 1059–1062. IEEE, 2005. ISBN 0-7803-8834-8.

[Tri03] Elena Trichina. Combinational Logic Design for AES SubByte Transformation on Masked Data. Cryptology ePrint Archive (http://eprint.iacr.org/), Report 2003/236, 2003.

[TS07] Kris Tiri and Patrick Schaumont. Changing the Odds against Masked Logic. In Eli Biham and Amr M.Youssef, editors, *Selected Areas in Cryptography, 13th International Workshop, SAC 2006, Montreal, Quebec, Canada, August 17-18, 2006, Revised Selected Papers*, volume 4356 of *Lecture Notes in Computer Science*, pages 134–146. Springer, 2007. Available online at http://rijndael.ece.vt.edu/schaum/papers/2006sac.pdf.

[TV03] Kris Tiri and Ingrid Verbauwhede. Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 2003.

[TV04a] Kris Tiri and Ingrid Verbauwhede. A Dynamic and Differential CMOS Logic Style to Resist Power and Timing Attacks on Security IC's. Cryptology ePrint Archive (http://eprint.iacr.org/), Report 2004/066, 2004.

[TV04b] Kris Tiri and Ingrid Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, volume 1, pages 246–251. IEEE Computer Society, February 2004. ISBN 0-7695-2085-5.

[TV04c] Kris Tiri and Ingrid Verbauwhede. Place and Route for Secure Standard Cell Design. In Jean-Jacques Quisquater, Pierre Paradinas, Yves Deswarte, and Anas Abou El Kadam, editors, *Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS '04), 23-26 August 2004, Toulouse, France*, pages 143–158. Kluwer Academic Publishers, August 2004.

[TV04d] Kris Tiri and Ingrid Verbauwhede. Secure Logic Synthesis. In Jürgen Becker, Marco Platzner, and Serge Vernalde, editors, *Field Programmable Logic and Application, 14th International Conference, FPL 2004, Leuven, Belgium, August 30-September 1, 2004, Proceedings*, volume 3203 of *Lecture Notes in Computer Science*, pages 1052–1056. Springer, August 2004. ISBN 978-3-540-22989-6.

[TV05] Kris Tiri and Ingrid Verbauwhede. Design Method for Constant Power Consumption of Differential Logic Circuits. In *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, pages 628–633. IEEE Computer Society, 2005. ISBN 0-7695-2288-2.

[Unia] United Microelectronics Corporation. The United Microelectronics Corporation Website. http://www.umc.com/.

[Unib] University of California at Berkeley. The University of California at Berkeley Website. http://www.berkeley.edu/.

[VHUP05] Egon Valentini, Edmund Haselwanter, Robert Ulmer, and Thomas Popp. Configurable Logic Style Translation Based on an OpenAccess Engine. In *12th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2005), December 11-14th 2005, Gammarth, Tunisia, Proceedings*, volume 1, pages 389–392. IEEE, 2005.

[WH04] Neil H. E. Weste and David Harris. *CMOS VLSI Design – A Circuits and Systems Perspective*. Addison-Wesley, 3rd edition, May 2004. ISBN 0-321-14901-7.

[WOL02] Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger. An ASIC implementation of the AES SBoxes. In Bart Preneel, editor, *Topics in Cryptology - CT-RSA 2002, The Cryptographers' Track at the RSA Conference 2002, San Jose, CA, USA, February*

18-22, 2002, Proceedings, volume 2271 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2002.

[YB04] An Yu and David S. Brée. A Clock-less Implementation of the AES Resists to Power and Timing Attacks. In *International Conference on Information Technology: Coding and Computing (ITCC 2004), April 5-7, 2004, Las Vegas, Nevada, USA, Proceedings*, volume 2, pages 525–532. IEEE Computer Society, April 2004. ISBN 0-7695-2108-8.

[YFP03] Zhong C. Yu, Stephen B. Furber, and Luis A. Plana. An Investigation into the Security of Self-Timed Circuits. In *9th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC 2003), 12-16 May 2003, Vancouver, BC, Canada*, pages 206–215. IEEE Computer Society, 2003. ISBN 0-7695-1898-2.

[YS07] Pengyuan Yu and Patrick Schaumont. Secure FPGA circuits using controlled placement and routing. In *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis, Salzburg, Austria, September 30 - October 5, 2007*, pages 45–50. ACM Press, September 2007. ISBN 978-1-59593-824-4.

[YWV+05] Shengqi Yang, Wayne Wolf, Narayanan Vijaykrishnan, Dimitrios N. Serpanos, and Yuan Xie. Power Attack Resistant Cryptosystem Design: A Dynamic Voltage and Frequency Switching Approach. In *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, pages 64–69. IEEE Computer Society, 2005. ISBN 0-7695-2288-2.

[YY92] Masakazu Yamashina and Hachiro Yamada. An MOS Current Mode Logic (MCML) Circuit for Low-Power Sub-GHz Processors. *IEICE Transactions on Electronics*, E75-C(10):1181–1187, October 1992. ISSN 0916-8516.

[Zef07] Thomas Zefferer. Attacking DPA-Resistant Logic Styles in Practice. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, January 2007.

Author Index

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)