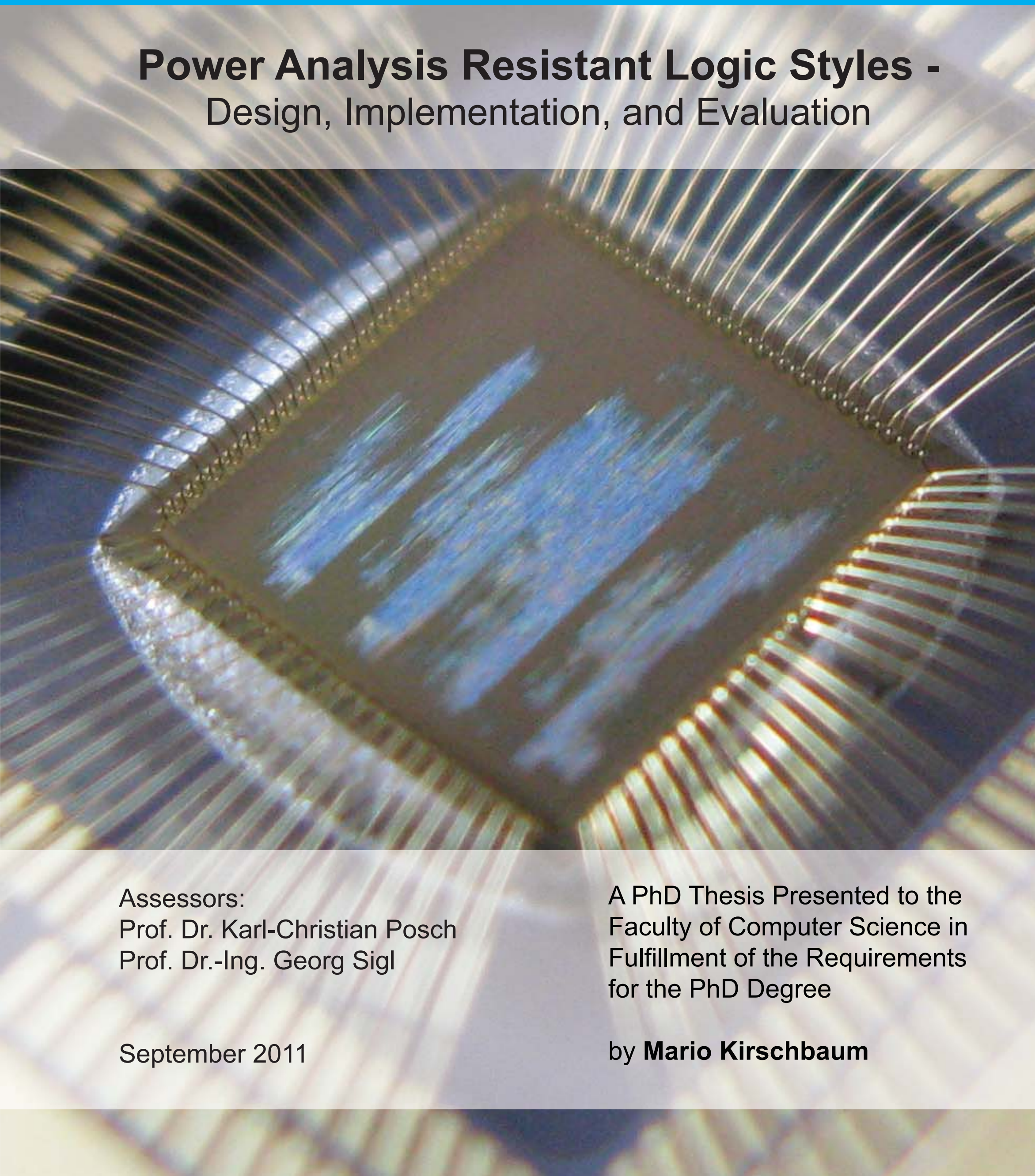
Graz University of Technology
Faculty of Computer Science
Institute for Applied Information Processing and Communications
IAIK

Power Analysis Resistant Logic Styles - Design, Implementation, and Evaluation



Assessors:
Prof. Dr. Karl-Christian Posch
Prof. Dr.-Ing. Georg Sigl

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

September 2011

by **Mario Kirschbaum**

Power Analysis Resistant Logic Styles — Design, Implementation, and Evaluation

by

Mario Kirschbaum

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.-Ing. Georg Sigl (TU Munich, Germany)

September 2011

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

Abstract

Many popular examples confirm that power analysis (PA) attacks pose a serious threat to cryptographic devices. During the last decade, several countermeasures against such attacks were proposed. Cell-level countermeasures, i.e. special logic styles, are believed to be one of the strongest approaches to counteract PA attacks.

This thesis focuses on the design, the implementation, and the evaluation of PA-resistant logic styles. The implementation of two different secure logic styles is covered by this thesis: iMDPL and DWDDL. Further, two different approaches to implement secure logic styles in a digital circuit are presented. In one case a whole hardware module is protected by a secure logic style, which results in a significant overhead in terms of required area. In the other case instruction-set extensions implemented on a modern 32-bit processor platform are protected by a secure logic style. All critical data outside of the protected area is strictly masked. It turns out that this approach is able to significantly reduce the area overhead.

Furthermore, different methods and techniques to investigate the effectiveness of PA-resistant logic styles are presented. There are basically two categories: theoretical and practical investigation methods. It is shown that already theoretical methods are able to reveal potential shortcomings in the design of PA-resistant logic styles. Further, it is demonstrated that a combination of different practical evaluation methods represents a powerful tool to investigate PA-resistant logic styles in every detail.

Acknowledgements

I would like to thank all colleagues and friends who supported me during the work for this thesis. Special thanks go to Thomas Popp who accompanied me in the early years of my research activity and with whom I worked on many interesting scientific experiments.

Furthermore, special thanks go to Jörn-Marc Schmidt and Manfred Aigner for their support in scientific and project-related questions. I also would like to thank Karl-Christian Posch, who has been the advisor and one of the assessors of this thesis, and Georg Sigl for serving as external assessor.

Many thanks go to my colleagues in the SEnSE group and to former members of the VLSI & Security group for countless fruitful discussions, interesting and partly funny scientific experiments, as well as for unraveling one or two knots in my brain: Thomas Plos, Michael Hutter, Alexander Szekely, Stefan Tillich, Martin Feldhofer, Christoph Herbst, Erich Wenger, and Sandra Dominikus. I also would like to thank Amir Moradi and Thomas Eisenbarth for the interesting and illuminating joint work and the fruitful discussions during the visits in Graz and in Bochum.

A thousand thanks go to my family for their lovely support, their guidance and their understanding throughout my whole life. Without them, this work would not have been possible at all. I also want to thank my family in-law for their support and for taking me into their family in such a lovely way.

Finally and most importantly, I want to thank my dear newly married wife Yvi for her outstanding support, for her lovely caring, and for her encouragement during my work for this thesis.

Mario Kirschbaum
Graz, September 2011

Table of Contents

List of Publications

1. 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.

2. 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.

3. 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.

4. 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.

5. 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*, Proceedings of 25th ACSAC, pages 43–50, 2009.

6. Stefan Tillich, Mario Kirschbaum, and Alexander Szekely. SCA-Resistant Embedded Processors – The Next Generation. In Carrie Gates, Michael Franz, and John P. McDermott, editors, *26th Annual Computer Security Applications Conference (ACSAC 2010), 6-10 December 2010, Austin, Texas, USA*, pages 211–220. ACM Press, 2010.

7. Jörn-Marc Schmidt, Thomas Plos, Mario Kirschbaum, Michael Hutter, Marcel Medwed, and Christoph Herbst. Side-Channel Leakage Across Borders. In Dieter Gollmann and Jean-Louis Lanet, editors, *Smart Card Research and Advanced Applications 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010, April 13-16, 2010, Passau, Germany, Proceedings*, Lecture Notes in Computer Science, pages 36–48. Springer, April 2010.

8. Stefan Tillich, Martin Feldhofer, Mario Kirschbaum, Thomas Plos, Jörn-Marc Schmidt, and Alexander Szekely. Hardware Implementations of the Round-Two SHA-3 Candidates: Comparison on a Common Ground. In *Proceedings of Austrochip 2010, October 6, 2010, Villach, Austria*, pages 43–48, October 2010. ISBN 978-3-200-01945-4.

9. Yossef Oren, Mario Kirschbaum, Thomas Popp, and Avishai Wool. Algebraic Side-Channel Analysis in the Presence of Errors. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems – CHES 2010, Santa Barbara, California, USA, August 17-20, 2010, Proceedings*, volume 6225/2010 of *Lecture Notes in Computer Science*, pages 428–442. Springer, 2010.

10. Stefan Tillich, Martin Feldhofer, Mario Kirschbaum, Thomas Plos, Jörn-Marc Schmidt, and Alexander Szekely. Uniform Evaluation of Hardware Implementations of the Round-Two SHA-3 Candidates. In *The Second SHA-3 Candidate Conference, Santa Barbara, USA, August 23-24, 2010*, 2010. No electronic version available.

11. Mario Kirschbaum and Jörn-Marc Schmidt. Learning from Electromagnetic Emanations - A Case Study for iMDPL. In *Second International Workshop on Constructive Side-Channel Analysis and Secure Design, 24-25 February 2011, Darmstadt, Germany*, Workshop Proceedings COSADE 2011, pages 50–55, 2011.

12. Amir Moradi, Mario Kirschbaum, Thomas Eisenbarth, and Christof Paar. Masked Dual-Rail Precharge Logic Encounters State-of-the-Art Power Analysis Methods. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems.*

13. Stefan Tillich, Mario Kirschbaum, and Alexander Szekely. Implementation and Evaluation of an SCA-Resistant Embedded Processor. In *Proceedings of the Tenth Smart Card Research and Advanced Application Conference, CARDIS 2011, September 15-16, 2011, Leuven, Belgium, Proceedings*, 2011.

List of Tables

List of Figures

Acronyms

AC	Alternating Current
AES	Advanced Encryption Standard
ALU	Arithmetic Logic Unit
AMBA	Advanced Microcontroller Bus Architecture
APB	AMBA Peripheral Bus
ASIC	Application-Specific Integrated Circuit
CLB	Configurable Logic Block
CMOS	Complementary Metal-Oxide Semiconductor
CRSABL	Charge Recycling Sense-Amplifier Based Logic
CSPRNG	Cryptographically Secure Pseudo-Random Number Generator
DDPL	Delay-Based Dual-Rail Precharge Logic
DES	Data Encryption Standard
DRP	Dual-Rail Precharge
DRSL	Dual-Rail Random-Switching Logic
DUT	Device Under Test
DWDDL	Double Wave Dynamic Differential Logic
EDA	Electronic Design Automation
EM	Electro-Magnetic
EMA	Electro-Magnetic Analysis
EPDU	Evaluation-Precharge Detection Unit
FF	Flip-Flop
FPGA	Field-Programmable Gate Array
FU	Functional Unit
GE	Gate Equivalent
GPIB	General Purpose Interface Bus
HD	Hamming Distance
HDL	Hardware Description Language
HW	Hamming Weight
I/O	Input/Output
IAIK	Institute for Applied Information Processing and Communications
iMDPL	Improved Masked Dual-Rail Precharge Logic
IRAM	Internal RAM
ISE	Instruction Set Extension
IU	Integer Unit
LAN	Local Area Network
LFSR	Linear Feedback Shift Register

MAJ	Majority
MDPL	Masked Dual-Rail Precharge Logic
MMU	Memory Management Unit
MOS	Metal-Oxide Semiconductor
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
MPW	Multi-Project Wafer
MUX	Multiplexer
NLFSR	Non-Linear Feedback Shift Register
OFB	Output Feedback
PA	Power Analysis
PC	Personal Computer
PDF	Probability Density Function
PRNG	Pseudo-Random Number Generator
PSU	Power Supply Unit
RAM	Random-Access Memory
RFID	Radio-Frequency Identification
ROM	Read-Only Memory
RSL	Random Switching Logic
SABL	Sense-Amplifier Based Logic
SCA	Side-Channel Analysis
SDF	Standard Delay Format
SDRAM	Synchronous Dynamic Random Access Memory
SNR	Signal-to-Noise Ratio
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random Access Memory
TDPL	Three-Phase Dual-Rail Precharge Logic
UMC	United Microelectronics Corporation
USB	Universal Serial Bus
VCD	Value Change Dump
VLSI	Very Large-Scale Integration
WDDL	Wave Dynamic Differential Logic
XOR	Exclusive OR
XRAM	External Random-Access Memory

1 Introduction

The application of security-related hardware devices has been rapidly increasing during the last decades. Devices operating in private environments have always been favored targets of very different types of fraudulent attempts. A popular example is the attack on the KEELOQ [EKM+08, KKMP09] code hopping scheme. Expending a certain effort, an attacker can disclose device keys and manufacturer keys, which allows to tamper keyless entry systems based on the KEELOQ scheme. Another recent publication presents successful attacks on the bitstream encryption feature of different Xilinx FPGAs [MBKP11, MKP11]. The performed attacks enable the extraction of secret keys used for the bitstream encryption. This allows an attacker to clone and to manipulate such devices.

In theory it is quite simple to protect a hardware device from being manipulated by locking up the device in a safe. Obviously, this approach renders the device useless for most security-related applications, like, for example, conditional access systems based on RFID technology or coding devices. Hence, besides meeting appropriate design goals like high performance, low area, or low power also the resistance against different types of attacks started to play a major role in the area of hardware design.

The first publication about power analysis (PA) attacks on the Data Encryption Standard [Nat99] (DES) by Kocher et al. [KJJ99] opened the door to a whole new research area that remained hidden from public for quite a long time. Until then, DES, which was published as a FIPS standard in 1977, was already known and used for more than 20 years. After Kocher et al. demonstrated the vulnerability of cryptographic hardware devices to PA attacks, countless interested research parties from all over the world rapidly started working on improving such attacks, on applying them to many other cryptographic devices, as well as on developing countermeasures against such attacks. During the last two decades, many highly interesting approaches to counteract PA attacks have been proposed by the *defense community*. The approaches were eagerly analyzed by the *attack community* and in many cases the countermeasure techniques were declared insufficient or erroneous.

In this thesis we focus on the design, the implementation, and the evaluation of cell-level countermeasures against PA attacks. This type of countermeasures copes with the fundamental relation between the processed data within a hardware device and its instantaneous power consumption. Hence, we assume that this approach to counteract PA attacks is the most promising one.

1.1 Our Contribution

Based on our findings of the security of the masked dual-rail precharge logic (MDPL) style [Kir07], we started our work for this thesis with the design of an improved version of the MDPL style, called iMDPL. Before actually implementing and fabricating the iMDPL style on a prototype chip, we performed detailed investigations of MDPL and other PA-resistant logic styles that suffered from an effect called early propagation. Our investigations were based on real measurements of an MDPL prototype chip as well as on transistor-level and logic-level simulations of different MDPL circuits. During our work, we analyzed and optimized different approaches to estimate the power consumption of digital circuits by means of simulations and to perform PA attacks in very early stages of hardware development [KP07]. In a joint work together with Thomas Popp we developed the iMDPL style [PKZM07], which prevents early propagation at each logic cell in the circuit. The drawbacks of iMDPL are a higher area requirement, a higher power consumption, and a significantly reduced maximum clock speed compared to an equivalent hardware implementation in CMOS logic.

We also implemented an iMDPL prototype chip and performed various evaluations and specific attacks on the chip [PKM09, KP09]. It turned out that the iMDPL style significantly reduces the effect of early propagation, but still shows a leakage. In further elaborate investigations based on our evaluation techniques using toggle counting we were able to resolve any open questions about iMDPL [MKEP]. We pinpointed the leakage in iMDPL and also confirmed that information about the mask value in an iMDPL circuit can be derived from the power consumption.

Our approach in case of the iMDPL prototype chip was the implementation of complete hardware modules (microcontrollers, AES modules) in a secure logic style, which results in a significantly increased area requirement. We followed a new approach which combines cryptographic instruction-set extensions, secure logic styles, and architectural masking [TKS10]. The advantage of the so-called *secure zone* approach is that only a relatively small module needs to be implemented in a secure logic style. We implemented this approach on a prototype chip [TKS11] and performed basic evaluations based on PA attacks so far. It turned out that this new approach is able to reduce the overall area overhead by combining two different countermeasures.

Besides the main work on the implementation and the evaluation of secure logic styles we also made significant progress in other areas. In a joint work together with Jörn-Marc Schmidt, we showed that attacks on electro-magnetic (EM) emanation measurements in combination with surface scanning can be used to perform highly detailed investigations of digital circuits [KS11]. Further, our evaluations showed that this technique enables the tracking of the data flow within a digital circuit to a certain degree.

In a joint work with Jörn-Marc Schmidt, Thomas Plos, Michael Hutter, Marcel Medwed, and Christoph Herbst we performed comprehensive experiments including 5 different devices and showed that integrated circuits do not leak side-channel information via the power-supply pins only [SPK+10]. Our investigations showed that also voltage fluctuations at conventional I/O pins can be exploited to mount successful side-channel attacks.

1.2 Organization of This Thesis

Chapter 2 introduces the power consumption of modern CMOS logic gates and their associated vulnerability to PA attacks. Subsequently, various countermeasure approaches against PA attacks are discussed. In the last part, practical examples of PA and electro-magnetic analysis (EMA) attacks on different devices are presented.

Chapter 3 focuses on cell-level countermeasures against PA attacks and presents an overview of different PA-resistant logic styles that were proposed during the last decade. The second part of this chapter deals with the implementation of PA-resistant logic styles in the context of a conventional VLSI design flow and discusses the necessary non-standard design steps based on different logic style approaches.

Chapter 4 discusses various methods to investigate the effectiveness of PA-resistant logic styles theoretically and practically. Based on the exemplary application of the presented methods on various PA-resistant logic styles the significance and the practicality of the different methods are pointed out.

Chapter 5 first summarizes the evaluation results of a prototype chip implementing the secure logic style MDPL. Subsequently, an improved version of the MDPL style, called iMDPL, is presented and investigated theoretically. Further, a prototype chip implemented in iMDPL as well as a comprehensive evaluation of this chip is presented. The presented evaluations involve all techniques presented in Chapter 4. The results show that the improvement significantly increased the resistance against PA attacks. The presented evaluations also reveal existing shortcomings in the iMDPL style.

Chapter 6 presents the so-called "secure-zone" approach where cryptographic instruction-set extensions (ISEs) are combined with different countermeasure approaches against PA attacks. Subsequently, a prototype chip that is based on the secure-zone approach and that contains two different secure logic styles is presented and evaluated. It turns out that the combination of different countermeasures is highly beneficial.

Chapter 7 contains the conclusions about our work in the field of designing, implementing, and evaluating PA-resistant logic styles. We also give an outline to current and future work to resolve any open questions about the prototype chip presented in Chapter 6.

2 Power Analysis Attacks

In this chapter we first elaborate on the reason why PA attacks on modern digital integrated circuits are feasible in the first place. We investigate the power consumption of a basic inverter implemented in complementary metal-oxide semiconductor (CMOS) logic and show that the instantaneous power consumption of the CMOS inverter strongly depends on consecutively processed values. Based on these findings, in the subsequent section we present an introduction to basic PA attacks. We discuss the fundamental steps of correlation-based PA attacks and also discuss similar attacks based on the EM emanation of digital circuits, so-called EMA attacks. In the third section we give an overview of possible countermeasures against PA attacks on different levels of application. In the last part of this chapter we present practical examples of PA and EMA attacks on different hardware devices. We demonstrate the vulnerability of unprotected modern cryptographic devices and show how weaknesses of imperfectly implemented countermeasures can be exploited. Furthermore, we discuss an example of an advanced EMA attack profiting from the measurement of specifically small areas on a digital integrated circuit. Parts of this chapter are based on results of a joint work together with Jörn-Marc Schmidt and were published in [KS11].

2.1 Side-Channel Leakage of CMOS Circuits

Today, a vast majority of digital integrated circuits is implemented in CMOS logic. The main characteristic of CMOS logic is the symmetrical structure based on p-type and n-type metal-oxide semiconductor field-effect transistors (MOSFETs). The total power consumption of a digital circuit implemented in CMOS logic is the sum of the static power consumption and the dynamic power consumption of every logic cell. The static power consumption is caused by a small leakage current that is flowing through the transistors that are turned off. An actual example is given in [WH04]: the leakage current of a MOS transistor in a 100 nm process lies typically in the nA range. In most applications, the static power consumption of CMOS circuits is neglected, except for low-power applications, where special low-leakage process technologies come into play. From a security perspective the static power consumption of CMOS circuits can be neglected, as the leakage current only shows a very low data dependency. Contrary, the dynamic power consumption is significantly higher than the static power consumption, and even more important, it shows a strong dependency on the data processed by a CMOS logic cell. In the following, the fundamental characteristics of the dynamic power consumption of CMOS circuits, which enable the execution of PA attacks, are described by means of a conventional CMOS inverter.

The schematic of a CMOS inverter is depicted in Figure 2.1 (left). The inverter basically consists of a p-type MOSFET P and an n-type MOSFET N. The output node q of the inverter is naturally afflicted with a vast number of parasitic capacitances. In a simplified model we can assume two lumped capacitances, which represent all parasitic capacitances related to the output node (indicated as C_{L1} and C_{L2}). Depending on the state transition of the CMOS inverter one of the capacitances is charged. The events in case input a switches from $1 \rightarrow 0$ (i.e. $q : 0 \rightarrow 1$) are illustrated in the equivalent circuit in Figure 2.1 (middle). The transistors are represented by switches $S1$ and $S2$. Assuming that C_{L1} is charged from the previous state and C_{L2} is discharged the following charging processes occur: C_{L1} is discharged internally via $S1$ and C_{L2} is charged via i_{DD}, i.e. the CMOS inverter consumes a specific amount of power to charge C_{L2}. In case input a switches from $0 \rightarrow 1$ (i.e. $q : 1 \rightarrow 0$), very similar charging processes occur in the circuit as shown in Figure 2.1 (right): $S1$ is opened, $S2$ is closed, C_{L2} is discharged internally via $S2$, and C_{L1} is charged via i_{DD}, which also consumes a specific amount of power.

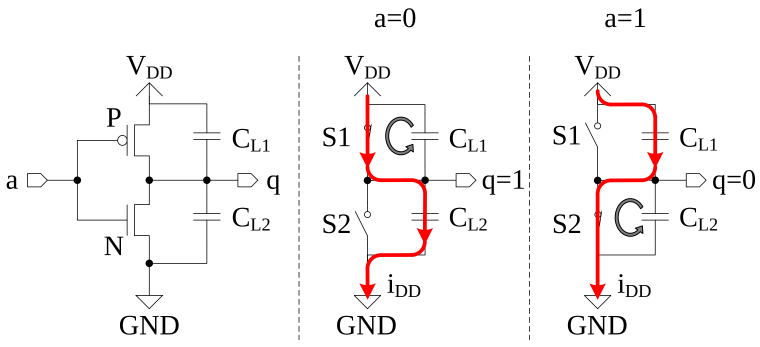


Figure 2.1: Depiction of the power consumption of a CMOS inverter: the schematic of the inverter (left), the equivalent circuit in case input $a = 0$ (middle) in case input $a = 1$ (right).

Besides the power consumption caused by charging processes, additional power is dissipated during the switching cycle of the transistors due to a direct-path current [Rab96]. There is a short period of time where both transistors are conducting simultaneously, causing additional dynamic power consumption. We can summarize the events in a CMOS inverter the following way: if the input of the inverter does not change (i.e. $a : 0 \rightarrow 0$ or $a : 1 \rightarrow 1$) the n-type and p-type MOSFETs keep their state. Hence, none of the output capacitances C_{L1}, C_{L2} needs to be charged and no direct-path current occurs. Neglecting the static power consumption, we can state that the power consumption of a CMOS inverter is 0 in case the input signal remains in its state. On the other hand, if the input of the inverter changes its state (i.e. $a : 0 \rightarrow 1$ or $a : 1 \rightarrow 0$) also the MOSFETs change their state. Hence, one of the output capacitances needs to be charged and a direct-path current occurs for a short period of time. We see, a change of the input value consumes a significant amount of power.

Furthermore, we can assume that the output capacitances C_{L1} and C_{L2} differ from each other, i.e. the power consumption of CMOS gates does not only disclose whether an output state transition occurred, it might also be possible to distinguish between the possible transition events $0 \rightarrow 1$ and $1 \rightarrow 0$. In the following section we present an introduction to PA attacks. In most cases it is sufficient to base a PA attack on the observation that a signal transition consumes a specific amount of power. The distinction between the possible switching events is usually not necessary.

2.2 An Introduction to Power Analysis Attacks

A device executing a cryptographic algorithm not only provides the intended encrypted or decrypted output (ciphertext or plaintext), but also emits various other information, so-called side-channel information, during its operation. Commonly known and well exploitable side channels are, for example, the timing behavior, the power consumption, or the EM emanation of a cryptographic device. While the timing side channel can be closed rather easily by means of carefully implementing a cryptographic algorithm, the power consumption and the EM side channels of devices turned out to be quite persistent. During the last years, most side-channel attacks on unprotected and on protected cryptographic devices were based either on power or EM measurements. The power consumption of a device can easily be measured with a digital oscilloscope by inserting a measurement shunt in the V_{DD} or the GND supply line of the device. Similarly, the measurement of the EM emanation of a device can be performed by placing an EM probe in the proximity of the measured device [Man03b].

PA and EMA attacks are passive and non-invasive attacks, i.e. an attacker neither influences the operation of the device (e.g. by manipulating the clock signal or the power supply) nor physically manipulates the attacked device (e.g. opening the chip's package and directly contacting metal wires). The target of PA attacks is to reveal secret information from a cryptographic device by analyzing the power consumption and extracting secret information piece by piece.

In the following, we describe the basic procedure of performing a PA attack by means of Figure 2.2. The attacked device (left) is supplied with known random input data and performs a cryptographic operation using the internally stored *secret key*. During the computation we measure the power consumption of the device and store the acquired power traces. We then generate a device model (right) to compute intermediate results of the cryptographic algorithm executed on the attacked device. Such an intermediate result usually depends on a small part of the secret key and on parts of the input data. Hence, we are able to compute hypothetical intermediate results based on guesses of a part of the secret key. For example, if the part of the attacked key is 8 bit wide we obtain 256 hypotheses. The *power model* is then used to transform the hypothetical intermediate results into hypothetical power consumption values. The final step in the attack is the analysis of the measured power traces and the power hypotheses by means of statistical methods, e.g. the linear Pearson correlation coefficient. As a result of such a correlation-based PA attack we obtain 256 correlation traces (bottom right in Fig. 2.2). If the device model and the power model were correct, one of the correlation traces (in this example the actual key byte was 107) will show a distinguishable correlation peak, indicating that this particular key hypothesis was correct[1]. The calculation of the hypothetical power consumption and the statistical analysis can then be performed for the remaining parts of the secret key in order to finally reveal the whole key. The advantage of PA attacks is that the measurement has to be performed only once, all parts of the secret key can be revealed using one set of measurements. A second advantage of PA attacks is that the commonly used power models are rather generic: as discussed in the previous section, the Hamming distance (HD) power model (i.e. number of changing bits) describes the power consumption of CMOS gates very well. As we will see in Chapter 5, also the Hamming weight (HW) power model (i.e. number of bits that are 1) provides a good approximation of the actual power consumption of a CMOS circuit.

2.2.1 PA Attacks in the Presence of Noise

There are different types of noise that may complicate or even completely prevent the execution of PA attacks on a device. As we will see in the next section, some countermeasures against PA attacks are based on the effects of noise. The following summary of different types of noise is partly based on [MOP07].

The first type of noise we refer to as *electronic noise*, which describes all possible sources of noise outside of a digital circuit. This type includes, for example, noise generated by the power supply of the measurement setup. Every power supply unit (PSU) shows a certain fluctuation of the output voltage, either because of a voltage swing in the supply of the PSU (usually 230 VAC) or because of load changes at the output that have to be compensated by the PSU. Further sources of electronic noise are active electronic devices in the proximity

[1]In the remainder of this thesis we will use a two-dimensional representation of the correlation results.

of the device under test and the measurement equipment like probes or the A/D converter in the digital oscilloscope.

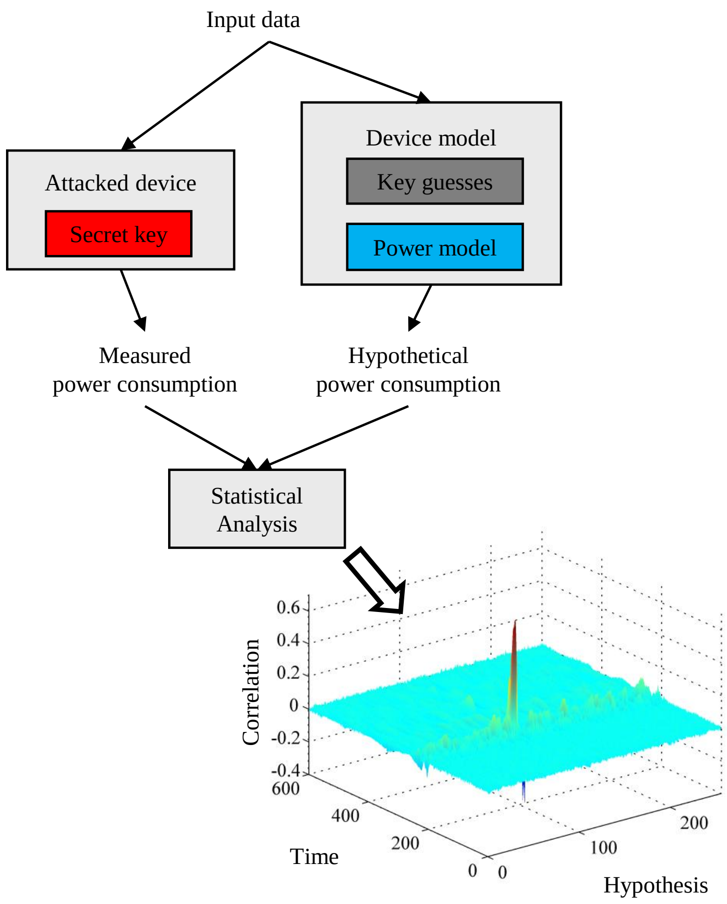


Figure 2.2: The principle of correlation-based PA attacks.

The second type of noise is called *switching noise* and directly originates from active parts in the measured digital circuit. Usually, a hardware design contains many different submodules, for example, an AES processor contains circuitry for the key schedule, the S-box operation, the MixColumns operation, and many more. At a specific moment in time, only a few of these modules process critical data values relevant for a PA attack, e.g. the output of the S-box module that propagates into a register. The switching activity in all other modules at this specific point in time affects the measurement of the actually leaking part of the circuit, and hence, complicates the execution of a PA attack. The EM measurement of specifically small parts of a circuit may help to reduce the electronic noise and the switching noise in the measurements to a certain degree.

Influence of Hardware Architectures on the Amount of Noise

In the following we give two examples of different AES hardware designs and discuss their influence on the amount of switching noise. Our first example is based on an AES implementation proposed by Feldhofer et al. [FWR05]. The main design goals of this module were low area and low power in order to be able to implement AES in highly resource-limited devices like RFID tags. The AES module is based on an 8-bit architecture and contains only a single S-box module and one multiplier for the MixColumns operation. This approach results in a very low complexity of 3 400 gate equivalents (GEs), in quite a long runtime of approximately 1 000 cycles for one encryption/decryption, but also in a very low amount of switching noise as only very few hardware modules are active in parallel.

Contrary to the low-area implementation, our second example is based on a high-performance AES implementation by Mangard et al. [MAD03]. This AES module contains 16 S-box modules and 16 multipliers implementing the Mix-Columns operation. On the one hand this results in a higher area requirement of 16 kGEs and only 34 cycles per encryption/decryption. On the other hand the amount of switching noise is significantly increased. In many cases the PA attack on an AES implementation is based on a single byte of the S-box output. In case of this high-performance AES module there are 15 other S-box modules, 16 multipliers, and many other modules working in parallel that produce significant amounts of switching noise. These examples show that the architecture of a design already influences the vulnerability to PA attacks to some degree. The next section discusses more effective ways of preventing such attacks.

2.3 Countermeasures Against PA Attacks

In the following we give a broad outline of countermeasure techniques that were proposed during the last decades. Based on the main stages in a typical VLSI design flow shown in Figure 2.3, we elaborate on different countermeasure approaches on the system level, on the architecture level, on the cell level, and on the environmental level. For some countermeasures we give practical examples and discuss the assets and drawbacks. The following summary of countermeasures on different levels is partly based on [MOP07].

2.3.1 Countermeasures on System Level

Countermeasures on the system level do not influence the fundamental vulnerability of a cryptographic device to PA attacks. However, there are some measures that can be taken on system level that significantly complicate the execution of PA attacks. However, some of these countermeasures entail new problems that have to be solved.

A PA attack usually requires more than one power trace, typically hundreds or thousands of power traces. A very simple approach would be to limit the number of cryptographic operations that can be performed within a specific time

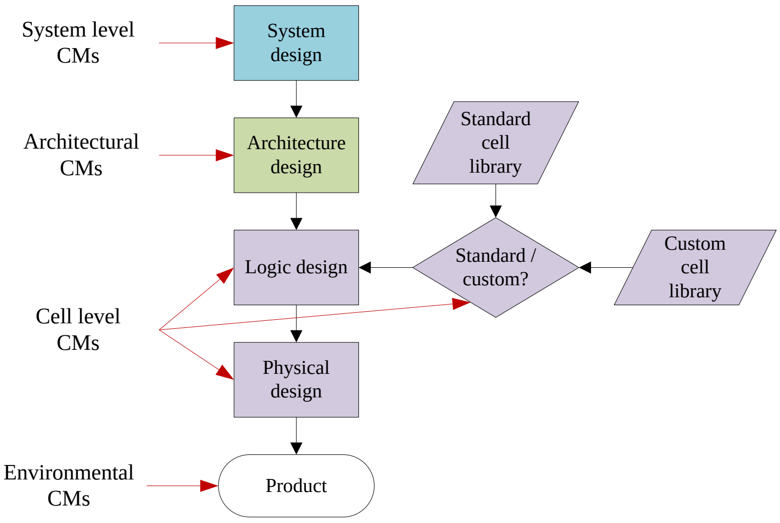


Figure 2.3: Countermeasures against PA attacks in the context of a VLSI design flow.

unit, e.g. at maximum two encryptions per minute. The obvious disadvantage of this approach is a significant decrease of performance and throughput, which would be intolerable in high-speed applications. Another method for impeding PA attacks would be the execution of regular updates of the secret key in order to prevent an attacker to gather enough information for an attack (e.g. in form of power traces) about one secret key. The disadvantages of regular key updates are: (1) the key update mechanism itself needs to be protected against PA attacks and (2) a secure key update has to be performed directly in the device's field of application.

2.3.2 Countermeasures on Architecture Level

On architecture level some countermeasures were proposed that directly deal with the problem of the relation between the processed data values within a cryptographic device and the instantaneous power consumption. Basically, there exist two approaches: masking countermeasures and hiding countermeasures. Figure 2.4 depicts the points in a cryptographic device where the two approaches apply. Both approaches can also be implemented on the cell level, which will be discussed in the next section.

Hiding Countermeasures

A hiding countermeasure tampers the physical side-channel output of a device in order to impede the extraction of critical information from the power consumption. There are basically two approaches of hiding countermeasures: the *randomization* of the power consumption, also called hiding in time, and the *equalization* of the power consumption, also called hiding in amplitude.

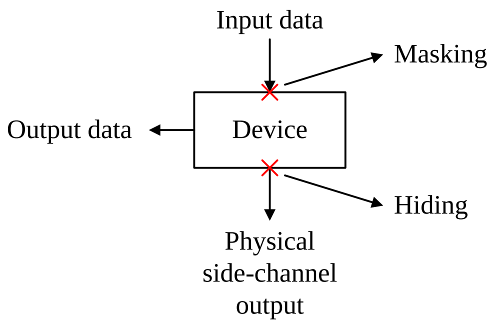


Figure 2.4: Illustration of two countermeasure approaches masking and hiding.

One example for hiding in time is the random insertion of additional operations during the execution of a cryptographic algorithm. The so-called *dummy operations* usually do not contribute anything to the actual cryptographic computation, which results in a decreased performance of the device. This approach decreases the signal-to-noise ratio in the power measurements, and thus complicates the execution of PA attacks. The countermeasure has to be implemented with great care as the dummy operations must not be distinguishable from normal operations executed on a device. Otherwise the dummy operations can easily be filtered from the power traces. Furthermore, the number of dummy operations must remain constant for each cryptographic computation. A variation of the number of executed dummy operations would lead to a dependency between the runtime of the algorithm and the number of inserted dummy operations. In Section 2.4.2 we will present an example of distinguishable dummy operations in an AES implementation.

A quite simple example for hiding in amplitude is the implementation of noise generators working in parallel to the cryptographic hardware. Such generators add a specific amount of switching noise to the power consumption, and thus complicate the execution of PA attacks. In order to have a significant influence on the overall power consumption, such noise generators usually require considerable amounts of area on a chip. A similar effect can also be achieved by a massively parallelized implementation of a cryptographic algorithm, e.g. an increase in switching noise in an AES-128 implementation can be achieved by realizing a 128-bit datapath and the usage of 16 S-boxes in parallel.

Masking Countermeasures

A masking countermeasure directly alters the processed data within a device, i.e. masking tries to break the dependency between the intermediate values in a cryptographic algorithm and the actually processed values in the digital circuit. Hence, the correlation between the unmasked intermediate values and the power consumption is broken, which theoretically makes PA attacks infeasible. The implementation of masking requires the adaption of an algorithm's operations in order to process the masked data correctly.

Masking is proven to be secure against first-order PA attacks. However, by means of second-order PA attacks a masking scheme can be broken. Such attacks exploit the fact that at least two masked intermediate values a_m and b_m in the digital circuit are masked with the same mask m. A preprocessing step combines two points t_a and t_b in the power traces that depend on a_m and b_m, respectively. The outcome is a single power value that depends on a_m and b_m, which can be used to perform a standard first-order PA attack. Also the hypothetical intermediate values have to be calculated using a combination of a and b. In [MOP07] it has been shown that the calculation of the *absolute difference* of the two points t_a and t_b is a good choice for second-order PA attacks on devices that leak the Hamming weight.

A masking technique has to be implemented very carefully: when consecutively processing the two masked values a_m and b_m in a CMOS circuit, the power consumption strongly depends on $HD(a_m, b_m) = HW(a_m \oplus b_m)$. In case of Boolean masking $a_m = a \oplus m$ and $b_m = b \oplus m$, and hence, the following equation holds:

$$\begin{aligned} HW(a_m \oplus b_m) &= HW((a \oplus m) \oplus (b \oplus m)) \\ &= HW(a \oplus b) \\ &= HD(a, b) \end{aligned}$$

We can see that a consecutive processing of two values masked with the same mask completely eliminates the effect of masking. Hence, designers need to handle masked values within a design with great care in order to avoid unintentional unmasking.

2.3.3 Countermeasures on Cell Level

PA-resistant logic styles cope with the fundamental reason for the vulnerability of digital circuits against PA attacks: the dependency between the processed values and the instantaneous power consumption of logic cells. The main advantage of a secure logic style would be that hardware/software designers could do their work completely undisturbed. They would not have to implement complex hiding or masking techniques on architecture level. During the last decades, several ideas of cell-level countermeasures have been proposed. Similar to the architectural level, the two categories of hiding and masking cell-level countermeasures have emerged.

Hiding Logic Styles

Hiding logic styles try to keep the instantaneous power consumption of a digital circuit constant during each clock cycle, which results in a constant amount of energy consumed in each cycle. Such logic styles are often based on full-custom logic cells and/or on complex routing techniques in order to achieve a data independent power consumption. Theoretically, this approach makes power analysis attacks infeasible, as no information about internally processed data can be extracted from the power consumption. Based on the example of a CMOS inverter discussed in Section 2.1, a basic hiding approach could be realized the following way: instead of using only one inverter, we implement a compound cell based on two identical inverters. The *direct* inverter performs the ordinary operation on the input data, the *complementary* inverter processes the inverted input data. Hence, in each clock cycle where the input signal changes both inverters do their work and switch their output nodes to the corresponding values. This way, in each switching cycle the total energy consumption of the compound cell $E_{total} = E_{0\to 1} + E_{1\to 0}$ and we cannot distinguish whether the direct inverter switched from $0 \to 1$ or from $1 \to 0$.

This approach seems promising, but what if the two inverters do not have exactly the same characteristics in terms of timing behavior? Only considering the energy consumption over a whole switching cycle does not reveal an information leakage, E_{total} remains constant. However, the observation of the instantaneous power consumption during a switching cycle is able to reveal differences caused by timing variations. As we will discuss later in Section 4.5.5, only considering the energy consumption over a whole switching cycle may disguise the presence of an information leakage.

The dual-inverter example suffers from another issue: what if the input signal does not change in a clock cycle? As discussed in Section 2.1, the dynamic power consumption of a CMOS inverter is dominating, i.e. in case the input signals remain constant both inverters consume a significantly lower amount of power. This already results in an information leakage: the difference between two consecutively processed bit values can be distinguished. It turns out that there are many circumstances that have to be considered for flawlessly implementing such a logic style.

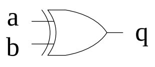
Masking Logic Styles

Masking logic styles follow another approach: instead of trying to keep the power consumption constant in each clock cycle, a masking approach randomizes the data processed in a digital circuit, and thus, also randomizes the power consumption. On the cell level usually Boolean masking is used, i.e. each value v in a circuit is XORed with a random unknown mask m: $v_m = v \oplus m$. Based on the example of a CMOS inverter, a masking approach results in the following behavior: the inverter receives the masked input value v_m and produces the inverted output value $q_m = \neg v_m$. In case the output q_m of the inverter switches to a new value, e.g. $0 \to 1$, we cannot tell if the value v or the mask m changed its state. Similarly, if q_m keeps its state, e.g. $0 \to 0$, it is possible that neither v nor m changed its state or that both v and m changed its state. Hence, as long as the mask value remains unknown, we cannot extract any information about the data value v from the instantaneous power consumption of the inverter.

Similar to hiding logic styles, also in case of masking the timing of the signals in a digital circuit has significant influences on the data-dependent information present in the power consumption. We assume that the value v is masked with m at a certain point in our example circuit by a simple XOR operation. The value v_m is then processed by the inverter. What if v and m arrive at different points in time at the XOR gate? It turns out, the XOR gate as well as the inverter may perform up to two switching events within a single clock cycle, depending on the arrival time of the input signals.

Such an effect is called *dynamic hazard* or *glitch*. Figure 2.5 illustrates the occurrence of a glitch at an XOR gate. With the arrival of $a = 1$ at time t_1 the XOR output q switches to 1. At time t_2, the second input signal b switches to 1, hence the output q switches back to 0. Glitches have significant effects on the power consumption [Rab96] of digital circuits, and hence, glitches also play a major role in the context of cell-level countermeasures.

Besides additional effort for implementing PA-resistant logic styles in the first place, in some cases there are considerable drawbacks: a significant overhead in terms of area requirement and/or power consumption, and a decrease of performance and throughput. We will discuss hiding and masking logic styles in more detail in Chapter 3 and in Chapter 4.



a	b	q
0	0	0
0	1	1
1	0	1
1	1	0

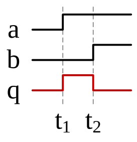


Figure 2.5: Example of a glitch occurring at the output q of an XOR gate due to different arrival times of the input signals a/b; left: an XOR gate; middle: XOR truth table; right: exemplary progress of signals at the XOR gate.

2.3.4 Environmental Countermeasures

Countermeasures can also be applied directly in the field of application of a final product in order to prevent PA attacks. Only preventing the physical access to the power supply of an integrated circuit is not enough. As Mangard showed in [Man03b], EMA attacks could be performed successfully in the proximity of the device as well as at a distance of several meters. In a joint work [SPK$^+$10] with Jörn-Marc Schmidt, Thomas Plos, Michael Hutter, Marcel Medwed, and Christoph Herbst we showed that not only the power supply pins of a device leak sensitive information about internally processed values. We successfully performed PA attacks on various devices (ATMega163 smart card, ARM7, Atmel 8051, FPGA, and a 180 nm ASIC) by measuring the voltage fluctuations at the I/O pins. Further, we showed that the data leakage could not be prevented by a transceiver circuit, for example, in a serial interface. It turned out that such countermeasures only have a minor impact on the resistance against PA/EMA attacks. Countermeasures on lower levels (architectural level or cell level) can provide a significantly higher degree of security.

2.4 Practical Examples of Side-Channel Analysis Attacks

In the following we present three practical examples of side-channel analysis attacks on different devices. In the first case, we demonstrate that modern wireless sensor nodes are highly vulnerable to EMA attacks. We show that the secret key of an off-the-shelf sensor node containing an AES-128 security processor can be revealed within a few hours. In the second case we investigate an AES-128 prototype chip protected by random insertion of dummy operations. By means of this example we demonstrate that such countermeasures have to be implemented with great care, otherwise the dummy operations can easily be recognized and the effect of the countermeasure can be eliminated. In the third example we present a technique we developed during the work for this thesis and which was published in [KS11]. We demonstrate the power of EMA attacks in combination with surface scanning, also called *stepped EMA attacks*. Our investigations of a prototype chip showed that the data flow within a digital circuit can be tracked to a certain degree, without physically tampering with the device under test.

2.4.1 Attacking an AES-128 Security Processor in a Modern Sensor Node

As a first practical example we demonstrate a correlation-based EMA attack on a modern wireless microcontroller module Jennic JN5148 [NXP11]. The JN5148 module contains an AES-128 security processor capable of AES encryption/decryption in counter mode: a *nonce* is concatenated with a *counter* value (which is increased after each computation) and is AES-encrypted with a secret key. The result of the AES encryption, a pseudo-random bit stream (*PRBS*), is XORed with the actual *plaintext*, which finally results in the *ciphertext*. Figure 2.6 depicts the AES encryption in counter mode.

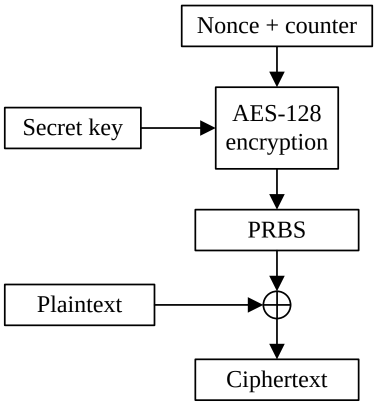


Figure 2.6: AES counter mode encryption.

We performed a known plaintext / known ciphertext attack on the JN5148 module, i.e. we assumed an attacker knows the *plaintext* and the *ciphertext*, and hence, the attacker also knows the result of the actual AES encryption *PRBS*. We targeted the last round of the AES encryption and tried to reveal the AES round key. In order to perform EM measurements of the JN5148 module we first had to build a special measurement coil as shown in Figure 2.7 (left). The coil was built using a 0.07 mm thin enameled copper wire, it consists of 10 windings and has an outer/inner diameter of approximately 0.7/0.4 mm. We placed the copper coil blindly on the JN5148 microcontroller through a small slit in the shielding integrated on the module as shown in Figure 2.7 (right).

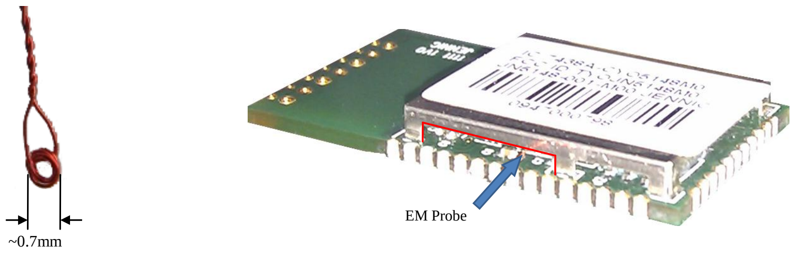


Figure 2.7: Self-built EM coil from 0.07 mm enameled copper wire, 10 windings, approximately 0.4 mm inner diameter (left), non-invasive placement of the copper coil through the shielding of the sensor module (right).

In a first measurement run we apparently placed the probe at a position on the microcontroller where no side-channel information from the AES security processor could be measured. By means of correlation-based attacks we could not reveal any information about the secret round key processed in the module. In a second attempt we placed the probe at different positions and inspected the measured EM traces on the oscilloscope during the repetitive execution of AES counter-mode encryptions. This way we discovered a position on the microcontroller where we measured a slightly increased EM emanation during the AES operations. The measured EM trace of our second measurement is shown in Figure 2.8: the trace indicates a slightly increased activity in the middle section.

At the second measurement position we were able to unambiguously reveal 15 out of 16 round-key bytes Figure 2.9 shows the correlation results of round-key byte 6: we performed the attack on 50 000 EM measurements, the correct key hypothesis is plotted in black. The obtained correlation value for round-key byte is 0.0562. According to [MOP07] this correlation value corresponds to a number of required EM traces of 8 800. The correlation values of the other 14 round-key bytes lie between 0.08 and 0.03, which corresponds to 4 000 and 30 000 required traces, respectively. It is very likely that a slightly different measurement position might also reveal the 16th round-key byte and also yield better results for the other key bytes, i.e. performing successful attacks using less traces.

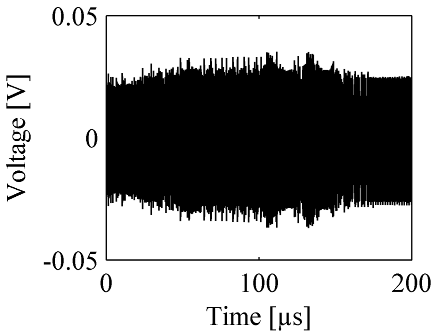


Figure 2.8: Sample EM trace of our second measurement of the JN5148 module: we can see a slight increase in the voltage in the middle section of the trace where the actual AES computation is performed.

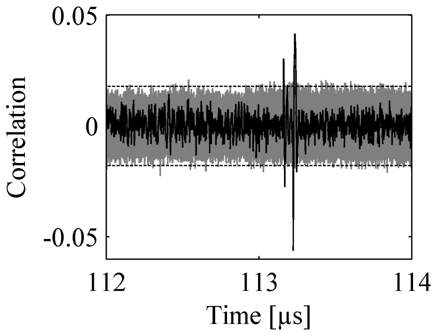


Figure 2.9: Result of the EMA attack on AES counter-mode encryptions in the JN5148 microcontroller, 50 000 traces, HW power model, correlation traces for round-key byte 6; the trace for the correct key hypothesis is plotted in black.

This example shows that commonly available sensor nodes are highly vulnerable to EMA attacks, if an adversary gains physical access to the device. We performed a basic attack without detailed knowledge about the architecture of the microcontroller. Even without actually seeing the chip on the sensor node due to the shield we were able to perform a successful EMA attack, which only took us a few hours including the first inconclusive measurement.

2.4.2 Exploiting Flaws in a Protected AES Implementation

The following example shows that the implementation of countermeasures has to be performed with great care in order to provide a certain level of security. We investigated an AES-128 prototype chip [Ach05] that is protected by the random insertion of dummy operations. The chip supports the insertion of dummy operations at three different points in time: before, during, and after the actual AES operation. Once the countermeasure is activated, the total number of inserted dummy operations remains constant in order to prevent the detection of the dummy operations by simply analyzing the runtime of the AES operation.

We performed power measurements of the prototype chip with three different configurations: the total number of inserted dummy operations in all measurements was set to 10, the number of dummy operations inserted before the AES operation in the three cases varied between 0-1, 0-3, and 0-7. In our measurements we saw that there occurs a slight voltage spike at the end of the AES operation, i.e. the power consumption in one clock cycle is slightly higher compared to all the others. Figure 2.10 shows the section between 0.04 V and 0.07 V of 500 power traces of each measurement. Around data point 2 500 we can see the shifting voltage spike (marked with a red ellipse). It turned out that this voltage spike is shifted in time, depending on the number of inserted dummy operations before the AES operation starts. In a very simple approach, an attacker could filter out all power traces where the voltage spike is shifted by at least one clock cycle, which would completely eliminate the effect of the countermeasure. This would also be the case for dummy operations inserted during the actual AES operation.

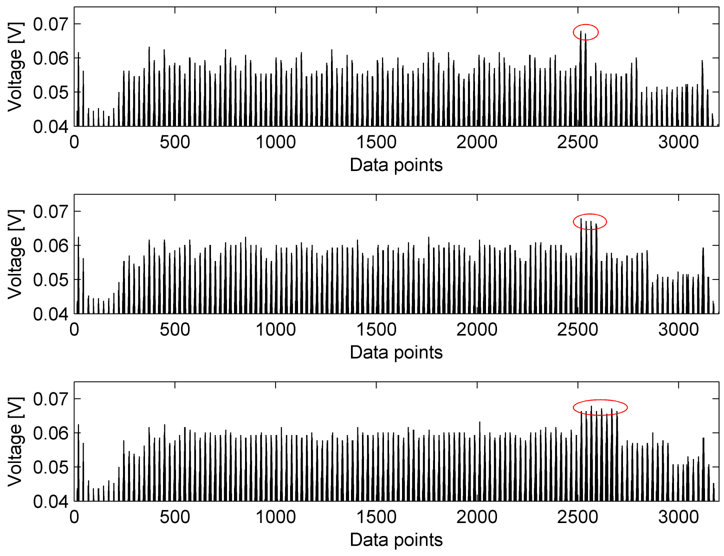


Figure 2.10: Detectable dummy operations in an AES-128 implementation: one voltage spike is shifted in time depending on the number of inserted dummy operations before the AES operation.

2.4.3 Data-Flow Tracking in a Digital Circuit

Based on an 8051-compatible microcontroller implementation on our iMDPL prototype chip, which is discussed in detail in Chapter 5, we performed stepped EM measurements using a 3-dimensional stepping motor setup. We stepped over an area of $3.45 \times 3.45\,mm$ on the chip die with a step size of 0.026 mm. At each coordinate we performed 1 000 move byte operations in the internal memory of the microcontroller, i.e. a randomly chosen byte value was moved from a source register to a target register. This resulted in a total of 17 689 measurement positions and 17.6 million EM traces.

On each measurement set we performed a bit-wise correlation-based EMA attack. This way, we obtained a correlation value for each bit of the moved byte value, at each point in time during the execution of the move byte operation, and for each coordinate on the chip die. Based on these results, we created correlation images of the chip die for each point in time, where the correlation value is color encoded. In a first step, we evaluated the correlation images of several data bits at different points in time. We discovered certain regularity in the evolution of the correlation images over time, i.e. a rather distinct data-flow path became visible within the images. In a next step, we superimposed the correlation images with the layout of our iMDPL prototype chip and compared our observations with simulations of the chip. More precisely, in the simulations we examined the data flow within the chip during the execution of the move byte operation and compared these findings with the correlation images. The results of our investigations are depicted in Figure 2.11. The figure shows correlation images of data bit 3 at six different points in time. In the top left plot (data point 145) we see that the correlation starts occurring in the register banks and the address decoder, and further develops stronger in the decoder and the ALU (data points 148 and 152; middle and bottom left plots). The correlation then *moves* to the memory multiplexer (data point 161, top right) and finally weakly develops from the MUX back to the register banks (data points 163 and 165; middle and bottom right plots). We followed exactly the same data flow in the simulations of the microcontroller: the move operation between two registers in the internal memory first occupies the register banks and the address decoder when the data value is read from the source register. The data is then fed into the ALU (no further processing of the byte value) and the memory multiplexer, which finally controls the route to the target register in the register banks.

This example shows that stepped EMA attacks are able to reveal detailed information about specifically small areas on a chip die. The analysis of stepped EM measurements may also be used to discover information about activated countermeasures, as most countermeasures cause slight differences in a digital circuit. The investigation of further devices based on EM measurements combined with surface scanning is marked for future work. By refining the measurement technique we expect to reveal further exploitable information from digital circuits, and hence, to enhance PA attacks on unprotected as well as on protected devices.

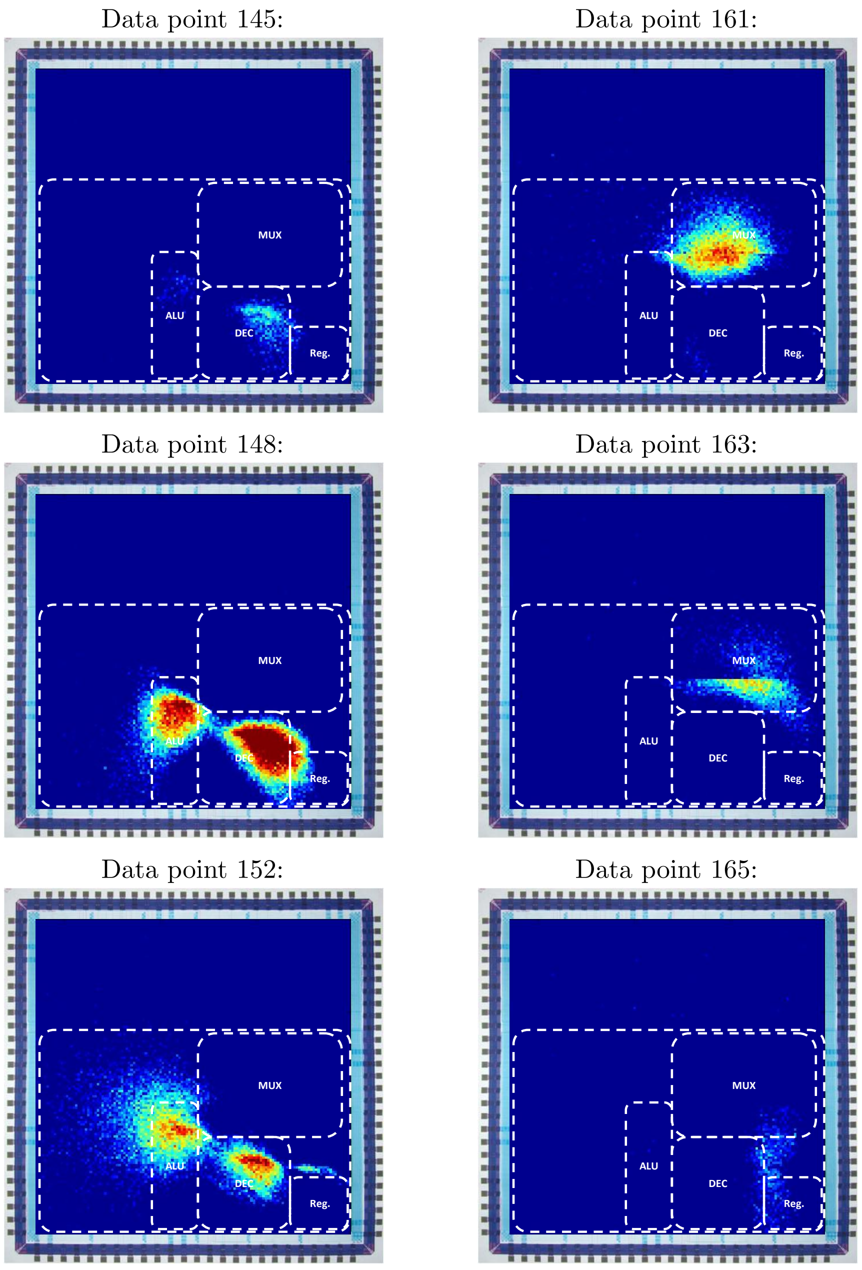


Figure 2.11: Evolution of the correlation of data bit 3 at different points in time: data points 145, 148, and 152 (left column); data points 161, 163, and 165 (right column); the yellow/red spots indicate a higher correlation.

In this chapter we showed that unprotected CMOS circuits are highly vulnerable to PA and EMA attacks. There exist several countermeasure techniques that complicate such attacks to a certain degree. We showed that such countermeasures have to be implemented with great care to ensure a flawless functionality and to prevent the detectability and the circumvention of the countermeasures. We also presented an advanced attack technique based on stepped EM measurements that allows to gather very detailed information of a digital circuit.

3 Power Analysis Resistant Logic Styles

The development of countermeasures against PA attacks in general is a versatile topic. It may comprise one or more of the following tasks: (1) the secure implementation of cryptographic algorithms or parts thereof (e.g. the implementation of a masked S-box or the implementation of operation shuffling in an AES module), (2) avoiding conditional branches depending on critical data in a hardware or software implementation of an algorithm, or (3) designing special hardware structures that do not reveal any information about the data processed within a digital circuit in terms of power consumption or timing (cf. Section 2.3.3). During the work for this thesis we focused on the latter approach. We worked on the design, the implementation, and the evaluation of cell-level countermeasures against PA attacks. Such countermeasures are commonly denoted as PA-resistant logic styles. As we showed in Chapter 2.1, the unbreakable relation between the data processed within a conventional CMOS circuit and its inherent power consumption characteristic represents a fundamental problem in respect of resistance against PA attacks. All special logic styles as well as all other countermeasures against PA attacks obviously share a common goal: a significant complication or even a total prevention of PA attacks on integrated circuits. More specifically, the goal of PA-resistant logic styles is to tamper the power consumption of a device with the result that no conclusions about any intermediate value processed within the device can be drawn from the instantaneous power consumption of the digital circuit. PA-resistant logic styles can basically be classified into two categories: (1) logic styles that are based on full-custom logic cells and (2) logic styles that are based on commonly available standard-cell libraries.

In the following, we first present a brief overview of some PA-resistant logic styles that were proposed during the last decade. In doing so, we elaborate on the two categories of full-custom based and semi-custom based approaches. In a next step we examine the implementation of special logic styles in the context of a semi-custom VLSI design flow and focus on two main stages, the logic design stage and the physical design stage, where we will discuss the non-standard design steps necessary in order to realize PA-resistant logic styles.

3.1 Logic Styles Based on Full-Custom Design

The implementation of logic styles based on full-custom logic cells requires detailed knowledge and experience in dealing with modern electronic design automation (EDA) design tools. A designer has to individually layout each required logic cell from scratch and hence she has to master all required design rules of the target process technology. Design rules mainly handle all kinds of geometric constraints for structures on all layers. In the following, some expressive examples out of countless design rules are given:

- **Cell height and width:** in many cases, only a few parts of the whole digital circuit are implemented in a secure logic style. Hence, a PA-resistant design based on full-custom cells does not exclusively utilize these specially designed cells but also utilizes standard cells from the cell library tailored for a specific target process technology. Because of mixing standard cells and full-custom cells, the full-custom cells are required to have the same cell height as standard cells and their width needs to be a multiple of the minimum standard-cell width in order to fit the placement grid. These design rules also determine the dimension and the exact position of the V_{DD} and *GND* rails and also define a specific area where the I/O pins of the cell can be placed.

- **Minimum and maximum widths:** structures, e.g. wires, falling below a specific minimum width run the risk of ending up as disjointed parts after the fabrication process. Maximum width rules typically apply to contacts between layers to ensure contacts of consistent quality [Kae08]. Usually, the maximum width rule is not violated during the full-custom design of cells due to space limitations given by the maximum cell height and possible cell widths.

- **Minimum spacings:** a designer has to take care of minimum spacing rules between adjacent structures in order to minimize the risk of short circuits. There are also minimum spacing rules that cannot be influenced by a designer: these rules apply to structures placed on different layers in order to avoid unwanted interactions between these structures. Such rules are usually determined by the target process technology.

There is one very important fact about design rules: not violating any design rule *does not guarantee* a fully functional and correct hardware implementation. Fulfilling all design rules just maximizes the chances of a successful fabrication of the design. A significant drawback for the designer of full-custom cells is that the design rules of different process technologies are mostly incompatible, i.e. a designer again has to start from scratch when switching to a new process technology. There may also be changes in the design rules between different versions of the same technology. This may result in the need of a partial redesign of previously designed full-custom cells when switching to a new version of a process technology.

Apart from constraints given by the design rules, a PA-resistant full-custom design putting emphasis on security in the first place provides high flexibility and optimization possibilities regarding secondary design goals like performance and area requirements. PA-resistant logic styles based on full-custom cells usually achieve higher performance and require less area than standard-cell based logic styles. This comes at the price of a tremendously increased effort for designing and verifying the full-custom cells. A full-custom approach hence only gets lucrative when the chip is produced in rather high volumes.

In the following we briefly introduce two logic styles proposed during the last years which are based on a full-custom design approach. We elaborate on the basic working principle of the logic styles and point out the crucial points during implementation. An overview of the logic styles and the required special steps necessary in a conventional design flow is given in Section 3.3. Later in Section 4.5 we will investigate the security of various logic styles.

3.1.1 SABL

Sense amplifier based logic (SABL) was proposed by Tiri et al. [TAV02] in 2002 and belongs to the category of countermeasures that try to hide the power consumption in the amplitude dimension. SABL introduces a precharge and an evaluation phase in each clock cycle, whereas the phases are directly derived from the clock signal, i.e. the clock needs to be routed to every SABL cell in a circuit. In the precharge phase the output nodes of a SABL cell are precharged to V_{DD}, in the evaluation phase one of the output nodes is conditionally discharged to GND, depending on the logic function of the cell and the input values.

The security of SABL mainly relies on two requirements: (1) logic cells having a highly constant internal power consumption and (2) on balanced complementary wires to ensure a data independent power consumption of each output node. Hence, the effort for implementing SABL is twofold: first, the logic cells need to be carefully implemented from scratch in order to avoid any data dependencies of the cells in terms of switching delay or power consumption. Second, during the physical layout of a SABL design the complementary wires of each cell have to be balanced to prevent a data leakage due to different wire capacitances.

3.1.2 RSL

Random switching logic (RSL) was introduced by Suzuki et al. [SSI04] and is based on specially crafted masking logic cells. Each cell only processes masked data values and has an additional enable input for initiating precharge and evaluation phases. In order to avoid a signal dependent switching behavior of the

RSL cells, the input signals at each cell need to be valid before the cell is enabled. This requirement results in a significant effort for correctly implementing the enable signal: each logic depth in an RSL circuit has to be provided with an enable signal having a specific timing. If the timing is too tight, input signals arrive after the enable signal resulting in a data-dependent switching behavior. On the other hand, if the timing of the enable signals is too relaxed, the maximum clock frequency of an RSL circuit might significantly be decreased.

3.2 Logic Styles Based on Semi-Custom Design

The second approach to implement PA-resistant logic styles is based on utilizing conventional standard-cell libraries. As one might say, the effort for breaking the strong relation between processed data within a digital circuit and the instantaneous power consumption is shifted to a higher level of abstraction, from the transistor level to the cell level. The main advantage of standard-cell based logic styles is that the underlying process technology can be switched without the need of expending huge effort. Also in case of a new revision of a previously used technology, probably only a few changes have to be performed, e.g. the names of logic cells have to be updated in the library describing the secure cells. Most semi-custom based logic styles implement so-called macro cells, which represent a compound of two or more basic logic gates. The goal of a compound gate can either be to equalize to randomize the power consumption, which corresponds to hiding and masking countermeasures, respectively.

In the following we briefly introduce one logic style of each category and discuss the basic working principles. As both logic styles were implemented during the work for this thesis, a detailed description of the logic styles MDPL and WDDL is given in Chapter 5 and in Chapter 6, respectively. Details on the realization of the logic styles using a semi-custom design flow are given in Section 3.3.

3.2.1 MDPL/iMDPL

The masked dual-rail precharge logic (MDPL) style was proposed by Popp and Mangard [PM05]. MDPL is based on standard cells, and hence, the implementation of MDPL does not require the design of new logic cells from scratch. As the name suggests, MDPL combines a masking technique with the dual-rail precharge (DRP) principle. The DRP technique avoids the occurrence of glitches and the masking breaks the relation between the intermediate values processed in an MDPL circuit and the instantaneous power consumption. As we will discuss in Section 5.1, it turned out that MDPL suffers from an effect called early propagation, which significantly reduces the security of an MDPL circuit. Hence, we implemented an improved version of MDPL, called iMDPL, which tries to prevent the occurrence of early propagation. Our implementation of iMDPL on a prototype chip as well as our extensive evaluation will be presented in Chapter 5.

3.2.2 WDDL/DWDDL

The wave dynamic differential logic (WDDL) style was proposed by Tiri and Verbauwhede [TV04a]. Similar to MDPL, WDDL is based on conventional cells available in a standard-cell library. WDDL is a pure DRP logic style that tries to keep the power consumption of a circuit constant, and thus, independent of the values processed. In order to achieve a constant power consumption, WDDL requires balanced complementary wires. During our work, we implemented an advanced version of WDDL, called DWDDL, which is based on two identically placed and routed WDDL circuits and does not require balanced wires. More details on the structure of our WDDL and DWDDL implementation on a prototype chip are given in Chapter 6.

3.3 The Implementation of Logic Styles in the Context of VLSI Design

A modern VLSI design flow includes several rather complex steps, and almost in each single step a specific EDA tool has to be used. There are basically two different ways to implement digital circuits: ASICs and FPGAs. With regards to the actual implementation of hardware circuits, the two approaches differ from each other mostly in the advanced steps of a design flow, where the design is tailored to the underlying hardware structures. In case of an application specific integrated circuit (ASIC) the hardware structure is customizable to a large extent: the placement and routing of modules and cells have to be performed by the designer, or more precisely by an appropriate EDA tool. Except for a list of design rules that have to be fulfilled, which is mostly done by the EDA tools, an ASIC designer has many degrees of freedom during the physical layout stage to adapt an ASIC to the application's needs. Contrary, a field-programmable gate array (FPGA) consists of a prefabricated array of configurable logic blocks (CLBs) embedded in a massive fabric of programmable switches, representing a configurable interconnect. In this case, the application needs to be adapted to fit an FPGA's predefined hardware structure. Everything that can be done on an FPGA can also be done on an ASIC. Note that the inverse does not hold true. With regard to implementing PA-resistant logic styles, the fixed structures on FPGAs are very restrictive and do not allow some required special steps.

Hence, during the work for this thesis we focused on the implementation of ASICs and the integration of PA-resistant logic styles in a semi-custom design flow. This is quite a complex topic since most logic styles require several non-standard design steps in order to be implemented correctly. Not only the common design rules have to be fulfilled, also all special requirements for achieving a certain level of security have to be met. For example, a SABL cell needs to have a constant power consumption independent of the input values, hence, the two complementary output nodes as well as the complementary input nodes need to have identical electrical characteristics (resistance, inductance, and capacitance). Variations between two complementary nodes would result in a data-dependent power consumption of the logic cell and thus in a decreased PA resistance. The implementation of secure logic cells might also include a careful placement of input and output pins during the physical layout phase of the cells. The pins might need to be arranged in an appropriate manner to enable the balanced routing of complementary wires to the subsequent logic cells.

In the following sections we will focus on two stages of a typical design flow which are essential for the implementation of PA-resistant logic styles: the logic design stage and the physical design stage. For each stage we discuss the typical design steps and introduce the special steps that were required to implement the iMDPL and the DWDDL style on our ASIC prototype chips. We also briefly discuss the special steps necessary for implementing full-custom based logic styles. The descriptions of the design flow stages in the following sections are partly based on [Kae08].

3.3.1 Logic Design Stage

In the logic design stage a high-level design is transformed into a gate-level netlist. Depending on the approach, either a standard-cell library or a full-custom cell library is utilized in order to bind the design to a specific process technology, e.g. using Faraday's cell library FSA0A_C [Far04] which is tailored for the UMC L180 GII logic process [Uni].

Full-Custom Design of Logic Cells

The implementation of a full-custom based logic style requires the execution of some preceding design steps. First, each logic cell has to be constructed based on a transistor network implementing the desired logic function. Next, the physical layout of the transistors within the logic cells has to be performed. If the cells are intended to be combined with a parallel routing approach, also a special placement/alignment of the input and output pins has to be considered.

Depending on the intended special properties of a cell (timing, power consumption), a designer possibly has to perform many iterations of physical cell layout and subsequent transistor-level simulations in order to verify the correct and secure behavior of each cell. Certainly, the construction of the cells has to be performed according to the design rules of the target process technology in order to guarantee[1] a fully functional chip in the end.

Logic Synthesis

The synthesis step itself can be performed almost in a straightforward manner: in most cases a reasonable subset of the standard cells is used for logic synthesis. This is especially the case for dual-rail based logic styles (full-custom based or

[1]Strictly speaking, there is no way to *guarantee* the fabrication of a fully functional chip. The design rules of a process technology are usually defined in a way that maximizes the chances of obtaining a flawless chip.

standard-cell based), as typical synthesizer tools are only able to handle single-rail standard cells. The standard cells are substituted by the dual-rail cells in a later step.

For each standard cell used in the subset an appropriate secure cell counterpart has to be available (for an exchange during the logic style conversion). In case of DWDDL and iMDPL we constrained the subsets to the following logic cells: INV, AND/NAND, OR/NOR, XOR/XNOR, buffers, and a few different types of flip-flops, e.g. flip-flops with or without asynchronous reset/preset. In many cases, only a few parts in a design are implemented in a secure logic style. In such a case the logic synthesis for the secure part of the design is constrained to this specific subset of cells, whereas the unprotected parts of the design can certainly be synthesized using all cells available in the library.

Netlist Conversion

The main step when implementing a secure logic style is the conversion from CMOS logic to secure logic. For this step we used a logic style conversion tool developed by Valentini et al. [VHUP05]. The so-called *netlist converter* takes a gate-level netlist as input and transforms each standard cell to a secure cell.

In case of DRP logic styles like DWDDL, iMDPL, or SABL the netlist conversion includes the following steps: (1) the substitution of standard cells with dual-rail cells, (2) the generation of complementary interconnect for the dual-rail part of each cell, and (3) the insertion of interface cells between the standard logic domain and the secure logic domain. The interface cells produce the complementary and precharged dual-rail input signals and also convert dual-rail signals back to conventional single-rail signals and remove the precharge phase. This way, DRP circuits can be easily combined with standard CMOS circuits. In case of iMDPL the netlist conversion step also includes the integration of a so-called *mask unit*, which produces the different mask signals required in an iMDPL circuit from a random mask value. After the netlist conversion, balanced signal trees have to be created for correctly distributing the different mask signals to the iMDPL cells.

3.3.2 Physical Design Stage

In the physical design stage the placement of all submodules, cells, and interconnect contained in the design is fixed on the chip die. The physical design stage consists of the following main steps, which are discussed in the following: floorplanning, placement, and routing.

Floorplanning

In a first step in floorplanning the dimensions of the chip die are fixed, depending on the complexity of the design and the number of I/O pads. In case of a highly complex design containing a vast number of logic cells but relatively few I/O pads, the *core area* containing all logic cells determines the actual size of the chip die, which results in a so-called *core limited* design. If the number of I/O pads is high in relation to the required core area, the placement of the I/O pads around the core area determines the size of the chip die, which results in a so-called *pad limited* design. Furthermore, the V_{DD} and GND rings around the core area as well as the V_{DD} and GND supply lines for the logic cells are created. Many designs also contain one or more V_{DD} and GND stripes running perpendicular to the cell supply lines in order to ensure a constant power supply of the whole digital circuit.

Partitioning: During floorplanning, the building blocks of the design can be assigned to particular regions in the core area, i.e. different blocks/submodules of a design can be strictly separated from each other. In this step, also a method called *partitioning* can be initiated, which is usually applied in case of very huge designs, with a complexity of several mega gate equivalents (MGEs), in order to keep the run times and the memory requirements of the physical design steps within a reasonable range. Partitioning also has the advantage that different partitions can be developed in parallel by different design groups. In case of partitioning, the design steps placement and routing described in the following first have to be performed separately for each sub-partition, then for the top-level partition. The timing information obtained during these subsequent steps is required for the clock-tree synthesis during the placement step. This allows a more accurate generation of the clock tree in the top-level partition.

The usage of the partitioning feature allowed us the realization of two identically placed and routed WDDL circuits which we later combined to one DWDDL circuit, as presented in Section 6.3.4.

Placement

During placement the position of each logic cell on the chip die is almost finalized. In subsequent optimization steps the placement can still slightly be changed. The placement step considers eventual constraints determined during floorplanning like regions or partitions. An important sub-step during placement is the clock-tree synthesis which makes sure that the clock signal is equally distributed to all sequential cells in a circuit. In an optimal case the clock signal arrives at exactly the same moment in time at each cell, i.e. the *clock skew* is 0.

In case of iMDPL the clock-tree synthesizer has to perform extra work. For example, in order to force the complementary signals in the precharge phase (clock is 1) to 0, the flip-flops in an iMDPL implementation contain two NOR gates connected to the clock signal. Hence, for iMDPL implementations the number of clock sinks is approximately tripled compared to an implementation in conventional CMOS logic.

Also in case of secure logic styles relying on balanced complementary wires, e.g. SABL and WDDL, special steps are necessary during placement depending on the implemented routing technique. For example, the *fat wire* approach proposed by Tiri and Verbauwhede [TV04c, TV05a, TV06] relies on diagonally arranged input and output pins of complementary logic cells. In such a case it has to be ensured that the logic cells are not placed mirrored in the design, which would alter the direction of the diagonally arranged I/O pins of the cells.

Routing

Until now, all connections between the logic cells in the design where just virtual. In the routing step these connections are realized with metal wires. Routing is usually performed in multiple steps, followed by several optimization iterations. During its work, the routing tool also checks for electrical shorts and design-rule violations and tries to resolve these.

For many PA-resistant logic styles the routing of the interconnect plays a major role. Pure DRP logic styles that rely on balanced complementary wires depend on highly sophisticated techniques to achieve identical electrical characteristics of two wires. The already mentioned fat wire approach requires several complex non-standard design steps during synthesis, placement, routing, as well as during the finalization of the layout [TV06]. Such techniques result in a significant additional effort for implementing a digital circuit.

The design steps that require modifications for implementing different PA-resistant logic styles are depicted in Figure 3.1. Implementing iMDPL mostly requires custom steps during and after logic synthesis, in our implementation DWDDL additionally requires the partitioning step during floorplanning. The implementation of WDDL, which relies on parallel routing, additionally requires custom steps during placement and routing. Additionally to these steps, a logic style based on full-custom cells and parallel routing like SABL also involves the effort for implementing and testing the logic cells. It is important to note that full-custom cells and parallel routing techniques are inevitably affected to a certain degree by variations occurring during the fabrication process.

During the work for this thesis we focused on the implementation of standard-cell based logic styles which do not solely rely on perfectly fabricated hardware structures and which do not require that many complex and expensive design steps. Our goal was to develop a rather easily implementable logic style that provides a certain level of security against PA attacks. Hence, we implemented iMDPL and DWDDL on two prototype chips.

Besides the correct implementation, a designer also has to analyze the functionality and the effectiveness of a logic style. In the next chapter we discuss different methods for verifying and evaluating PA-resistant logic styles on different levels of abstraction.

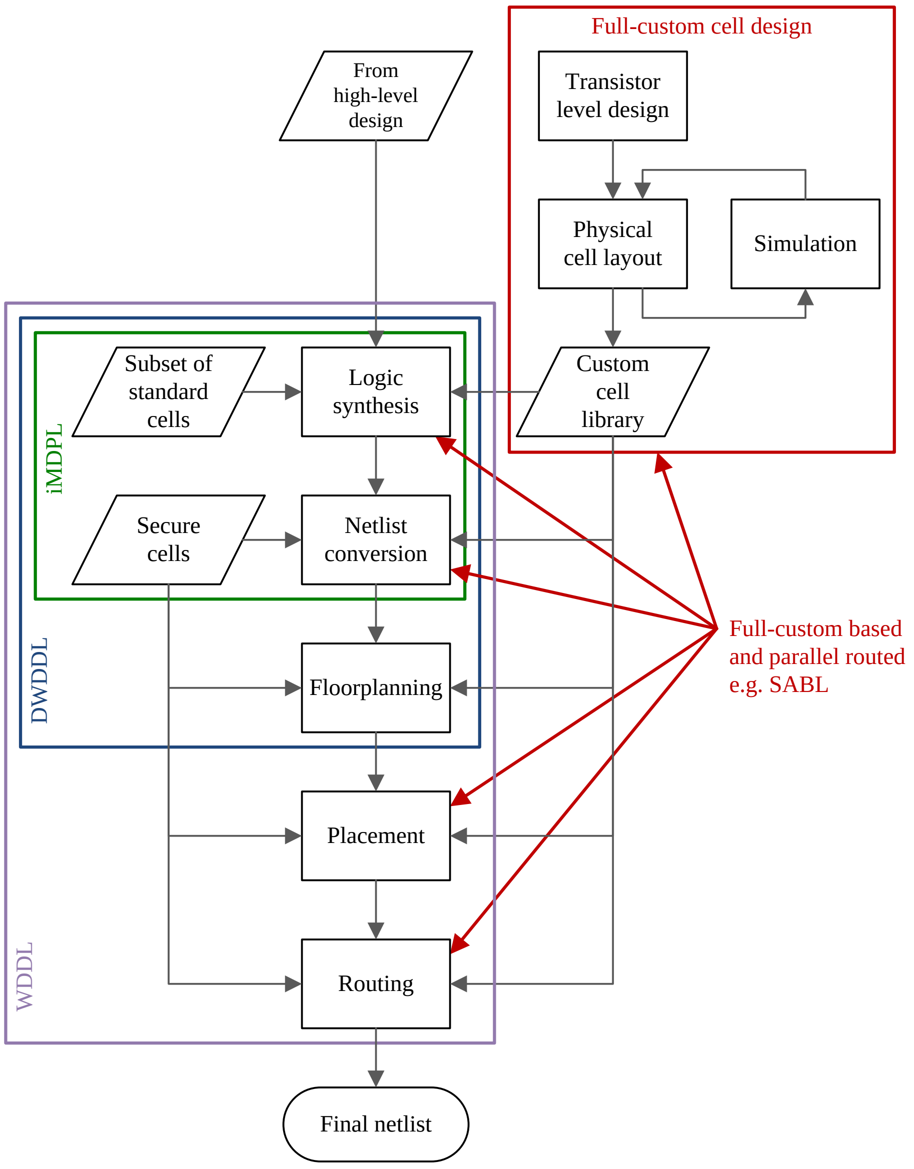


Figure 3.1: Modified steps for implementing PA-resistant logic styles in a conventional design flow.

4 Verification and Evaluation of Logic Styles

The next logical step after developing and implementing a PA-resistant logic style is its verification and evaluation. In this chapter we elaborate on different strategies to verify the correct functionality of an implemented logic style on different levels. We start with the theoretical investigation which is mostly based on experience gathered from other logic styles. Then, we discuss the possibility to evaluate special logic styles based on simulations on different levels of abstraction, the transistor level and the logic level. In doing so, we will also discuss the possibility to perform PA attacks on power traces derived from simulations. In the third and fourth section we discuss the assets and drawbacks of FPGAs and ASICs with reference to evaluating PA-resistant logic styles. Further, we point out the necessity of a comprehensive and reliable measurement and analysis setup and present some special features and optimization techniques we developed for our setup. In the last part of this chapter we elaborate on various exemplary applications of the presented evaluation techniques. We will discuss investigations performed and highlight potential drawbacks of individual methods.

4.1 Theoretical Investigation

The most simple yet in some cases very effective method for investigating a logic style's security is the theoretical investigation based on the fundamental design and operating principles of PA-resistant logic styles. The tools applied during this phase often are pen, paper, simple logic simulators like the Hades simulation framework [HBGL11], or basic spice simulators in order to verify the correct functionality of basic compound cells and logic gates. The effectiveness of theoretical investigations is mostly based on experience gathered from security evaluations of previously proposed and investigated logic styles and/or countermeasures in general, e.g. implementing a masking technique without preventing the occurrence of glitches should sound the alarm bells.

A theoretical investigation is also the starting point for developing a new PA-resistant logic style from scratch or for improving an existing logic style, regardless of whether the planned logic style is based on full-custom cells or not. Unfortunately, the advantageous simplicity of this investigative approach comes at a price: severe limitations when estimating the extent of many low-level effects that may occur in a final digital circuit. For example, the sheer magnitude of effects like imbalanced complementary wires, signal delays due to parasitic capacitances, or unforeseen differences between transistors or any other structures in a circuit due to process variations is highly unpredictable. The impact of many effects can only be assumed to a certain degree, the whole extent has to be evaluated by means of simulations and/or the production of a prototype chip. Nevertheless, later in Section 4.5 we will show that a theoretical investigation based on findings in other logic styles is able to reveal potential shortcomings also in recently proposed logic styles.

4.2 Practical Verification by Means of Simulations

Contrary to the very basic simulations of compound cells described in the previous section, in this section we focus on the simulation of complete digital circuits containing a few hundreds up to hundreds of thousands of logic gates. Such simulations play a major role when it comes to the development of a functioning integrated circuit. In order to guarantee the flawless operation of a chip, repeated testing and simulation throughout the whole VLSI design cycle is inevitable. In the event of errors occurring during simulations in a specific design phase, only the last changes need revision. If simulation errors occur after the transition from one design phase to the next, the performed operations in the design cycle need to be examined. Simulations on different levels of abstraction may also be used to estimate and to analyze the power consumption of a digital circuit. Furthermore, it is even possible to implement PA attacks on simulated power traces. In the following sections we elaborate on two approaches to estimate the power consumption of a digital circuit and discuss their suitability for verifying the effectiveness of PA-resistant logic styles.

4.2.1 Practical Verification at the Transistor Level

Transistor-level simulations, often referred to as SPICE simulations, represent the most accurate way of investigating many different aspects of an electronic circuit. The name SPICE originates from “Simulation Program with Integrated

Circuit Emphasis" and describes a tool developed by Laurence Nagel et al. in 1973, which was later adapted and further improved. Based on detailed descriptions of every component in a circuit, SPICE simulations provide very detailed information for each simulation time step about the voltage levels of each node in the circuit and the current flow between nodes by translating the whole circuit into non-linear differential equations and solving these equations. In case of VLSI designs, SPICE simulations are usually performed after finishing the physical design phase in order to maximize the accuracy of the simulations and to be able to make as precise assumptions as possible about a prototype chip that is possibly fabricated based on this particular design.

The very high accuracy of the power estimation abilities of transistor-level simulations make such simulations very suitable to be used in PA attacks, and hence to investigate the effectiveness of PA-resistant logic styles. The main drawback of such simulations is the high complexity and the associated tremendous effort for performing such simulations for larger circuits. For example, on an Intel® Xeon® Quadcore 2.33 GHz machine with 16 GB of RAM the transistor-level simulation of a few hundred clock cycles of a medium-sized digital circuit consisting of 250 kGEs (equals to approximately 1 million transistors) can easily last for a few hours. Considering the fact that hundreds or even thousands of simulated power traces might be required to perform a meaningful PA attack, transistor-level simulations turn out to be quite unsuitable for this task, unless a powerful computer cluster is available to perform the simulations.

However, transistor-level simulations are mandatory for designing a logic style that is based on full-custom cells. The correct functionality of the cells has to be verified in the first place. In case of a PA-resistant design, also the independence of internal node transitions, the power consumption of the cell, and the states of the input and output nodes has to be ensured.

4.2.2 Practical Verification at the Logic Level

Logic-level simulations target the outcome of the logic design phase in a typical VLSI design flow: a gate-level netlist bound to a specific process technology. Originally, logic-level simulations are performed to verify the correct functionality of a hardware implementation. Compared to transistor-level simulations, one of the main advantages of such basic simulations on logic level is the relatively high level of abstraction: intra-cell characteristics like signal delays between transistors, rise and fall times of output nodes, or capacitances of internal and external nodes are not considered. Thus, only simple logic functions (e.g. AND, OR, XOR, INV) need to be simulated which results in reasonable simulation times, even in case of designs with 500 kGEs and above.

The typical output of a conventional logic-level simulation only contains the signal transitions of the chosen parts in the simulated design. This output can be stored in form of a value change dump (VCD) file. A basic VCD file contains three parts: (1) a file header containing basic information about the simulation, (2) information about the probed signals, and (3) information about the signal transitions and corresponding simulation times. The more common type of

VCD files is a so called *four state* VCD file, where a signal can have one of the following four states: 0 (logic zero), 1 (logic one), x (unknown logic state), z (high-impedance state). The rather unusual *extended* VCD file format contains additional information about the signal states like the driving strength (weak, large) and the direction (input, output) of a signal. All of our experiments during the work for this thesis were based on standard four state VCD files.

The third part in a VCD file containing information about the signal transitions is more or less a list of simulation time markers and corresponding signal assignments. At the very beginning of the third part, the initial states of all signals are recorded. Then, the simulator starts a new entry at the current simulation time if at least one of the probed signals changes its state. Each entry contains a list of changed signals and their new values. Besides the possibility to create VCD files, many logic simulators provide different options for determining the signal delay of logic cells. In the following, we discuss three usual options: unit delay, zero delay, and back-annotated delay.

- Unit delay represents the standard approach for logic-level simulations. The simulator applies the same constant propagation delay for each logic cell in the simulated design which is typically 1 ns. In a unit delay simulation, the basic timing behavior of a design is reproduced correctly, i.e. timing differences between two input signals of a gate still occur. It is important to note that signals originating from equal logic levels have exactly the same timing, which might disguise timing issues in the investigated logic style.

- In zero-delay simulations all signal transitions caused by an initial event at time t_i are summed up at this particular simulation time t_i, which is mostly the clock event. As almost no timing information, except for the clock period, is included in the simulation results, the zero-delay mode potentially disguises major flaws in a secure design and is thus not recommendable for verifying logic styles. For example, severe effects like early propagation can not be identified. Hence, zero-delay simulations turn out to be rather impractical for the investigation of logic styles.

- The third possible timing option does not only involve the logic level: back-annotated delay simulations utilize additional timing information of a circuit obtained from advanced design steps, in most cases place and route. After the physical layout and the RC extraction of a design, timing data of interconnect and cells can be calculated and exported to a standard delay format (SDF) file. By means of SDF, a gate-level netlist obtained from previous design steps can be back-annotated. Many simulators are able to process SDF information, which results in a significant increase in accuracy of the simulation results in terms of timing behavior.

In conventional simulations the *toggle counting* of all signals in a circuit can be used to identify weak spots or faulty parts. If the requirement of a good input stimuli coverage is met, signals that hardly ever change their state in the circuit should be inspected in more detail [Kae08]. The technique of toggle counting can also be used to derive power traces from logic-level simulations ([KP07], [MOP07], [ST07]). In Section 2.1 it has been shown that the instantaneous power consumption of a CMOS circuit is strongly related to signal transitions at logic gates: in general it can be said that a signal change ($0 \to 1$, $1 \to 0$) causes a certain amount of power consumption, whereas a signal keeping its state ($0 \to 0$, $1 \to 1$) consumes a significantly lower amount of power. In its simplest form, the derivation of power-estimation traces from VCD files works the following way.

- At each simulation time marker in the VCD file where a signal transition ($0 \to 1$ or $1 \to 0$) occurs, the power-consumption value for this specific point in the simulation time is increased by 1.
- Constant signals ($0 \to 0$ or $1 \to 1$) do not contribute anything to the power consumption, the power-consumption value for this specific point in the simulation time is not increased.

Figure 4.1 depicts the process of toggle counting including all required ingredients. Starting point is a gate-level netlist obtained from synthesis in the logic design phase. The optional but advisable step of the physical design phase including RC extraction and delay calculation is indicated with dashed lines. The logic simulator, in our case Cadence NCSim [Cad11], takes the following parts as input data: (1) a gate-level netlist and a functional description of the cells from the library, (2) stimulus data generated with simulation scripts or an HDL test bench, and optionally (3) delay information of cells and interconnect. The output of the simulation, a VCD file, is then analyzed according to the rules described above. It is also possible to include back-annotated information obtained from the physical design step in the process of toggle counting: the information about output-node capacitances can be used to generate more accurate transition weights.

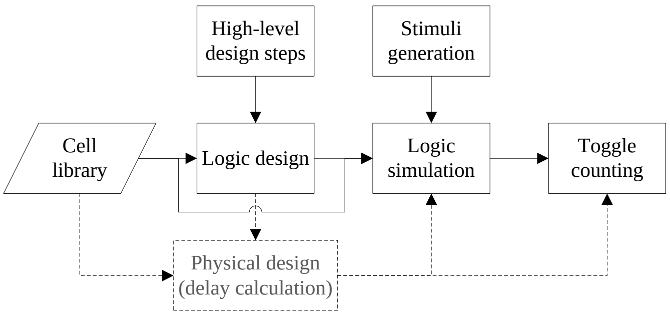


Figure 4.1: Toggle-counting process: the basis is a logic simulation of a gate-level netlist with optional back-annotation of delay information obtained from the physical design stage (indicated with dashed lines). The toggle-count trace is finally derived from summing up all signal transitions at each simulation time marker.

Our MATLAB SCA Toolbox [Ins] contains a tool called "VCD Analyzer" that allows us to derive toggle count traces by analyzing VCD files. The tool allows the extraction of toggle counts of arbitrary signals within each module and submodule that is covered by the logic-level simulation. Furthermore, the tool allows the definition of individual transition weights for each signal, which corresponds, for example, to slight differences between complementary wires in a DRP circuit. During the work for this thesis we enhanced our toggle counting approach and intensively utilized our VCD Analyzer tool to pinpoint the leakage in iMDPL (cf. Section 5.5.3).

An example of a toggle-count trace obtained from unit-delay simulations of an MDPL prototype chip is shown in Figure 4.2. We can see around 4 000 simultaneous signal transitions near the clock events. This is because in a unit-delay simulation all signal transitions happen only at a few different moments in time (multiples of $1\,ns$). The number of points in time where transitions may occur in a unit-delay simulation depends on the combinational depth in the simulated circuit: an advance from one logic depth to the next one (i.e. a signal propagating through a combinational cell) is accompanied by an advance in simulation time. In the small zoomed time range around simulation time 1.5 µs we can see the progression of the toggle-count trace in more detail: the shape of the trace is determined by the internal structure of the simulated circuit (e.g. clock tree, mask tree, combinational blocks). The following investigations of a simulated circuit based on the visual inspection of toggle-count traces and on simple calculations can easily be performed.

- Except for security-related issues, the visual inspection of toggle-count traces may already be used to determine a device's power consumption behavior. The occurrence of significant toggle-count peaks (as noticeable in the example in Figure 4.2) indicates a highly irregular power consumption of the design, possibly due to highly parallelized combinational blocks. This may indicate a reduced practicality of the design in low-power applications like passive RFID tags. Of course, in order to reach clarification precise power simulations, e.g., based on SPICE, would be necessary.

- In case of dual-rail precharge (DRP) circuits, the **total number of toggle counts** in each clock cycle should be constant per definition. An integration of the toggle counts over single clock cycles and a subsequent comparison could already reveal errors during the implementation of a logic style or a principal weakness in the design of a logic style. Such a simple comparison might also reveal some severe relations between processed data and the toggle counts.

- A visual inspection of the toggle-count traces may also reveal **timing issues** in the simulated circuit. Significant timing differences of signal transitions (shifting shape of the traces) may indicate the presence of combinational branches. Also weaknesses in the implementation of countermeasures based on the random insertion of dummy operations or operation shuffling may be discovered.

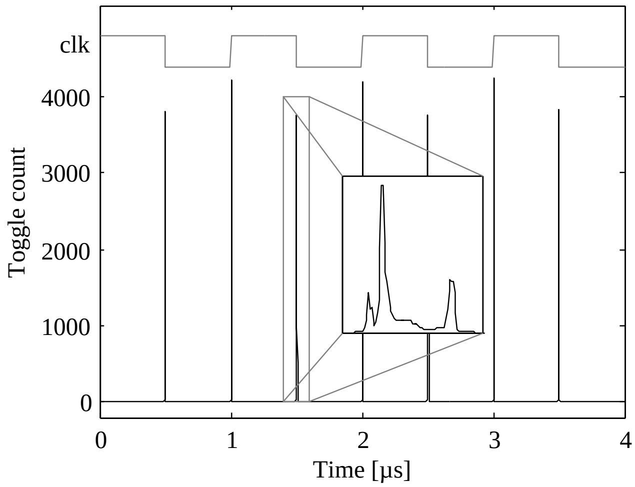


Figure 4.2: Example of a toggle-count trace derived from an MDPL circuit. The signal-transition activity at the clock events can be clearly seen as well as the progression of the toggle-count trace in the zoomed area.

The verification of the latter two issues by means of visual inspection is easily possible in the presence of severe flaws in a design. However, this is not the case if, e.g., only a very small portion of the whole design suffers from faults or only a certain cell type rarely used in the circuit contains minor errors. A very comprehensive evaluation of a design can be done by performing PA attacks based on toggle-count traces. In contrast to commonly known PA attacks based on the measurement of an actual device (cf. Section 2.2), we can also use a set of toggle-count traces to perform such attacks. A major advantage of attacks utilizing toggle-count traces is the possibility to focus on single modules of a design. This approach is similar to the EM measurement of a specific area of a chip's surface and allows the suppression of noise generated by other modules in the design.

The main advantage of logic-level simulations is their higher level of abstraction compared to transistor-level simulations. This comes at the price of a significantly reduced accuracy of the simulations in terms of simulated power consumption values, signal timing, as well as in terms of cell internal processes. By additionally utilizing back-annotated delay information the accuracy of the simulated signal timings can be greatly increased, which entails only a slight increase in simulation time. A further increase in terms of validity of the results obtained from PA attacks can be achieved by utilizing advanced toggle weighting techniques like random weighting or back-annotated weighting. All things considered, the toggle counting method based on logic-level simulations represents a very powerful tool for investigating and evaluating PA-resistant logic styles at a convenient level of abstraction. The simulations provide meaningful results due to the close relation to the target CMOS technology and can be performed in a reasonable time frame by the abstraction of transistor-level processes.

4.3 Practical Verification by Means of FPGAs

Before actually fabricating an ASIC prototype chip, it is often reasonable to extensively test the correct functionality of a hardware design. As introduced in the previous sections, simulations on different abstraction levels can be performed for this purpose. Between simulations and the fabrication of an ASIC there is a further possibility to implement a design, perform reliable functional tests, and also to investigate the PA resistance to a certain degree.

FPGAs provide the possibility to implement and to execute almost arbitrary hardware designs besides providing the flexibility of a pure software approach, namely a fast development and low costs. Furthermore, FPGAs provide the high-performance advantage known from pure hardware approaches. All things considered, FPGAs enable rapid and cheap prototyping besides easy reprogrammability.

FPGAs are highly suitable for evaluating many types of countermeasures on different levels. In [TKS10], we published a comprehensive approach for implementing a PA-resistant embedded processor based on a Leon3 processor (presented in detail in Chapter 6). The practical results of the processor protected with architectural masking as well as a preliminary leakage evaluation based on EMA attacks were realized using a Virtex 4 FPGA. With regards to evaluating cell-level countermeasures like special logic styles, the applicability of FPGAs is definitely restricted: logic styles relying on a special internal structure or on particular characteristics of the underlying logic cells can not be implemented on an FPGA. For example, the logic style SABL relies on logic cells having a constant internal power consumption. Such a behavior can not be reproduced by utilizing the lookup tables on an FPGA. Furthermore, the implementation of special logic styles on an FPGA requires additional effort for manipulating the synthesized netlist as discussed for example in [TV04a, YS07]. However, the approach is similar to our procedure for implementing (i)MDPL and WDDL on ASIC (cf. Section 3.3). In Section 4.5 we will refer to some examples of logic style evaluations which are based on the implementation of hardware designs on FPGAs.

4.4 Practical Verification by Means of Prototype Chips

In the following we want to elaborate on three main aspects of fabricating prototype chips in the context of PA-resistant logic styles: production costs and effort, significance of obtainable evaluation results, and performance in terms of the required time to obtain meaningful results. Subsequent to these discussions about prototype chips we introduce the measurement and analysis setup we built up and steadily optimized during the work for this thesis.

Costs and effort: At a first glance the production and the subsequent evaluation of a prototype chip is the most expensive way in terms of time, effort, as well as in terms of monetary expenses. However, a closer look significantly mitigates these statements: let us assume a designer who is intensively working on the development of a PA-resistant logic style. Given the fact that she is working in this area, we further assume that the designer already has detailed knowledge about VLSI design in general and that she has access to the necessary tools and process technology data as well as a comprehensive measurement and analysis environment. The basic design of a logic style, provided that the work is not solely performed in a theoretical way, most probably already includes low level design processes like cell design, physical layout, and transistor-level simulations. We might say, a designer being able to perform these practical steps has a fully adequate VLSI design flow at her disposal. From a financial viewpoint we can state that the license expenses for all required tools are already covered. Roughly speaking, starting from back-annotated logic-level simulations or transistor-level simulations of a design, the actual production of a prototype chip is only two steps away: (1) placing I/O pads in the design and (2) bearing the relatively low costs of a multi-project wafer (MPW) run.

Significance of results: Against all the odds, the production and the subsequent evaluation of a prototype chip is not necessarily the most informative way of verifying the correct functionality of a logic style. The significance of the achievable results strongly depends on a reasonable design implemented on the prototype chip. In Section 2.3 we discussed that a highly parallelized hardware design may have significant influences on the quality of the measurements of a device due to the increased amount of switching noise. This is also the case if a secure module implemented in a PA-resistant logic style is surrounded by other rather complex hardware modules. Huge amounts of noise can effectively disguise flaws in a logic style.

As a first example for this problem we want to refer to our iMDPL prototype chip, which is discussed in detail in Chapter 5: among other parts our chip contains an implementation of an 8051-compatible microcontroller with an AES coprocessor implemented in the iMDPL style. Additionally, exactly the same AES module has been implemented in a standalone version in the same logic style. Hence, the only difference between these two instances of AES hardware is the amount of switching noise caused by the microcontroller during its operation. As we will see in Chapter 5, iMDPL causes a considerable increase in hardware complexity, and hence, the amount of switching noise is significantly increased. The results of PA attacks on both modules have shown that the standalone AES module can be attacked in a straightforward manner, whereas the attack results on the AES coprocessor attached to the microcontroller do not show any distinct leakage. In case we only had analyzed the AES coprocessor we would have falsely stated that iMDPL provides perfect security.

We want to point out another example based on a prototype chip published by Tiri et al. [THH^{+}05]. The authors fabricated a chip in a 180 nm CMOS process technology consisting of an AES module, a fingerprint matching engine, and some memory. The modules were implemented in CMOS logic and in WDDL in two separated cores on the chip. The results of correlation-based PA attacks on the AES module implemented in WDDL revealed a rather high resistance compared to the unprotected implementation in CMOS logic. A closer look at the architecture of the evaluated AES module reveals that the module was based on a highly parallelized high-speed implementation proposed by Hodjat et al. [HHL+05]. In this implementation each round of AES is performed within a single clock cycle, i.e. the AES module already produces a huge amount of switching noise additionally to the other two modules. With reference to the fact that WDDL suffers from early propagation [SS06], such results have to be treated with caution. It is very likely that the increase in PA resistance due to the WDDL style is significantly lower, i.e. most probably the PA resistance is due to the increase in switching noise for the most part.

These examples show that it is very important to investigate the effectiveness of a logic style based on a reasonable design. Other effects, e.g., an increased amount of switching noise, could significantly falsify the evaluation results of a particular device. Furthermore, the evaluation of different hardware implementations could lead to considerably different results.

Evaluation performance: Although we stated that logic-level simulations already represent a very fast approach of obtaining toggle-count traces, the acquisition of measurement data from a real chip is by far the fastest possibility. Besides the effort in terms of time, the major difference between simulated traces (obtained either from transistor-level simulations or logic-level simulations) and measured traces is the amount of electronic noise: during the measurements, several noise sources may significantly affect the quality of the measurements (cf. Section 2.2.1), which may result in highly disturbed or even in unusable power or EM traces. Similar to the effect of switching noise caused by an inappropriate hardware design, electronic noise may also falsify the evaluation results of a prototype chip. Furthermore, the evaluation of real measurements can mostly reveal the basic existence of a leakage. Discovering the exact source of a leakage solely by means of measurements is only possible to a certain degree. Special hardware structures implemented on a chip may help to discover some specific flaws (as shown in Section 5.5.4), but in some cases only simulations may be able to completely reveal the source of a leakage in an implementation. The evaluation speed of a prototype chip also strongly depends on the performance of the measurement and evaluation setup. During the work for this thesis we permanently optimized our measurement and evaluation setup based on MATLAB, which we briefly introduce in the next section followed by the presentation of realized optimization techniques.

4.4.1 Measurement and Analysis Environment

A typical power/EM measurement and analysis setup as it is depicted in Figure 4.3 consists of the following parts: a host PC, a digital oscilloscope, and the device under test (DUT) operated on an appropriate evaluation board. In an optimal case, the oscilloscope and the DUT are fully controlled via the host PC, which then has the following tasks: (1) communicate with the DUT, for example, over a serial, parallel, or USB connection, (2) control the digital oscilloscope and acquire the measured traces via GPIB, LAN, or PCI (in case the oscilloscope is directly integrated into the host PC), and (3) perform statistical analysis of the measured traces. We usually assume an advanced attacker who has a little additional information in the form of a trigger signal, i.e. we assume that an attacker knows the time when the DUT starts executing the target operation (e.g. an AES encryption).

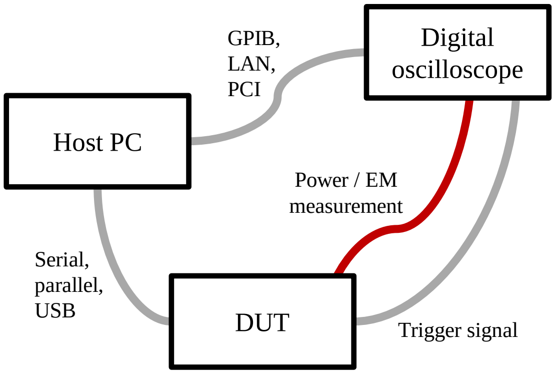


Figure 4.3: A typical power/EM measurement setup consisting of a host PC, a digital oscilloscope and the device under test.

When we started our work for this thesis, our original measurement setup was based on a slightly aging LeCroy LC584 digital oscilloscope [LeCa]. The oscilloscope was capable of communicating via GPIB and we used a GPIB to LAN converter for control and acquisition, which represented a major bottleneck in this measurement setup. The measurement performance, we usually denote it as measured traces per hour (tph), strongly depends on the length (number of samples) of the recorded traces and also slightly depends on the device and the required communication for each measurement run. We made the best of this setup and were able to achieve measurement rates around 25-40 ktph, whereas the length of the traces was around 15 000 sampled data points. Considering the fact that some of our latest measurement experiments comprised more than 17 million traces with 50 000 sampled data points (cf. stepped EM measurement in Section 2.4.3), we would have required weeks in order to perform such measurements.

Not long ago we acquired a new LeCroy WavePro 725Zi digital oscilloscope [LeCb]. The main advantage of the new oscilloscope is that the oscilloscope hardware itself is directly integrated into a host PC, backed by an Intel® Core ™ 2 Quad processor. Hence, the bottleneck of transmitting the measured traces is completely canceled out as the traces can be stored directly to a mass storage device with a very high performance. Our existing measurement routines were not nearly able to fully exhaust the new throughput potential, so we had to optimize other parts of our measurement setup.

During the work for this thesis we evaluated plenty of different devices. Each device had its own peculiarities, strengths, and weaknesses in terms of operational complexity, measurability, or stability. Those devices being concerned with protection measures had one thing in common: the demand for a high-speed measurement setup in order to perform the required number of power or EM measurements in a reasonable time. The term "required number of measurements" represents the recording of a sufficient amount of traces (power or EM) from a device to be able to make clear statements about the actual security level of the particular protection measure implemented on that device. In the following we briefly elaborate on practical measurement optimization techniques we developed during the work for this thesis. We present the fundamental idea behind the approaches and provide improvement factors based on practical observations.

Optimizing the Communication

As our measurement and evaluation environment is fully based on IAIK's Side-Channel Analysis Toolbox (SCA Toolbox) [Ins] for MATLAB, we also handle the communication with the devices under test in MATLAB. The serial interface is a very simple and common interface for communicating with small devices, especially for communicating with self-built prototype chips. During our measurements of various devices, we repeatedly encountered unrecoverable communication errors over the serial interface, which resulted in the need for discarding the actual measurements and restarting the whole measurement process. During the work for this thesis we developed a strategy to significantly improve the communication stability of the serial interface within MATLAB. The MATLAB serial object is based on Java, so we decided to implement a serial interface using basic C functions. We realized our serial communication toolbox with a daemon application called *rs232_daemon* running in the background, which handles the actual serial port using C functions. The communication interface between the daemon application and MATLAB is based on inter-process communication using a named pipe. For each communication attempt from MATLAB the named pipe is called, the corresponding data is transmitted to the rs232_daemon and the connection to the named pipe is closed again. It turned out that this approach resulted in a 100% stable communication with all our devices, we never encountered any communication errors again. Furthermore, with this approach we were able to speed up the communication over the serial interface by a factor between 2 and 3.

Optimizing the Trace Acquisition

Our new LeCroy WavePro 725Zi digital oscilloscope opened up new traces-acquisition possibilities. In a first step, we developed a comprehensive MATLAB toolbox to enable the complete remote control of the oscilloscope. The toolbox is able to utilize OLE Automation, an inter-process communication channel based on COM, in order to directly access the oscilloscope's COM objects, which results in a very fast control interface from within MATLAB.

In order to further speed up the acquisition of the recorded traces we utilize the *sequence mode* of the oscilloscope in combination with storing the recorded traces directly to a mass storage device: consecutive trigger events are stored sequentially in one single trace, together with the information about the beginning and the end of each single event. This way the oscilloscope stores hundreds or even thousands of traces into one large file contrary to storing thousands of small files, which saves a lot of time.

Optimizing the Throughput of the Device Under Test

After intensively working on optimizations of the communication with the DUT and the acquisition of traces we also improved the throughput of our devices in terms of computations per time unit. A possible solution was to minimize the communication overhead for each measurement run. In case of microcontrollers and processors we implemented a batch mode: the device receives an initial value and automatically performs a certain number of computations, for example AES encryptions. The ciphertext obtained from one encryption run can be directly used as plaintext for the following encryption run, which is referred to as output feedback mode (OFB). The plaintext for the next encryption run can also be generated from the initial value and a counter value.

By means of our work on the discussed components we were able to tremendously increase the overall performance of our measurement setup. Taking the measurement performance of around 40 ktph from our previous setup as a basis, we were able to increase the overall performance by a factor of >100. At the moment, our fastest acquisition setup achieves approximately 5 to 6 Mtph, which is equal to approximately 1 500 tps. Considering the specification of the oscilloscope there is still some potential left, as the maximum trigger rate is indicated with 1 Mtps.

A faster measurement setup not only reduces the required time to obtain practical results from a prototype chip and hence makes the measurements "more pleasant", a significant reduction of the measurement runtime also minimizes long-term effects on the measured traces like temperature drift (concerns for the example power supply, the prototype chip, and the measurement probe) or other possible events that may severely disturb the measurement process.

4.5 Exemplary Application of the Evaluation Techniques

In the following we elaborate on the application of the presented evaluation techniques by means of a few examples. First we will discuss the results of simulations and evaluations on SABL cells performed by different parties [TAV02, TV03, TV04b, SA05, TV05b, KKT06]. We then discuss the results of Tiri and Schaumont, who performed toggle count simulations on an RSL circuit [TS07]. Subsequently, we present our theoretical investigation of the DRSL style, a successor of RSL, proposed by Chen and Zhou [CZ06], as well as the evaluation results based on DRSL implementations on an FPGA [SS08a]. Based on our investigation of the DRSL style, we briefly discuss possible issues of MDPL and iMDPL. Both logic styles will be discussed in detail in Chapter 5. We also mention the security evaluation of MDPL based on an FPGA [SS08b]. In the sections 4.5.5 and 4.5.6 we provide theoretical investigations on TDPL [BGLT06] and DDPL [BGL$^+$11], respectively.

4.5.1 SABL

As introduced in Section 3.1.1, SABL is based on full-custom cells that try to achieve a constant internal power consumption, independent of the processed values. In the original publication [TAV02] the authors compare the effectiveness of different SABL gates (e.g. INV, NAND) with static complementary CMOS gates based on the energy consumption per cycle. We note that a comparison based on the energy consumption over a whole evaluation-precharge cycle potentially disguises differences in the instantaneous power consumption. In [TV04b] the authors of SABL proposed an advanced version of the logic style, charge recycling SABL (CRSABL), which reduces the overall power consumption of the cells besides preserving the energy masking effect of SABL. A comparison of different logic styles in [SA05] did not confirm the original findings of the authors of SABL. Tiri and Verbauwhede also proposed an improved version of SABL [TV05b] that prevents early propagation. A study on SABL and the improved version of SABL published by Kulikowsky et al. [KKT06] confirmed that the original version of SABL suffers from early propagation, whereas improved SABL successfully prevents early propagation. These examples show that SPICE simulations are very suitable for evaluating the effectiveness of a logic style based on full-custom cells. However, on the basis of such simulations only it remains difficult to estimate the effectiveness of such a logic style on real silicon.

4.5.2 RSL

We introduced the basic functionality of RSL in Section 3.1.2. RSL is a single rail masked logic style proposed by Suzuki et al. [SSI04]. Tiri and Schaumont [TS07] discovered that the mask value of RSL can be derived by simple power analysis, which significantly reduces the effectiveness of the logic style. The authors performed their experiments with the cycle-based simulator GEZEL [SV02] and obtained toggle-count traces from a test circuit implementing the AES S-box transformation and the AddRoundKey function. The results show that a non-switching mask value ($0 \rightarrow 0$, or $1 \rightarrow 1$) causes a significantly different toggle count that a switching mask ($0 \rightarrow 1$, or $1 \rightarrow 0$). The authors from [TS07] also correctly stated that if the vulnerability of an implementation can be shown using an inaccurate technique like toggle counting, the vulnerability will definitely not be eliminated if the evaluation is performed with an even more accurate technique like SPICE simulations or the prototype measurements. This example shows that toggle-count traces derived from basic logic-level simulations already represent a powerful tool to evaluate the effectiveness of logic styles to a certain degree.

4.5.3 DRSL

Chen and Zhou proposed an advanced version of RSL, which is called dual-rail random switching logic (DRSL) [CZ06], to eliminate the need for the quite complex generation of the enable signal for each RSL gate. A basic DRSL NAND gate consists of two conventional RSL NAND gates and a precharge detection unit, which is very similar to the phase detection unit in the iMDPL style (cf. Section 5.3, Figure 5.6). In the evaluation phase each RSL gate remains deactivated until all input signals are in differential state. Due to the fact that the enable signal for the RSL gates is derived from the input signals by means of a small combinational circuit (OR-AND-INVERT), the enable signal is produced shortly after the last input signal reached its differential state. Hence, the RSL gates are only activated if all input signals are valid. However, in the precharge phase the timing conditions are different: if the first signal leaves its differential state, the combinational logic producing the enable signal opens a small time frame. Within this time frame the first signal precharged may also cause a switching of one of the RSL gates before the precharge detection unit is able to deactivate (precharge) the RSL gates. This effect may result in a data dependent switching of the DRSL cells in the precharge phase. Considering the fact that the mask values $m/\overline{m}$ in a DRSL circuit is provided by a signal distribution network similar to the clock tree in a digital design, we can assume that the precharge of the DRSL gates mostly depends on the precharge timing of the mask. Furthermore, as DRSL does not require balanced complementary wires the two mask trees most probably show different timings on arrival at the DRSL cells, which most probably results in the detectability of the mask due to recognizable differences in the power consumption of a DRSL circuit.

Saeki and Suzuki have also performed a security evaluation of DRSL [SS08a]. Their experiments were based on FPGA implementations of basic DRSL cells on an FPGA. By means of power measurements of the FPGA they discovered that timing differences between input signals may cause a leakage in a DRSL circuit. In this particular case an FPGA having four-input lookup tables could be used to recreate the functions of basic DRSL gates.

4.5.4 MDPL and iMDPL

Similar to DRSL, the logic styles MDPL and iMDPL are also based on a single mask bit and they also do not rely on balanced complementary wires. In both MDPL and iMDPL the masks for each logic cell are distributed similar to the clock signal using a signal distribution network. Slightly different timings of the complementary mask trees probably results in a mask dependent switching behavior of an MDPL/iMDPL circuit. In Section 5.5.3 we will present the results of a detailed evaluation of an iMDPL circuit based on real measurements and back-annotated logic-level simulations. The results show that unbalanced complementary wire trees may introduce significant differences in the power consumption which results in the vulnerability of such logic styles.

Suzuki and Saeki also performed security evaluations of MDPL based on the implementation of basic MDPL cells on an FPGA [SS08b]. Their results verified that different arrival times of the input signals at MDPL gates may cause a significant vulnerability in an MDPL implementation. Also in case of MDPL an FPGA could be used to reveal flaws in the design of the logic style.

4.5.5 TDPL

The three-phase dual-rail precharge logic (TDPL) style was proposed by Bucci et al. [BGLT06] in 2006. Besides the common precharge and evaluation phases, TDPL introduces a third phase, the discharge phase in each clock cycle. The third discharge phase ensures that the energy consumption of a TDPL cell is constant in each precharge-evaluation-discharge cycle: each output line is precharged and discharged exactly once in each cycle.

In [BGLT06] the authors of TDPL perform a security evaluation of their logic style based on the total energy consumed per cycle, whereas one cycle consists of the three phases precharge, evaluation, and discharge. An important issue first noticed about the TDPL style is the generation and the distribution of the three separated control signals required for introducing each phase, which results in a significant effort. Furthermore, the control signals need to have a specific timing in order to prevent early propagation: the input signals of the previous logic stage need to arrive during the active precharge phase, otherwise the TDPL cells switch at data dependent times.

The security evaluations in [BGLT06] are based on transistor-level simulations of a TDPL AND/NAND gate and the simulated current traces show significant fluctuations in the evaluation and in the discharge phase, which we examined more carefully based on the following theoretical investigation. First, the TDPL style is based on the dual-rail principle, i.e. each TDPL cell has two complementary output wires. Second, the authors claim that the logic style does not require balanced complementary wires, i.e. the electrical characteristics of both output wires of a TDPL cell are totally random. It is true that precharging and discharging both wires during one cycle always consumes the same amount of energy, but the discharging of the output wires does happen at different moments in time: one wire is discharged in the evaluation phase, the other wire is discharged in the discharge phase. In other words, depending on the output value of the TDPL cell two different wires switch their states at two different points in time. Hence, it is very likely that the current fluctuations in the evaluation and in the discharge phase recognizable in the author's simulation results correlate with the input data of the logic cells, which results in a significant vulnerability to PA attacks.

4.5.6 DDPL

The delay-based dual-rail precharge logic (DDPL) style [BGL+11] is based on the DRP principle and does not require balanced complementary wires. The idea of DDPL is to encode the processed data values in a circuit in the time domain, i.e. the delay between the switching events of two complementary wires $a/\overline{a}$ determines the value v transferred: a positive delay $t_{\overline{a}} > t_a$ represents $v = 1$ (i.e. the complementary wire $\overline{a}$ switches to 1 some time after the direct wire a), whereas a negative delay $t_a > t_{\overline{a}}$ represents $v = 0$. The authors of DDPL state that this way all complementary wires are precharged and discharged within each operating cycle, which results in a constant amount of energy consumed (per cycle). The security evaluation of DDPL presented is based on transistor level simulations, and the delay value was determined with 1 ns. The simulation results clearly show two distinct current peaks in the evaluation phase, separated by the specified delay. The authors state that this delay has to be resolved by a power measurement in the first place in order to detect a data dependency in a DDPL circuit.

Our evaluations on an MDPL circuit based on transistor-level simulations discovered that the MDPL cells switched at data dependent times due to early propagation [Kir07, PKZM07] (the results will be presented in detail in Section 5.2). The simulation results showed that the data dependent delay was in the range of 0.8-1 ns. In real measurements of our MDPL prototype chip this delay caused a significant vulnerability to PA attacks. Hence, it is very likely that the PA resistance of a DDPL circuit may be considerably reduced due to the data encoding in the time domain.

It turns out that each of the discussed evaluation techniques has its assets and drawbacks. Depending on each particular case, a designer has to choose an appropriate method to evaluate the implemented countermeasure. It also turns out that a reasonable combination of the discussed evaluation techniques most likely results in a thorough investigation of an implemented logic style and the complete discovery of potential weaknesses. For example, after discovering a leakage by performing measurements of a prototype chip, the next logical step is to pinpoint the leakage by means of simulations. Focusing on single submodules in a huge design helps to rule out unaffected modules in a first simulation step. Figuratively speaking, we significantly reduce the effort of finding the needle in the haystack, i.e. we discard piece by piece of the haystack without discarding the needle. In the next step, we can reproduce the effects discovered in the simulation results by means of a theoretical investigation. The further course of action possibly initiates a restart of the loop: (1) improvement of the current logic style followed by a theoretical investigation, (2) verification by means of simulations, and (3) production of another prototype chip and evaluation of the improvement. In the next chapter we present our evaluations of MDPL and iMDPL following exactly this approach.

5 Evaluating the Improved Version of the Masked Dual-Rail Precharge Logic Style

In the previous three chapters we introduced PA attacks and presented techniques for the implementation and the evaluation of special logic styles resistant against such attacks. In this chapter we provide a thorough investigation of the iMDPL style. First, we introduce the original MDPL style and present our evaluation results of MDPL on the SCARD [SCA05] chip. Most of the evaluations were performed during the work for the master thesis [Kir07] and were published in a joint work together with Thomas Popp, Thomas Zefferer, and Stefan Mangard in [PKZM07].

Based on the evaluation results of MDPL, we then present an improved version of the logic style, called iMDPL, which was developed during the work for this thesis together with Thomas Popp. We justify our decisions and provide a detailed investigation based on an improved logic cell. Subsequently, we present an iMDPL prototype chip which we designed in the course of the GRANDESCA [GRA07] project. We first present the architecture of the chip consisting of an 8051-compatible microcontroller core with an AES coprocessor as well as a standalone AES core, whereas both cores were implemented in CMOS logic and in iMDPL. Then, we present the results of the PA attacks that were performed on the different cores on the GRANDESCA chip. The results are based on a joint work together with Thomas Popp and were published in [KP09].

As it turns out, our improvement of the MDPL style significantly increases the resistance against PA attacks. Nevertheless, the iMDPL style still showed a data leakage which was thoroughly investigated in a joint work together with Amir Moradi, Thomas Eisenbarth, and Christof Paar. The results were published in [MKEP]. Our investigations were based on power and EM measurements as well as on logic-level simulations of the GRANDESCA chip. We present the results of our investigations and show that the PA resistance of iMDPL suffers from imbalanced complementary wires. Furthermore, we show that there is a strong mask-dependent switching behavior of the iMDPL cells, which causes a severe leakage of information about the mask value and enables the extraction of the mask value directly from the power traces. We verify these results by means of a special mask-observation feature of the GRANDESCA chip. In the last part of this chapter we draw conclusions about the implemented iMDPL style, the performed experiments, and the discovered problems.

5.1 The Original MDPL Style

The masked dual-rail precharge logic (MDPL) style was proposed by Popp and Mangard [PM05] in 2005. MDPL was developed in order to obtain a logic style that (1) provides a certain level of resistance against PA attacks, (2) is based on conventional cells available in a standard cell library (contrary to full-custom based logic styles, cf. Section 3.1), and (3) does not require any additional constraints (e.g. balanced routing, special signal timings) for a correct implementation. The MDPL style tries to break the dependency between the processed data within a digital circuit and the instantaneous power consumption by applying a random mask bit m to every data value v in the circuit. MDPL implements a Boolean masking approach where an exclusive-or (XOR) operation is used to mask every data value: $v_m = v \oplus m$. Hence, the power consumption is independent of v which fully prevents PA attacks. This holds true under the assumption of an ideal behavior of a digital circuit where no glitches occur during the propagation of signals through the circuit. In [MPG05, MPO05, SSI07] it was shown that glitches may significantly reduce the effectiveness of a masking approach, besides having a major influence on the overall power consumption of a circuit (cf. Section 2.3.3).

Table 5.1: Truth table of a basic AND function (columns A-C, rows 1-4) and the realization of an MDPL-AND function (columns D-K, rows 1-8).

Column →	A	B	C	D	E	F	G	H	I	J	K
Row ↓	a	b	q	m	a_m	b_m	q_m	$\overline{m}$	$\overline{a_m}$	$\overline{b_m}$	$\overline{q_m}$
1	0	0	0	0	0	0	0	1	1	1	1
2	0	1	0	0	0	1	0	1	1	0	1
3	1	0	0	0	1	0	0	1	0	1	1
4	1	1	1	0	1	1	1	1	0	0	0
5	0	0	0	1	1	1	1	0	0	0	0
6	0	1	0	1	1	0	1	0	0	1	0
7	1	0	0	1	0	1	1	0	1	0	0
8	1	1	1	1	0	0	0	0	1	1	1

In order to avoid the occurrence of glitches, MDPL is based on the dual-rail precharge (DRP) principle. A pure DRP logic style is a hiding countermeasure that tries to keep the power consumption of a digital circuit constant in each clock cycle (cf. Section 2.3.3). DRP logic also prevents the occurrence of glitches by introducing a precharge and an evaluation phase in each clock cycle and by utilizing positive monotonic Boolean functions only. Each signal in a DRP circuit is represented by two complementary wires d and $\overline{d}$. In the precharge phase, both wires d and $\overline{d}$ are 0. Depending on the logical value of the corresponding signal either d or $\overline{d}$ switches to 1 in the evaluation phase. This glitch-preventing feature of DRP logic is utilized by the MDPL style in order to guarantee that the masking technique is not affected by the occurrence of glitches.

In the following, the internal structure of a basic MDPL AND cell is discussed. The truth table of an AND function $q = a \wedge b$ is shown in Table 5.1, columns A-C, rows 1-4. In case of MDPL an AND cell has six input values $a_m, b_m, m, \overline{a_m}, \overline{b_m}, \overline{m}$ and two output values $q_m, \overline{q_m}$. The truth table of an MDPL AND cell is shown in Table 5.1, columns D-K, rows 1-8. As each data value in an MDPL circuit is masked, the values a_m, b_m, q_m (columns E-G) can be derived from XORing the unmasked values a, b, q (columns A-C) with the mask m (column D). The values $\overline{m}, \overline{a_m}, \overline{b_m}, \overline{q_m}$ (columns H-K) represent the complementary values to m, a_m, b_m, q_m. It turned out that q_m and $\overline{q_m}$ can be calculated by the majority (MAJ) function which has three inputs and one output: the output of this function is 1 if at least two inputs are 1, otherwise the output is 0. An MDPL AND cell can hence be built with two MAJ gates (*MAJ I*, *MAJ II*) as depicted in Figure 5.1 (left). The mask value controls which MAJ gate calculates the actual AND function and which gate provides the complementary values only:

- In case $m = 0$ (Fig. 5.1, center), *MAJ I* calculates the AND function $q_m = a_m \wedge b_m$ and *MAJ II* simply provides the corresponding complementary values $\overline{q_m} = \neg q_m$.
- In case $m = 1$ (Fig. 5.1, right) *MAJ II* calculates the AND function $\overline{q_m} = \overline{a_m} \wedge \overline{b_m}$ and *MAJ I* provides the complementary values $q_m = \neg \overline{q_m}$.

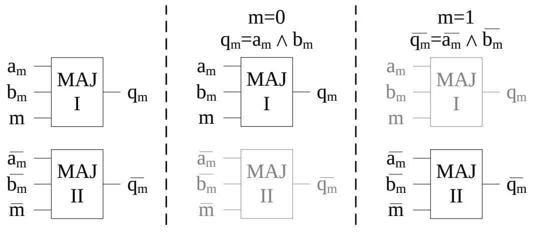


Figure 5.1: Schematic of an MDPL-AND cell (left): two complementary MAJ gates processing masked values $a_m, b_m, \overline{a_m}, \overline{b_m}$ and the masks $m, \overline{m}$ implement the AND function. In case $m = 0$ (center) *MAJ I* calculates the AND function, in case $m = 1$ (right) *MAJ II* calculates the AND function.

One major advantage of MDPL is that MAJ gates are usually available in common standard cell libraries, so there is no need to implement MDPL cells from scratch. More complex combinational cells are based on basic MDPL AND cells as shown in [PM05].

5.2 Evaluation of an MDPL Prototype Chip

The MDPL style was fabricated in real silicon in the course of the project "Side Channel Analysis Resistant Design" (SCARD) [SCA05, AMM$^+$06] in 2005. One of the chip's purposes was the evaluation of different PA-resistant logic styles, among several others MDPL was implemented on the chip. The SCARD chip contains eight functionally identical cores that were implemented in different logic styles. Each core contains an 8051-compatible microcontroller and an AES coprocessor. Most investigations of the MDPL style implemented on the SCARD chip were performed during the work for the master thesis [Kir07]. Some of our results about the security of MDPL circuits were published in [PKZM07]. In the following, the results of our investigations are briefly summarized, which represent the basis of the further work we performed: the design and the implementation of the improved MDPL style.

Our target was to evaluate the PA resistance of the MDPL style. We first focused on the microcontroller cores implemented in CMOS logic and in MDPL on the SCARD chip. In order to prove the basic existence of a leakage in a hardware implementation, the execution of very simple operations helps to estimate the power consumption of the evaluated circuit more accurately: e.g. keeping the switching noise as low as possible and being able to make clear decisions whether a power model based on the Hamming weight or on the Hamming distance matches better. Neither the complexity of the attacked operation nor the number or the size of hardware modules working in parallel to the actually investigated circuit have any influence on the basic existence of a leakage in the evaluated logic style. More precisely, the leakage might be buried in huge amounts of noise, which does not change the fact that there indeed is a leakage. A *perfectly secure* logic style would not reveal any information about the processed data even if only one single secure logic cell would be analyzed in a completely noise-free environment.

In the first attack scenario on the SCARD chip, a very basic *move byte* (MOV) operation was executed where a known byte value is moved from a source register in the internal memory of the microcontroller to a target register in the internal memory. This way we discovered the basic existence of a leakage in both attacked microcontroller cores, CMOS and MDPL. Figure 5.2 shows the correlation results of the PA attacks on the MOV operation performed in the CMOS (Fig. 5.2, left) and the MDPL (Fig. 5.2, right) microcontroller cores. The correlation trace of the CMOS core shows two distinct correlation peaks, the first peak corresponds to fetching the byte value from the source register. On fetching, the byte value first propagates through an output data multiplexer in the internal memory, followed by the main memory multiplexer in the memory controller, back through an input data multiplexer in the internal memory to the target register. One clock cycle later, the byte value is removed from the internal buses and is stored in the target register, which causes the second correlation peak. The correlation trace of the MDPL core also shows a distinct leakage, but the shape of the trace significantly differs from the CMOS one. Similar to the CMOS core, the first correlation peak in the MDPL core occurs around 1 µs, shortly after the falling clock edge (the beginning of the evaluation phase). At this time, the byte value originates from the source register, propagates through the same modules mentioned above, and arrives at the input of the target register. In the following precharge phase, the internal buses are cleared and the byte value is stored in the register.

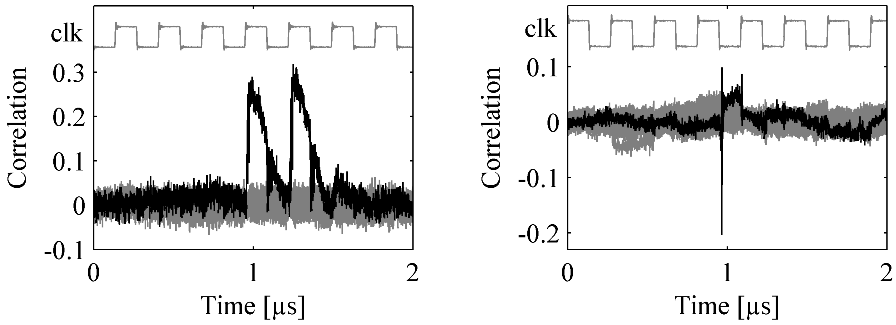


Figure 5.2: Results of the PA attacks on the SCARD chip microcontroller cores; left: CMOS core, 5 000 traces; right: MDPL core, 5 000 traces; MOV operation in the internal memory; the correlation trace is plotted in black, the clock signal is plotted in gray.

The fact that the correlation peak in the precharge phase is significantly smaller compared to the evaluation phase is referable to the internal structure of an MDPL circuit. The mask values are distributed like the clock tree via signal distribution networks having a relatively low logic depth, i.e. in most cases the mask is the first signal that changes its state at the combinational MDPL cells in the precharge and in the evaluation phase. Hence, a major part of an MDPL circuit is precharged as soon as the mask leaves its differential state: from Table 5.1 it can be derived that two out of four MAJ gates are precharged directly on arrival of a precharged mask value. This results in a precharging of the whole MDPL circuit in a very short period of time, which causes huge amounts of switching noise. As discussed in Section 2.2.1, switching noise may disguise a data leakage to a certain degree.

Evaluations on dual-rail precharge logic styles performed by Suzuki and Saeki [SS06] showed that MDPL suffers from an effect called early propagation. The authors investigated the time of evaluation of MDPL cells in different conditions of input signal delays. Delay differences between two or more input signals at a logic gate are not unusual, they emerge from signals originating from different logic depths in a combinational circuit. A very simple example is shown in Figure 5.3 based on the logic function $q = (a \vee b) \wedge c$: assuming all three signals a, b, and c originate from logic depth 0, c is directly available at the AND gate in depth 2 (disregarding wire delays). The signals a and b first have to propagate through the OR gate in depth 1, and hence, the intermediate result $a \vee b$ arrives later at the AND gate.

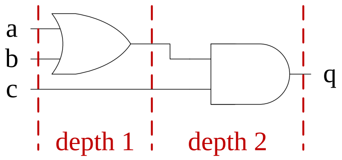


Figure 5.3: Illustration of signal delay differences in conventional logic circuits based on a simple logic function $q = (a \vee b) \wedge c$; c arrives earlier at the AND gate than $a \vee b$.

By means of measurements of an MDPL circuit implemented on an FPGA the authors of [SS06] found out that MDPL cells show a data dependent switching behavior in both the evaluation and the precharge phase. In order to unambiguously verify these findings we performed transistor-level simulations of the MDPL core. Our simulation environment was based on Synopsys Nanosim [Syn] and we performed simulations of the MOV operation for three different byte values ($0x00$, $0x0F$, and $0xFF$) and for different mask values.

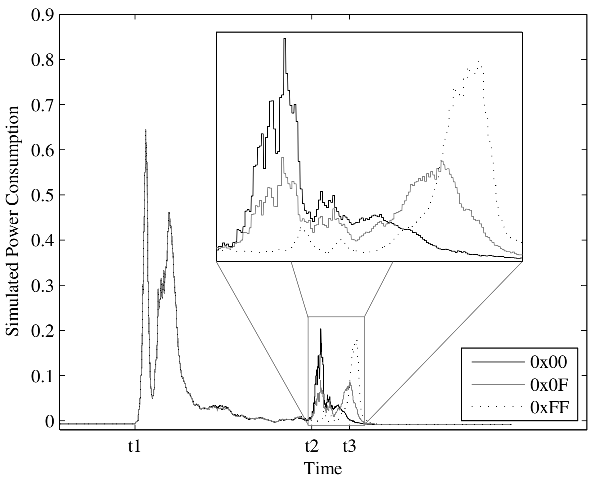


Figure 5.4: Simulated power consumption of the MDPL core during a MOV operation with mask 0; the traces show the beginning of the evaluation phase where different byte values are fetched from the source register, $0x00$ (black solid), $0x0F$ (gray solid), $0xFF$ (black dashed).

Figure 5.4 shows the power consumption of the MDPL core in the evaluation phase for the three byte values and a fixed mask value $m = 0$. The power consumption at the beginning of the evaluation phase t_1 is independent of the moved byte value. However, at the simulation time markers t_2 and t_3 we can see that the power consumption is distinctly related to the value of the moved byte. More precisely, each bit being 0 in the byte value causes a power consumption at time t_2 whereas each bit being 1 causes a power consumption at time t_3. The time difference between t_2 and t_3 is in the range of 0.8 - 1 ns. The MDPL circuit showed exactly the same behavior in case of $m = 1$. It turned out that early propagation has an unmasking effect in an MDPL circuit, for example in case of an MDPL AND cell: although the internal gates process masked data values only, the time of evaluation of one of the underlying MAJ gates solely depends on the unmasked data value. This is because a changing mask bit only switches the evaluating MAJ gate, the delay conditions of the input signals stay approximately the same (disregarding slight differences between complementary signals due to wire characteristics).

5.3 Design of the Improved MDPL Style

The evaluation results of MDPL showed that the early propagation effect significantly reduces the resistance against PA attacks. A data dependent switching behavior of logic gates in a PA-resistant logic style is a major exclusion criterion. We went for a revision of the MDPL style and tried to improve the existing MDPL cells in a way that the time of evaluation of the internal logic gates does not depend on the unmasked input values. Such an improvement requires each logic gate in the circuit to have information about the beginning of evaluation and precharge phases. There exist different methods for recognizing and introducing the different phases: for example, DDPL and SABL derive the phases directly from the clock signal (cf. [BGL+11] and [TAV02]), RSL and TDPL use additional signals with special timing constraints to detect the phases within the circuit (cf. [SSI04] and [BGLT06]). In MDPL, the detection of each phase is fairly easy due to the differential encoding of the signals: $ev = a_m \vee \overline{a_m}$. The signal ev indicating an active evaluation phase is 1 if one of the differential signals $a_m, \overline{a_m}$ is 1 and ev remains 0 (precharge phase) if both signals are 0. On the basis of this observation we enhanced the MDPL style in a way that early propagation is prevented in both the evaluation and the precharge phase. The basic idea to achieve this goal is the following:

- The beginning of the **evaluation phase** of an MDPL cell is delayed until all input signals are valid, i.e. all input signals are in differential state.
- The **precharge phase** for the output signals of an MDPL cell is initiated if the first input signal leaves its differential state, i.e. if both wires of a signal are 0.

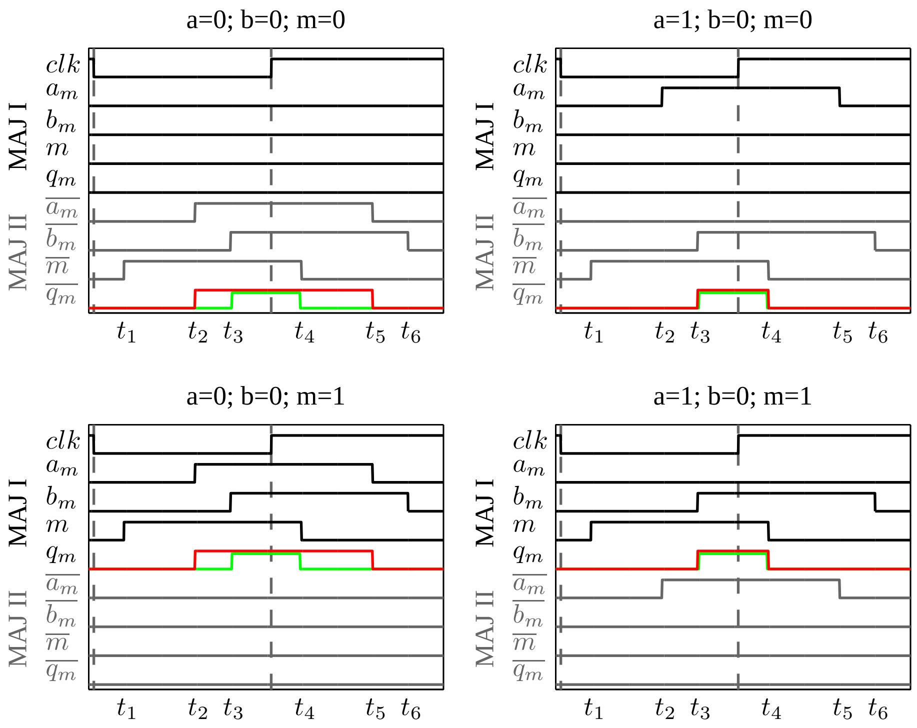


Figure 5.5: Theoretical investigation of an MDPL AND cell based on the times of evaluation of two majority gates *MAJ I* and *MAJ II* in four different signal states. The *clk* signal on top of each indicates the beginning of the evaluation and the precharge phase, the data dependent output transitions of the MAJ gates are plotted in red, the output transitions of the improved MAJ gates are plotted in green.

A theoretical investigation based on the evaluation times of two complementary MAJ gates in a basic MDPL AND cell reveals the potential improvement of this approach. Figure 5.5 shows the input and output signals of two complementary majority gates ($MAJ\ I$, $MAJ\ II$) in four different signal conditions: all possible cases for the unmasked signal a and mask m, whereas b is constantly 0. In each plot the *clk* signal on top indicates the beginning of the evaluation phase (falling clock edge) and the precharge phase (rising clock edge). The significant output transitions of $q_m/\overline{q_m}$ are plotted in red. The timing conditions are as follows: $m/\overline{m}$ arrive first at the MAJ gates at time t_1. This is the usual case in an MDPL circuit because the mask values originating from a *mask unit* propagate through a signal distribution network with a relatively low logic depth similar to the clock tree. In this example signal a arrives earlier at the MAJ gates than signal b, i.e. a originates from a shorter combinational circuit than b. Figure 5.5 top left shows the case $a = 0, m = 0$: the output $MAJ\ II/\overline{q_m}$ switches to 1 shortly after $\overline{a_m} \rightarrow 1$ at time t_2; we can observe a similar behavior in case $a = 0, m = 1$

(bottom left) where $MAJ\ I/q_m$ switches to 1 at time t_2. In both cases where $a = 1$, $MAJ\ II/\overline{q_m}$ and $MAJ\ I/q_m$ switch to 1 at time t_3 regardless of the mask value ($m = 0$, top right; $m = 1$, bottom right). The MAJ gates also show a data dependency in the switching time in the precharge phase: depending on the value of a, in each case one MAJ gate is precharged at time t_5 ($a = 0$, left plots) or t_4 ($a = 1$, right plots). The mask value only switches the evaluating MAJ gate, it has no influence on the timing behavior of the gates.

Based on these findings we extended the MDPL cells by an evaluation-precharge detection unit (EPDU) which consists of a phase-detection unit and an SR-latch stage. The schematic of an iMDPL AND cell is depicted in Figure 5.6. The output pr of the NAND gate in the phase-detection unit (left) is determined ($pr = 1$) if one of the input signal pairs $a_m/\overline{a_m}$, $b_m/\overline{b_m}$, $m/\overline{m}$ is not in differential state, i.e. if both signals are 0 (precharged). If the last input signal reaches its differential state, the input signal is simultaneously available (neglecting minor delays due to different wire lengths) at the phase-detection unit and at the SR-latch stage. The SR-latch stage, consisting of three 3-input NOR gate pairs, does not evaluate until the last differential input signal propagated through the phase-detection unit and $pr = 0$.

The green signal traces in Figure 5.5 (left plots, $\overline{q_m}$ and q_m) indicate the behavior of improved MDPL cells: they evaluate only if all signals have reached their differential state (at time t_3), and they start precharging if the first signal leaves its differential state (which is usually the mask value at time t_4). In case $a = 1$ the switching times of $MAJ\ II/\overline{q_m}$ and $MAJ\ I/q_m$ already match the *late evaluation/early precharge* conditions.

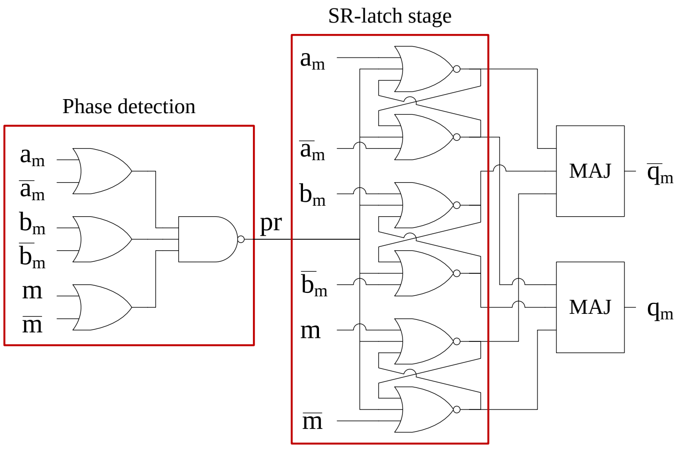


Figure 5.6: Schematic of an iMDPL-AND cell; the phase detection unit provides an evaluation/precharge signal pr for the following SR-latch stage; the SR-latch stage does not evaluate (*pr=0*) until all input signals are in differential state and precharges the MAJ gates if the first input signal leaves its differential state (*pr=1*).

In iMDPL the relation between the processed data a/b and the time of evaluation is broken. Our evaluations (theoretically and simulation based) showed that this holds true for any timing condition, e.g. if b arrives earlier than a, as well as for any possible state combinations of a, b, and m. Each NOR latch produces 0 at both outputs as long as the output of the phase-detection unit $pr = 1$. The NOR latches start evaluating simultaneously on arrival of $pr = 0$. At this time the input signals at each NOR latch are already in differential state, i.e. the NOR latches simultaneously provide differential data to the subsequent MAJ gates.

Similar to MDPL, more complex combinational iMDPL cells can be built from such basic iMDPL AND cells. If the outputs q_m and $\overline{q_m}$ of an AND cell are simply swapped, we obtain a NAND cell. An XOR function $q = a \oplus b$ can also be written as $q = (a \wedge \overline{b}) \vee (\overline{a} \wedge b)$, applying De Morgan's law we obtain: $q = \overline{\overline{a \wedge \overline{b}} \wedge \overline{\overline{a} \wedge b}}$. Negation of a signal is achieved by swapping the two complementary lines, so we can build an iMDPL XOR gate based on three iMDPL NAND gates. Similar to combinational MDPL cells, also flip-flops in an MDPL circuit need to be improved to prevent early propagation. An iMDPL data-flip-flop (D-FF) depicted in Figure 5.7 has to perform four tasks (from left to right in the figure):

1. Prevent early propagation in the evaluation and the precharge phase: this task is performed by an EPDU, consisting of a phase-detection unit and an SR-latch stage as depicted in Figure 5.6 in case of an iMDPL-AND cell.

2. In the next step, *switch mask*, the current mask m is removed and the mask of the next clock cycle m_n is applied before the data value is stored in the D-FF. As iMDPL implements Boolean masking, the mask switch can be performed with two XOR operations: $q_{mn} = (q_m \oplus m) \oplus m_n$. There is also an easier way: an iMDPL circuit can be supplied with the mask $m/\overline{m}$ and the mask XORed with the next mask $m \oplus m_n$. In this case we can save one XOR operation in each D-FF. Furthermore, the used iMDPL XOR cell can be slightly modified as only four instead of six inputs are required[1].

3. The third task is to store the data value masked with the next mask m_n in a CMOS D-FF. In the next clock cycle, the data value is already masked with the actual mask value and can be processed correctly in combinational logic.

4. The fourth task of an iMDPL D-FF is to keep both complementary signals at 0 during the precharge phase. This function is realized by two NOR gates connected to the clk signal: during the precharge phase ($clk = 1$) both NOR gates produce 0 at their outputs, regardless of the D-FF outputs.

[1]The remaining two input signals of a standard iMDPL XOR cell can be kept constant or the internal structure of the XOR cell can be simplified and built with a CMOS AND, an OR, and a MAJ gate. These gates can be precharged correctly and they do not produce glitches, which makes them suitable for this task.

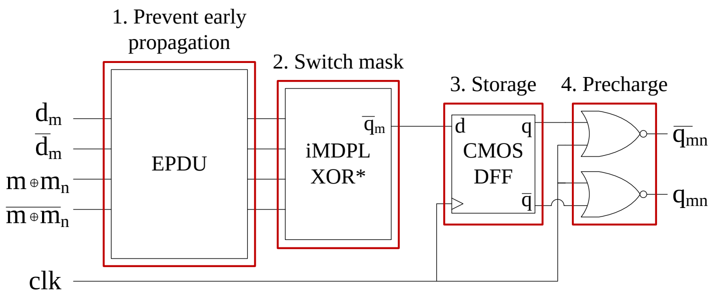


Figure 5.7: Schematic of an iMDPL-DFF cell; the EPDU prevents early propagation, the XOR replaces the current mask with the next mask, the CMOS D-FF stores the data value, the NOR gates realize precharged outputs during the precharge phase.

5.4 Implementation of the GRANDESCA Chip

During the FIT-IT project "Generating RANDom values for Encryption in the presence of Side Channel and other Attacks" (GRANDESCA) [GRA07] we implemented different hardware modules in iMDPL on a prototype chip. Similar to the SCARD chip, the purpose of the GRANDESCA chip was the implementation and the evaluation of masked logic styles in practice to improve the resistance against PA attacks. In the following sections we present the architecture of the GRANDESCA chip, elaborate on the implemented hardware modules on the chip, and finally present results of the fabricated chip.

5.4.1 Architecture of the GRANDESCA Chip

The GRANDESCA chip contains two types of cores: (1) an 8051-compatible microcontroller with an AES coprocessor and (2) a standalone AES core. Each of the cores was implemented in two different logic styles: unprotected CMOS logic and iMDPL. The architecture of the GRANDESCA chip is depicted in Figure 5.8: the chip contains two *MCcores*, two *AEScores*, a *control logic*, and three modules which are responsible for the mask management (*PRNG*, *mask reader*, and *mask observer*). The control logic acts as a multiplexer: depending on the selected active core (*core selection* signal), the control logic provides the clock signal, memory and user I/O signals only to one of the four cores. In the following we describe the main modules of the GRANDESCA chip in more detail, justify our decisions regarding the implemented cores, and point out special features of the chip in order to being able to thoroughly investigate the iMDPL style.

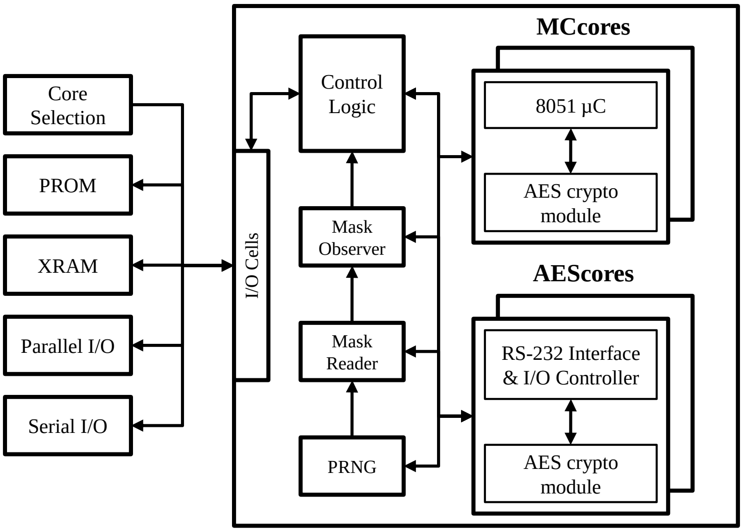


Figure 5.8: Architecture of the GRANDESCA chip.

8051-Compatible Microcontroller Cores: *MCcores*

Each MCcore contains an implementation of an 8051-compatible microcontroller and an AES crypto module that is used as a coprocessor. The 8051-compatible microcontroller has the following features: a serial interface (RS232), 128 bytes of internal memory (IRAM), and an 8-bit parallel I/O port. The executed program is stored on an external program ROM (PROM) and the microcontroller also provides access to an external memory module (XRAM). The microcontroller core represents a very good target for evaluating the PA resistance of a logic style like iMDPL, since absolutely no optimizations or special hardware structures are implemented in the microcontroller module. Various operations and programs can be utilized to evaluate the effectiveness of iMDPL in the 8-bit microcontroller. The architecture of the AES coprocessor in the MCcores is identical to the architecture of the standalone AES cores which is introduced in the following section.

Standalone AES Cores: *AEScores*

The AES modules support AES-128 encryption/decryption in ECB mode, and the implementation follows the standard version of the AES hardware architecture presented by Mangard et al. in [MAD03]. The architecture implements the logical 4×4 state layout of AES in hardware and contains one single MixColumns module attached to the leftmost column of the 4×4 AES state. A barrel shifter implements the ShiftRows operation and the four S-boxes attached to the topmost row of the 4×4 AES state are combinational and one-stage pipelined implementations as described by Wolkerstorfer et al. in [WOL02]. Initially, the

AES state values are loaded column-wise from the right within 4 clock cycles and each consecutive AES round requires 9 clock cycles. In a first step, each of the 16 AES state cells applies the AddRoundKey operation. In the following 5 clock cycles, the AES state rows are shifted down through the S-boxes and the ShiftRows operation. Afterwards, the AES state columns are shifted to the left through the MixColumns module which requires another 4 clock cycles. An example of the data processing in the AES module is depicted in Figure 5.9 based on 4 clock cycles. In parallel to the MixColumns operation, the next AES round key is calculated. Within the last MixColumns cycle also the next AddRoundKey operation is applied. When all AES rounds are finished, the final AES state values are read out column-wise to the left. The AES modules are equipped with an AMBA APB interface. We used a serial to AMBA APB conversion module as a bridge between the simple UART interface of the GRANDESCA chip and the AMBA APB bus.

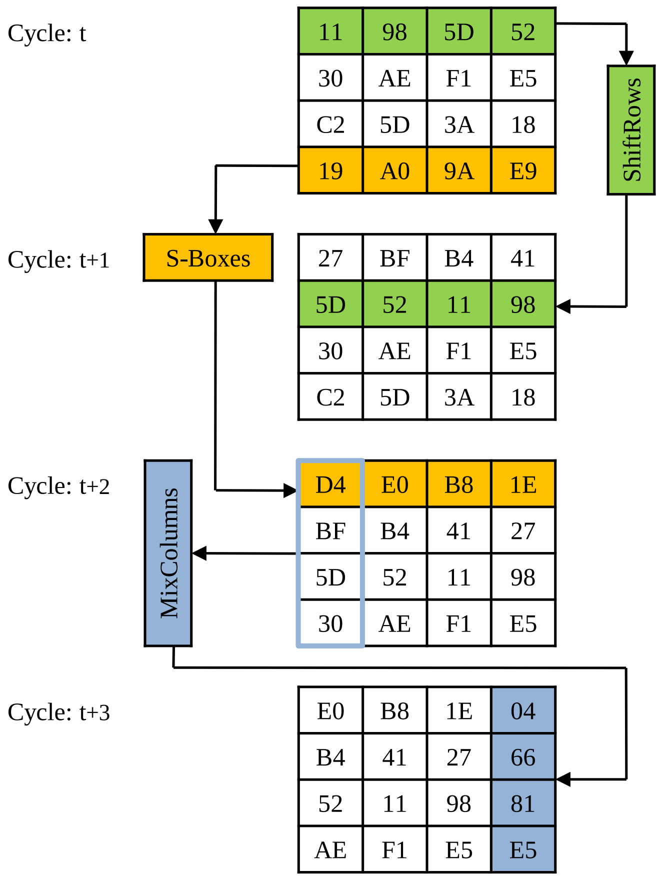


Figure 5.9: Example of the data flow in the AES hardware module by means of four consecutive clock cycles.

The reason why we implemented identical AES modules as coprocessors in the MCcores and as standalone cores was the following: our findings on the SCARD chip suggested that the 8051-compatible microcontroller produces a huge amount of noise. The noise significantly counteracts our efforts to investigate individual modules in the cores (i.e. the AES coprocessor in the MCcores) and also falsifies our evaluation results. By implementing the standalone AES modules we increased our chances to obtain more meaningful results from our investigations on the GRANDESCA chip.

Mask Management on the GRANDESCA Chip

The focus of our work during the GRANDSCA project was the implementation and evaluation of the iMDPL style, we did not intend to develop a cryptographically secure random number generator (CSPRNG). Hence, a pseudo-random number generator (PRNG) based on a non-linear feedback shift register (NLFSR) that provides one unbiased random bit per clock cycle for the iMDPL cores has met our demands. In order to perform advanced experiments regarding the mask in an iMDPL circuit, we implemented some special features and commands for the PRNG which are discussed based on the state machine of the PRNG depicted in Figure 5.10. After startup in the *reset* state the PRNG is brought to the *stop* state. Without any further initialization after startup, the PRNG provides a valid and constant mask 0. Starting from *stop* state, we can load an initial seed into the PRNG *load_seed*. In *run* state the PRNG provides one fresh mask bit per clock cycle. The output of the PRNG can be overridden with 0 (*const_0*) or 1 (*const_1*), which also stays active during state transitions between *stop* and *run*. The override can also be disabled again (*no_const*) and does not influence the state of the PRNG in the background, i.e. in *run* state the PRNG keeps running and serves as a noise generator. Each iMDPL core

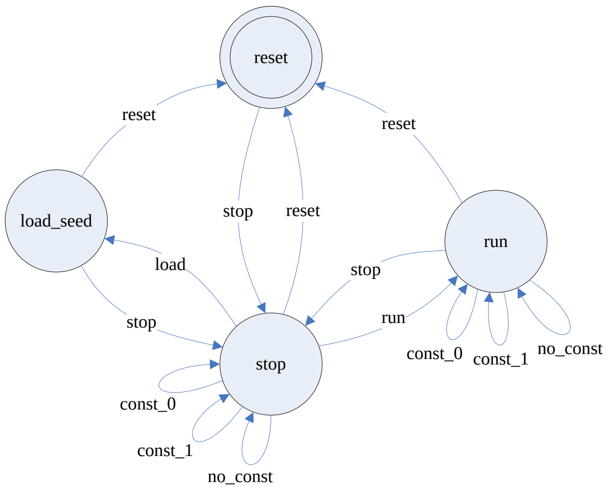


Figure 5.10: PRNG state machine.

contains a so-called *mask unit*, which obtains a new mask bit from the PRNG in each clock cycle. The mask unit is integrated in the top-level of each core that is implemented in the iMDPL style and provides the different types of masks required in the iMDPL circuit: m, $\overline{m}$, $m \oplus m_n$, $\overline{m \oplus m_n}$. The advantage of using a mask unit is the following: Since the mask unit is integrated in the respective core, the mask unit is part of the clock tree. Hence, the timing constraints of the masks with respect to the clock signal are satisfied, i.e. the mask does not change before an iMDPL flip-flop has stored the data value at its input.

The *mask reader* module implemented on the GRANDESCA chip offers the possibility to supply the iMDPL cores with an external mask instead of using the mask provided by the PRNG. This feature enables the usage of arbitrary mask values, e.g. to introduce a bias in the mask values, which helps to investigate the side channel resistance of the masking technique. The *mask observer* module enables the observability of the current mask bit provided by the PRNG. As this special feature must not cause any additional leakage during normal operation of the PRNG, the observation of the mask bit can explicitly be enabled and disabled. The mask reader and the mask observer were implemented as a multiplexer next to the PRNG (outside of the iMDPL cores) and are attached to the control logic, which allows us to control both modules from each of the four cores implemented on the GRANDESCA chip. This way we prevented the mask units and the mask signal trees within the iMDPL cores from being affected by these two special modules. Implementing additional circuitry directly in the signal tree of the mask m would most probably have caused significant differences between m and $\overline{m}$ in the iMDPL circuits, which would have forged our evaluations of iMDPL on the GRANDESCA chip. By default, the mask reader and the mask observer are deactivated, i.e. the mask provided by the PRNG is used and the mask is not redirected to an output pin of the chip. We took a further measure to decouple the influences of activated mask reader or mask observer on the iMDPL circuits: we inserted an additional flip-flop between the output of the PRNG and the iMDPL circuits. This way, an externally provided or observed mask bit arrives only one clock cycle delayed at the mask unit of the active iMDPL core, which reduces the influence on the power consumption in the actual clock cycle.

However, we have to note that the implementation and the usage of such special hardware structures in the context of masking techniques needs to be carried out with great care in order to prevent the accidental cause of leakages. Needless to say that such structures are only tolerable in prototype devices, they are absolutely inappropriate in security related final products.

5.4.2 Implementation of the iMDPL Style

The implementation of the iMDPL style on our prototype chip was performed according to the design-flow steps described in Section 3.3. In a first step the logic synthesis of each iMDPL core on the chip was performed using a subset of standard logic cells. The second and main step after finishing the synthesis was the conversion of the CMOS logic to iMDPL using a netlist converter tool.

After an additional custom design step that created balanced signal trees for the different masks in the iMDPL circuit, the gate-level netlist including the iMDPL cores was ready for a conventional place and route. In the end, we performed simulations to verify the correct functionality of the design before finishing the layout of the chip (adding the I/O pads and the die-seal ring). After that, the design was ready for tape-out.

5.4.3 Production of the GRANDESCA Chip

The GRANDESCA chip was produced in a 180nm CMOS process technology from UMC [Uni] using a standard cell library from Faraday [Far04]. The overall die size of the GRANDESCA chip including pads is $\approx 17\,mm^2$. A summary of the core circuit complexities and the maximum clock frequencies is shown in Table 5.2. The table clearly shows the costs of the logic style iMDPL: the area of an iMDPL implementation requires approximately 18-20 times the area of a corresponding CMOS implementation and the maximum clock frequency is approximately one fifth. As a comparison the table also contains estimations for the MCcore and AEScore modules implemented in MDPL. Compared to CMOS an MDPL implementation requires approximately 5 times the area, thus, the improved version of the MDPL cells requires approximately 4 times the area of original MDPL cells.

Table 5.2: Circuit complexities of the cores on the GRANDESCA chip and their maximum clock frequencies. The complexities of the MDPL cores (marked with an asterisk) are estimations.

	Core area [GEs]	Frequency [MHz]
MCcore CMOS	30 800	14
MCcore iMDPL	575 000	2.7
AEScore CMOS	10 800	>15
AEScore iMDPL	197 000	7.4
MCcore MDPL*	154 000	7
AEScore MDPL*	54 000	>10

The floorplan of the GRANDESCA chip is shown in Figure 5.11. The two unprotected implementations in plain CMOS are placed top right (*AEScore CMOS* and *MCcore CMOS*), the stand-alone AES processor implemented in iMDPL is placed top left (*AEScore iMDPL*), and the 8051-microcontroller including the AES coprocessor implemented in iMDPL takes approximately two thirds of the whole area from the bottom (*MCcore iMDPL*). The control logic, PRNG, mask reader, and mask observer are placed in between the four cores.

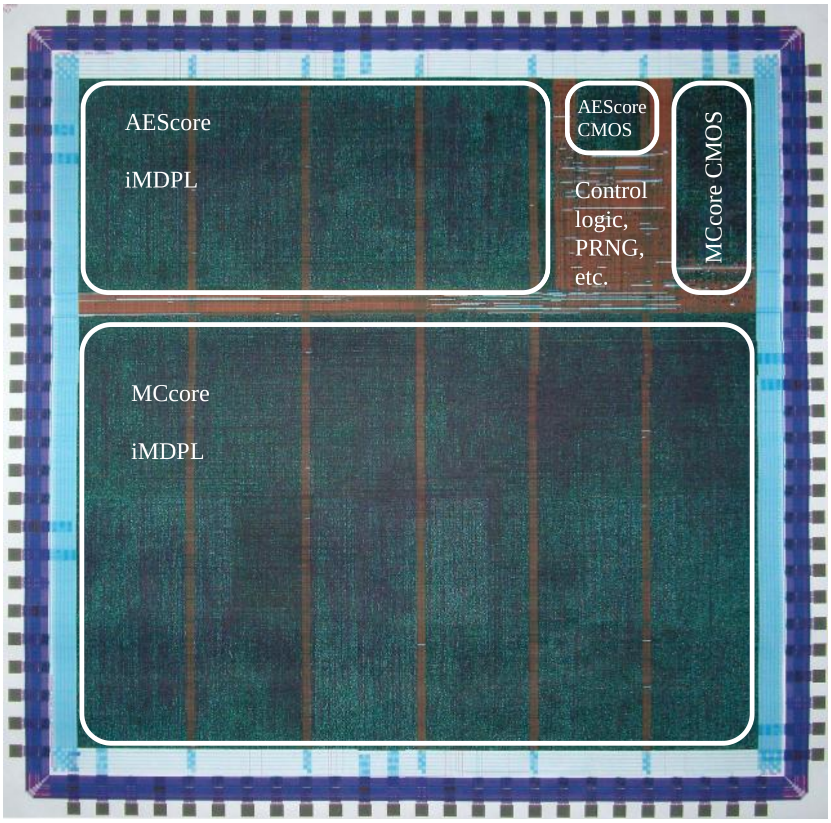


Figure 5.11: Floorplan of the GRANDESCA chip.

5.5 Evaluation of the iMDPL Style Implemented on the GRANDESCA Chip

In the following, we first introduce the measurement setup we used for performing the power measurements of the GRANDESCA chip. We then present the results of the PA attacks performed on different cores and operations: (1) on an internal MOV operation in the MCcore CMOS, (2) on the standalone AES module in the AEScore CMOS, (3) on an internal MOV operation in the MCcore iMDPL, and (4) on the standalone AES module in the AEScore iMDPL. Subsequently we provide a summary of the performed attacks and also discuss the results of PA attacks on the AES modules working as coprocessors in the MCcores implemented in CMOS and in iMDPL.

For evaluating the PA resistance of the iMDPL style on the GRANDESCA chip we built up a standard measurement setup (cf. Section 4.4.1): (1) an evaluation board holding the GRANDESCA chip and external devices like a PROM and an XRAM module and providing the required power supply and I/O functionality, (2) a digital oscilloscope and a differential probe for measuring the instantaneous power consumption of the chip during operation, and (3) a PC for controlling the oscilloscope and communicating with the GRANDESCA chip on the board. Power measurements are usually performed using a 10 to 50 Ω resistor in the V_{DD} or GND line of the device under test [BGLR08, PKZM07]. We tried a different approach and used a BAT41 small-signal Schottky diode operated in forward direction in the V_{DD} line of the GRANDESCA chip. The idea behind this approach is based on the non-linear relation between the current flow and the voltage drop: we assume that the power spikes occurring mainly at clock events are reduced to a certain degree, which gives the opportunity to increase the vertical resolution of the measured oscilloscope channel without clipping any possibly important parts of the measured signal. Several measurements and PA attacks on different devices using a BAT41 Schottky diode consistently resulted in an improvement of the PA attacks, i.e. the number of required power traces to perform a successful attack was reduced in every case. One single drawback has to be noted about this measurement approach: the correlation traces obtained from measurements using the Schottky diode are slightly blurred over time. This effect can be seen when looking at Figure 5.12 (left) where the correlation trace is blurred and Figure 5.2 (left) where the correlation trace shows a cleaner progression. Hence, in applications where a clear separation of the correlation is required the common 10 to 50 Ω resistor is still the first choice. Table 5.3 summarizes the most important parameters of our measurements performed on the GRANDESCA chip.

Table 5.3: Summary of the parameters for the measurements of the GRANDESCA chip.

Oscilloscope	LeCroy LC584
Probe	LeCroy AP034
Probe Coupling	1 $M\Omega$ AC
Clock frequency	1.8432 MHz oscillator for MCcores 3.6864 MHz oscillator for AEScores
Measurement shunt	BAT41 Schottky diode in the V_{DD} line
Sampling rate	2 GS/s

5.5.1 PA Attack Results

The practical results of the PA attacks on the GRANDESCA chip were published in [KP09]. In the following sections, we present and discuss the results of the attacks on the different cores implemented on the GRANDESCA chip. All our experiments were performed by means of conventional correlation-based PA attacks.

PA Attacks on the MCcore CMOS

In our first attack we focused on the 8051-compatible microcontroller in the MCcore CMOS. Similar to the attack on the SCARD chip described in Section 5.2, we executed a simple *move byte* (MOV) operation in the internal memory and performed a correlation-based PA attack on the known byte value. The attack was performed using the Hamming weight (HW) of the moved byte value to generate the power hypothesis. Figure 5.12 shows the result of the attack on the MOV operation in the MCcore CMOS. The left plot shows 40 correlation traces, 39 of them represent the correlation results with randomly chosen byte values. Around the time markers 2-2.5 µs we can identify a significant leakage: the correlation values reaches 0.334, which corresponds to a number of required power traces of 240. Figure 5.12 (right) shows the evolution of the maxima of the correlation values over the number of traces.

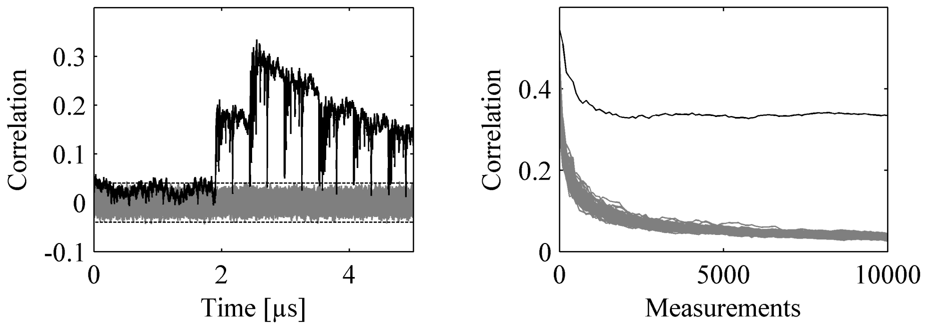


Figure 5.12: Result of the PA attacks on the GRANDESCA MCcore CMOS: internal MOV operation in the IRAM, 10 000 traces, HW power model; left: 40 correlation traces, 39 traces represent randomly chosen byte values; right: evolution of the maxima of the correlation traces over the number of traces; the trace for the correct hypothesis is plotted in black.

PA Attacks on the AEScore CMOS

In our first attack on the internal MOV operation of a known byte value we verified the basic existence of a leakage in the GRANDESCA chip. In our second PA attack we targeted the AEScore CMOS and exploited the leakage to reveal the secret key used for performing the AES operations. Contrary to the MOV operation in the microcontroller where the data buses returned to zero after transferring the byte value, the data buses in the AEScore show a different behavior. The intermediate values processed in the AES module change almost every clock cycle, i.e. the instantaneous power consumption depends on two consecutive byte values propagating through the internal buses. Hence, we based our attacks on the Hamming distance (HD) power model which corresponds to the number of different bits between two bit sequences. A further important criterion of performing PA attacks is to choose an appropriate *selection function* for the attacks. The selection function represents a part of the intermediate value computed by the attacked device and should be related only to a small part to the secret key, preferably 1 or 2 bytes. This allows the examination of all possible key hypotheses for the selected part of the secret key within a reasonable time. In case of the attacks on the internal MOV operation there was no secret key involved, the selection function was the moved byte value.

In case of a brute force attack, the whole ciphertext of an AES-128 encryption represents the selection function which is related to 128 key bits. The examination of all possible key hypotheses thus includes 2^{128} AES keys which corresponds to approximately 3×10^{38} possibilities that would have to be checked, which renders such an attack infeasible. An important feature determining the effectiveness of a PA attack is the non-linearity degree of the relation between the part of the secret key and the selection function: in case of a non-linear relation a one-bit difference at the input of the selection function leads to a multi-bit difference at the output, i.e. only slightly differing key hypotheses produce significantly different outputs in the selection function. Hence, incorrect key hypotheses yield only considerably smaller correlation values than correct ones in case of non-linear selection functions. In case of AES, the output of the S-box transformation is a common choice for the selection function: the Rijndael S-box was specifically designed to feature a highly non-linear relation between its input and output values. Furthermore, the S-box transformation within an AES operation only depends on 8 bit of the secret key, which results in a feasible examination of 2^8 key hypotheses. In case of our attacks on the AEScore CMOS, the hypotheses were based on the HD of two consecutively processed output values of one S-box, which resulted in a manageable key space of 2^{16}.

In order to simplify our evaluation, we performed a logic simulation of a high-level model of the AES module and worked out the appropriate key hypothesis for the PA attack. By means of the simulations we discovered the consecutively processed intermediate byte values and derived the appropriate byte transitions for generating our key hypothesis. We assumed that one of the two involved key bytes is already known, which allowed us to reduce the effort during the examination of the key hypotheses from 2^{16} to 2^8.

We defined a PA attack on the AEScore to be successful it at least 9 of the 16 key bytes could be distinctly revealed by the attack. We assumed that the remaining 7 key bytes can also be revealed by a brute force attack within reasonable time. Our approach is based on the findings of brute-force attacks on the 7-byte key of DES of Kumar et al. [KPP+06a, KPP+06b]. On the basis of the ninth-highest correlation value we used the number of required power traces to perform a successful PA attack as comparative value. We used the *rule of thumb* [MOP07] (Chapter 6.4.1) to calculate the number of required traces. Figure 5.13 (left) shows the correlation results of the attack on key byte 2 in the AEScore CMOS. Around the time marker 2 µs we can see a distinct correlation peak of 0.0334, which corresponds to a number of required power traces of approximately 25 000.

PA Attacks on the MCcore iMDPL

After obtaining reference values by attacking the CMOS cores, we performed the same attacks on the MCcore and AEScore implemented in iMDPL. Again, we first performed a PA attack on a MOV operation within the MCcore iMDPL. We performed 40 000 measurements and analyzed the power traces using the HW power model of the moved byte value. Figure 5.14 shows the result of the attack: a correlation peak around the time marker 2 µs can clearly be seen. The maximum correlation value of 0.0407 corresponds to a number of required power traces of approximately 17 000, which means that compared to CMOS we require 70 times more power traces to perform a successful PA attack on the iMDPL core.

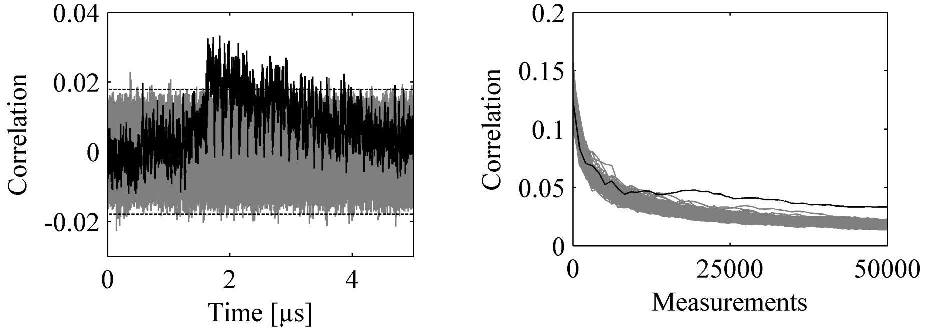


Figure 5.13: Result of the PA attacks on the GRANDESCA AEScore CMOS, 50 000 traces, HD power model; left: correlation traces for key byte 2; right: evolution of the maxima of the correlation traces over the number of traces; the trace for the correct key hypothesis is plotted in black.

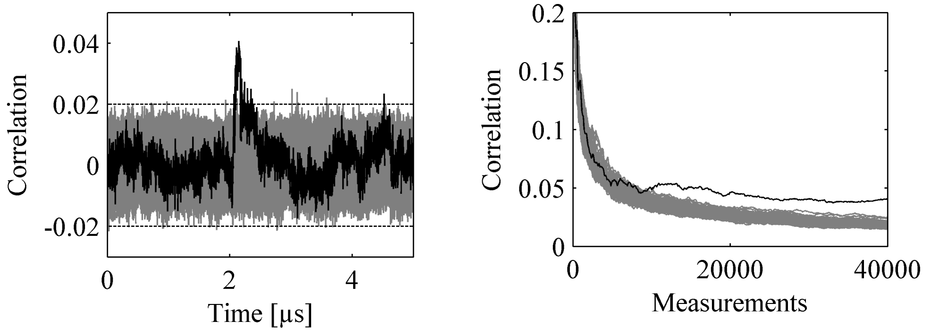


Figure 5.14: Result of the PA attacks on the GRANDESCA MCcore iMDPL: internal MOV operation in the IRAM, 40 000 traces, HW power model; left: 40 correlation traces, 39 traces represent randomly chosen byte values; right: evolution of the maxima of the correlation traces over the number of traces; the trace for the correct hypothesis is plotted in black.

PA Attacks on the AEScore iMDPL

We further performed a PA attack on the AEScore iMDPL. Contrary to CMOS, the power hypotheses are based on the HW of single bytes of the S-box output due to the precharge phase in iMDPL. Hence, the space for each key guess was by nature 2^8. Similar to AEScore CMOS, we defined a PA attack on the AEScore to be successful it at least 9 of the 16 key bytes can be distinctly revealed by the attack. Figure 5.15 (left) shows the correlation results of the attack using 5 100 000 measurements of key byte 15 in the AEScore iMDPL. Similar to the attack on the MCcore iMDPL, we can identify a distinct correlation peak. The obtained correlation value of 0.003 corresponds to 3 000 000 required power traces. In case of AEScore iMDPL we required approximately 120 times more power traces to perform a successful PA attack compared to AEScore CMOS.

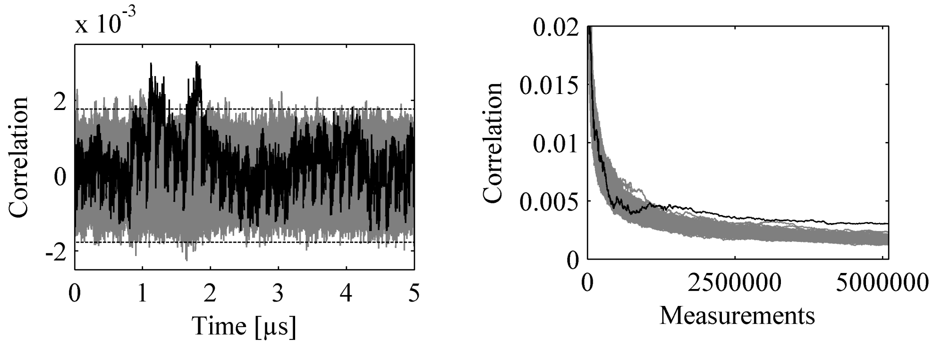


Figure 5.15: Result of the PA attacks on the GRANDESCA AEScore iMDPL, 5 100 000 traces, HW power model; left: correlation traces for key byte 15; right: evolution of the maxima of the correlation traces over the number of traces; the trace for the correct key hypothesis is plotted in black.

5.5.2 Summary of the Performed PA Attacks

Our evaluations have shown that the iMDPL style is able to significantly increase the PA resistance of a device. A comparison between MDPL and iMDPL shows that the remarkable leakage in MDPL caused by early propagation is considerably reduced in iMDPL. A comparison of the PA attack results on the CMOS cores and the iMDPL cores leads to an increase of the PA resistance by a factor of approximately 100 (70 to 120). Certainly, this factor depends on various influences such as the structure of the implemented hardware module or the amount of noise produced by adjacent modules. The PA resistance of iMDPL is increased at the expense of area and speed: an iMDPL implementation takes approximately 18 times the area of a corresponding implementation in CMOS logic and the maximum clock frequency of an iMDPL implementation is about a fifth compared to CMOS logic.

Table 5.4: Summary of the results of the performed PA attacks on the GRANDESCA chip. The evaluations are based on the correlation value ρ and the number of required power traces n to perform a successful attack; *N/A* means that the corresponding attack was not applied; *N/S* means that the corresponding attack was not successful, i.e. possibly more power traces are required to perform a successful attack.

Logic $\rightarrow$	CMOS		iMDPL	
Core $\downarrow$	HW	HD	HW	HD
MCcore MOV	$\rho \approx 0.334$ $n \approx 230$	N/S $n > 10\,000$	$\rho \approx 0.0407$ $n \approx 17\,000$	N/S $n > 40\,000$
AEScore	N/S $n > 50\,000$	$\rho \approx 0.0334$ $n \approx 25\,100$	$\rho \approx 0.00305$ $n \approx 3\,000\,000$	N/S $n > 5\,100\,000$
MCcore AES	N/S $n > 50\,000$	$\rho \approx 0.0331$ $n \approx 25\,600$	N/S $n > 6\,200\,000$	N/S $n > 6\,200\,000$

The results of the implemented PA attacks on the GRANDESCA chip are summarized in Table 5.4. Besides the previously discussed PA attacks on the internal MOV operation in both MCcores and the attacks on both AEScores, we also performed attacks on the AES coprocessor in both MCcores, attacks on the iMDPL cores with deactivated PRNG, as well as attacks using different power models.

As mentioned in Section 5.4.1, we also implemented AES modules working as coprocessors attached to the microcontroller modules in the MCcores. The results of the PA attack on the AES coprocessor in the MCcore CMOS are comparable to the results obtained from AEScore CMOS: all key bytes could be revealed by the attack, approximately 25 600 power traces are required to distinctly reveal the ninth-highest correlation value among them. In contrast, we were not able to distinctly reveal any key byte with an attack on the AES coprocessor in the MCcore iMDPL using up to 6 200 000 power measurements. We assume that the main reason for the failed attack is a high amount of noise caused by the rather huge implementation of the 8051-compatible microcontroller in iMDPL. This assumption is supported by the fact that the attack on the identically implemented AES module in the standalone AEScore iMDPL was successful using approximately 3 000 000 power traces. The only difference between the two AES modules is the active 8051-compatible microcontroller in the MCcore implemented in iMDPL which has a complexity of approximately 410 kGEs. Contrary, the microcontroller implemented in CMOS logic is much smaller (around 22 kGEs) and thus produces a significantly lower amount of noise.

As can be seen in Table 5.4, switching the power model for the attacks on the AES modules did not lead to any convincing results. In case of the attacks on the CMOS AES modules we were not able to reveal any of the key bytes by using a HW power model. This is not surprising considering the power consumption behavior of basic CMOS gates: signal changes consume significantly more power than signals keeping their state (cf. Section 2.1). The execution of a MOV operation in a microcontroller usually requires several clock cycles but only in one of the clock cycles the moved byte value propagates over the internal data buses, which causes a data leakage. In the following clock cycle the byte value is removed from the internal buses which causes a second data leakage. In case of the microcontroller implemented in the MCcore CMOS the internal buses are zero before and after the moved byte value is valid, which results in a leakage depending on the HW of the moved byte value: $HW(v) = HD(0, v)$. However, this is not the case in the AES modules. In each clock cycle the internal buses (mostly fed by registers) process different byte values as illustrated in Figure 5.9 on page 63. In most cases the internal buses do not switch back to zero between the processing of consecutive data values, i.e. the number of changing bits when switching from the actual data value A to the new value B determines the required amount of power to set the internal bus to its new state. Hence, the circuit leaks information about $HD(A, B) = HW(A \oplus B)$.

This observations do not hold true in case of iMDPL circuits, as all internal buses are precharged in every clock cycle, i.e. each bus switches back to zero between processing two consecutive data values. Hence, the HD of two consecutively processed byte values is not a suitable power model for attacks on iMDPL, and thus, it is not surprising that the attacks using a HD-based power model for attacking the AES modules implemented in iMDPL were not successful.

By means of power measurements we were able to discover the existence of a leakage in the iMDPL style. On the basis of bare measurements we were not able to retrieve any further information about the actual source of the leakage: are there flaws in the design of the evaluation-precharge detection units attached to each MDPL gate? Does early propagation still occur in iMDPL in a reduced form? Does the leakage originate from a completely different and so far undiscovered phenomenon in digital circuits? In order to answer these questions we performed a thorough investigation of the leakage in iMDPL which was published in [MKEP]. The results of our investigations are presented in the following section.

5.5.3 Exploring the Leakage of iMDPL

The PA attacks on the iMDPL cores on the GRANDESCA chip revealed that there is still a leakage in the logic style that enables the execution of successful PA attacks if a sufficiently large number of power measurements is performed. We pinpointed the source of the leakage in iMDPL by utilizing low level information obtained from simulations and the layout of the GRANDESCA chip. In this section, we provide a thorough investigation of the leakage in iMDPL by means of back-annotated logic simulations, PA attacks on toggle-count traces, and detailed layout information. Due to the complexity and the long simulation time of several encryption runs in an AES module of the GRANDESCA chip, we decided to base our investigations on a MOV instruction in the MCcore iMDPL. Each simulation run of an AES operation would require more than 100 clock cycles in the GRANDESCA design. According to the results of our PA attacks on the GRANDESCA chip we would require a significantly larger number of toggle-count traces in order to obtain reasonable attack results on an AES module compared to the MOV operation in the MCcore. In contrast, one MOV operation requires only 3 clock cycles in the microcontroller which allowed us to perform a sufficient amount of simulations to carry out reasonable investigations.

Isolation of the Leakage in the Time Domain

In a first step we isolated the leakage in the time domain. Based on attacks on measured power traces we identified the clock cycles where the leakage during the execution of a MOV operation in the MCcore iMDPL occurs. As our power measurements with the BAT41 Schottky diode were slightly blurred over time, we also performed EM measurements of the GRANDESCA chip. EM measurements are able to reduce the switching noise, which increases the SNR. Furthermore, EM measurements result in a better resolution over time, as significant inductive and capacitive influences are mostly canceled out.

Figure 5.16 (left) presents the results of the attacks on 30 000 measured power and EM traces. The correlation result obtained from the power traces (plotted in gray) shows a significant blurring over time, whereas the EM correlation (plotted in black) shows two distinct correlation peaks in the sixth clock cycle after the trigger signal *trg* goes HIGH, one peak in the evaluation and one smaller peak in the precharge phase. Figure 5.16 (right) shows the correlation result obtained from 256 toggle-count traces. The correlation trace matches the result obtained from EM measurements very well: we can also see two distinct correlation peaks in the evaluation and in the precharge phase in the sixth clock cycle.

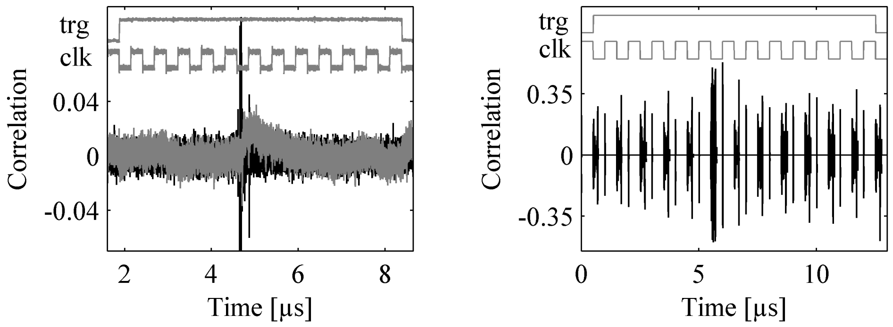


Figure 5.16: Results of PA/EMA attacks on the MCcore iMDPL, MOV operation; left: 30, 000 measured power/EM samples, correlation trace obtained from power measurement is plotted in gray, correlation trace obtained from EM measurement is plotted in black; right: 256 simulation runs, correlation trace obtained from toggle count traces is plotted in black; the clock and trigger signals are indicated on top of both figures in gray.

Isolation of the Leaking Submodules

In a second step, we started to investigate the simulation results in more detail in order to isolate the leaking submodules. Our VCD analyzing technique (cf. Section 4.2.2) allowed us to focus our evaluations on specific modules within the whole GRANDESCA design. In order to rule out the existence of a general flaw in our hardware design, we analyzed all signals within the GRANDESCA chip excluding the suspected MCcore iMDPL. The correlation results did not show any sign of a data dependency in the toggle-count traces, i.e. the leakage solely originates from one or more modules within the MCcore iMDPL. With further evaluations on different parts in the MCcore iMDPL design we were able to narrow down the number of suspicious modules to 3: (1) the memory module containing an address decoder and the registers, (2) the arithmetic logic unit (ALU) of the microcontroller, and (3) the memory controller containing the main data multiplexer (MUX). We also tracked the data flow during the execution of a MOV operation within the MCcore: the byte value is read from the source register in the RAM module and is fed to the main data MUX in the memory control unit and to a MUX in the ALU. The data is not further processed after the MUX in the ALU, but multiplexing the data in a combinational circuit already causes a distinct leakage as our evaluations showed. The data MUX in the memory control unit is significantly larger as it controls all read and write operations from and to different registers and memories. After passing this huge MUX the data byte leaves the memory control unit and is fed back into the RAM module where the data byte has to pass a last address MUX before it reaches its destination register. In our following investigations we focused on the major leakage caused by the main data MUX in the memory control unit.

Pinpointing the Leakage in iMDPL

In a third step we thoroughly investigated the data signals propagating through the main memory MUX by comparing different simulations with different mask values where different byte values were processed. First, we focused on two simulations with mask $m = 0$ and investigated the data bus between the memory control unit and the RAM module. The data bus showed a data dependent timing in the output signals do<0..7> coming from the memory control unit. We picked one of the data bits for a detailed investigation. The data signal do<6> consists of two complementary wires do_m<6> and $\overline{do}_m$<6>. In one simulation we moved byte value $0xFF$ in the other one the inverted byte value $0x00$, in both cases mask $m = 0$. In the first case do_m<6> goes HIGH in the evaluation phase and in the second case $\overline{do}_m$<6> goes HIGH in the evaluation phase. Between the two cases we discovered a timing difference of 0.267 ns, i.e. $\overline{do}_m$<6> goes earlier to HIGH than do_m<6>. The two complementary wires originate from buffers and each one is routed to approximately 9 widely distributed combinational iMDPL cells. Figure 5.17 depicts the routing of the wires do_m<6> (red) and $\overline{do}_m$<6> (blue). The sources of both wires (outputs of the buffers) are located bottom left, marked with O, each destination gate is marked with X. The dimensions of the depicted area on the chip die is approximately $1 \times 2\,mm$, which represents quite long distances for these two wires. The fact that iMDPL does not demand balanced routing results in significant differences in the length and the electrical characteristics of complementary wires. Such differences cause a data dependent signal timing, which results in a data dependent switching behavior of following combinational cells and thus causes a leakage in the iMDPL circuit. We further

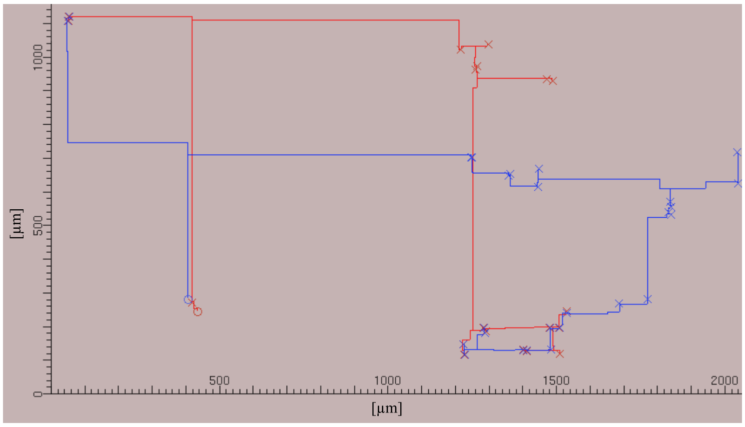


Figure 5.17: Routing of two complementary wires do_m<6> (red) and $\overline{do_m}$<6> (blue) in the GRANDESCA MCcore iMDPL. Both wires originate from buffers located bottom left, marked with an O; the signal destinations at logic gates are marked with an X.

added the simulations with mask $m = 1$ to our comparison and extended our evaluation to all 8 data bits. It turned out that each complementary wire pair shows slight data dependent differences in the signal timing. In case of MDPL the early propagation effect was very similar for each wire, i.e. the signal delays were approximately the same for a whole data bus. However, in case of iMDPL each wire pair shows a different timing behavior, depending on the specific routing properties of each single wire. For each wire pair do_m<0..7> and $\overline{do_m}$<0..7> the delay time $t_{dom} - t_{\overline{dom}}$ can be positive, negative, or even very close to zero when the data value changes. The data signals also show a severe timing dependency on the mask value. The signal delays of the 8-bit output signal of the memory multiplexer in the MCcore iMDPL are shown in Table 5.5. The table shows the relative delays of the data bus do_m<0..7> / $\overline{do_m}$<0..7> as well as the select signal sel_m / $\overline{sel_m}$ of the MUX in all combinations of $do = 0xFF$, $do = 0x00$, $m = 0$, and $m = 1$. Note that the value of sel was constantly 1 as we always performed a MOV operation. The delay values in the table show that a change of the data value do has an inconsistent effect on the single data bits: some of the data bits almost show no delay at all (e.g. do<5>) and some show a significant positive or negative delay (e.g. do<4>, do<6>). The results also show the following: the select signal sel of the MUX is not affected by changes in the data value do, but there is a massive dependency on the mask value m. A mask value of $m = 1$

causes $\overline{sel_m}$ to switch to HIGH 2.417 ns earlier than sel_m in case $m = 0$, which consequently affects the delays of the data signals *do*.

Table 5.5: Relative signal delays of the data bits *do*<0..7> the the select signal *sel* of the main memory MUX in the GRANDESCA MCcore iMDPL depending on the mask value data value *do* and the mask m.

	$m = 0$ $do =$ 0xFF (reference)	$m = 0$ $do =$ 0x00	$m = 1$ $do =$ 0xFF	$m = 1$ $do =$ 0x00
do<0>	0 ns	$-0.061\,ns$	$-1.833\,ns$	$-1.901\,ns$
do<1>	0 ns	$-0.229\,ns$	$-2.406\,ns$	$-2.506\,ns$
do<2>	0 ns	$-0.232\,ns$	$-1.553\,ns$	$-1.555\,ns$
do<3>	0 ns	$+0.215\,ns$	$-1.908\,ns$	$-2.269\,ns$
do<4>	0 ns	$+0.379\,ns$	$-1.550\,ns$	$-2.012\,ns$
do<5>	0 ns	$+0.004\,ns$	$-1.406\,ns$	$-1.489\,ns$
do<6>	0 ns	$-0.267\,ns$	$-1.257\,ns$	$-0.961\,ns$
do<7>	0 ns	$+0.234\,ns$	$-1.221\,ns$	$-1.441\,ns$
	$sel_m = 1$ $\overline{sel_m} = 0$		$sel_m = 0$ $\overline{sel_m} = 1$	
sel	0 ns		$-2.417\,ns$	

Mask-Dependent Switching of iMDPL Cells

Based on our simulation we were able to track down the reason for the dependency of the signal timings on the mask value. It turned out that significant timing differences between the two mask trees m and $\overline{m}$ occur in the circuit. Figure 5.18 depicts the timing conditions of the masked signals processed by the data MUX: input bus di_m<0..7> / $\overline{di_m}$<0..7>, select signal sel_m / $\overline{sel_m}$, and output bus do_m<0..7> / $\overline{do_m}$<0..7>. A MUX in an iMDPL circuit mainly consists of iMDPL NAND cells, i.e. the combinational circuit realizing the MUX starts evaluating with the last signal reaching its differential state. Our simulations showed that the input data di_m<0..7> / $\overline{di_m}$<0..7> originating from a register in the RAM module arrives at the MUX shortly after the falling clock edge t_0, i.e. the beginning of the evaluation phase. In contrast, the select signal sel_m / $\overline{sel_m}$ arrives considerably later at the MUX and shows a timing dependency on the mask value. As the select signal arrives later than the input data, the MUX starts evaluating depending on arrival of the select signal, and hence, the MUX passes the mask dependency on to the output data do_m<0..7> / $\overline{do_m}$<0..7>. We were also able to track down the source of the mask dependent timing behavior of the MUX select signal. The select signal itself originates from another MUX processing the actual command of the microcontroller. The command signal

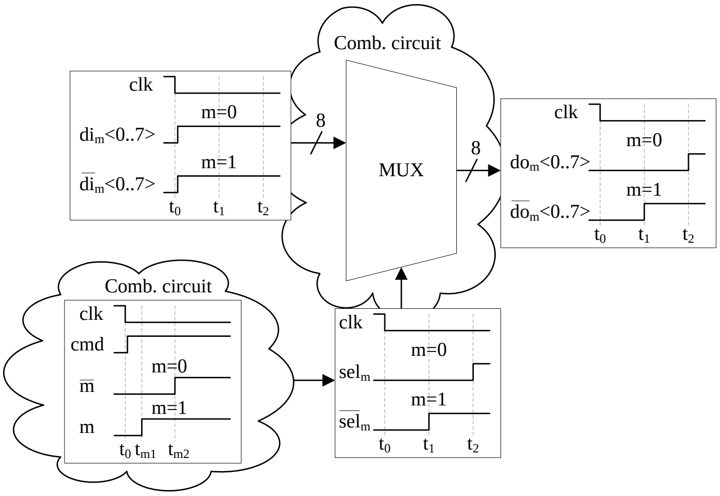


Figure 5.18: Timing relations of the data signals on a symbolic iMDPL multiplexer

cmd originating from a register in the memory control unit propagates through a combinational circuit, of course, together with the masks m and $\overline{m}$. The signal *cmd* arrives shortly after the falling clock edge t_0 at the combinational circuit, but the masks only arrive some time after t_0 due to the fact that the mask unit in an iMDPL circuit provides new masks $m/\overline{m}$ with the falling clock edge and the mask signals first need to propagate through the considerably large mask trees. More important, the complementary mask signals show a significant timing difference as iMDPL does not dictate special constraints for synchronizing the mask trees, which causes significant differences in the electrical characteristics of the two signal distribution networks. These differences result in a data dependent arrival time of the mask values at the combinational circuit processing *cmd*: t_{m2} in case $m = 0$ and t_{m1} in case $m = 1$, i.e. the combinational circuit passes the mask dependency on to the select signal. It is also probable that the different electrical characteristics of the two mask trees results in mask-dependent differences in the power consumption of an iMDPL circuit.

Another important fact we discovered in the simulation results is the following: in the precharge phase the investigated iMDPL cells related to the MUX show a mask-dependent precharge timing. The data signals show a relatively high logic depth compared to the mask signals: we discovered that the mask values arriving at the iMDPL cells have a logic depth of approximately 8, i.e. the mask values propagate through 8 cells (mostly buffers) before they arrive at the iMDPL cells. The data signals di_m<0..7> / $\overline{di}_m$<0..7> however propagate through more than 24 iMDPL cells, which in turn have a minimum logic depth of 4 (not counting more complex gates like XOR/XNOR). This results in a total logic depth of more than 96, and hence, the mask signal is the first signal that arrives at the iMDPL cells in the MUX. In the precharge phase, this results in a mask dependent precharge timing of many iMDPL cells. On the one hand, this explains why there is almost no leakage in the precharge phase: the data-dependent evaluation of iMDPL cells does not occur to the same extent because most of the iMDPL cells are precharged with the mask signal. Furthermore, the precharging of a major part of the iMDPL circuit with the mask signal happens within a very short period of time, which causes huge amounts of switching noise that disguises the switching activities of the internal data buses. On the other hand, due to the fact that the mask signals $m/\overline{m}$ arrive at slightly different times at the iMDPL cells (similar to the evaluation phase), a large part of an iMDPL circuit gets precharged at mask-dependent moments in time. This may result in detectable mask dependencies in the power consumption.

Our investigations have shown that the data dependent timing differences between single bits of a data bus are inconsistent and deviate case-by-case: some complementary wires show significant timing differences but some wires show almost no data dependent behavior at all. This is the reason why the leakage in an iMDPL circuit in a typical case is smaller compared to MDPL: the early propagation effect that occurred in MDPL caused higher timing differences (approximately 1 ns) and, more important, the differences were consistent for a whole data bus.

5.5.4 Detectability of the Mask Value in iMDPL

The exploration of the leakage in iMDPL in the previous section has shown that the mask value has a significant influence on the overall timing of the iMDPL circuit: the data signals propagating through the memory multiplexer show a clear dependency on the mask: a mask value of $m = 0$ introduces a signal delay of approximately $2\,ns$ compared to $m = 1$. Such an influence on the switching behavior of an internal signal bus suggests that also the instantaneous power consumption of an iMDPL circuit shows a dependency on the actual mask value. By means of extracting information about the mask value from power traces an attacker could weaken or even completely disable the effect of masking in an iMDPL circuit. Tiri and Schaumont have addressed this special vulnerability of masked logic styles [TS07, ST07]. In [TS07] the authors evaluated the random switching logic (RSL) style, a masking logic style that randomizes the data processed within a device. Based on a test circuit consisting of an AES S-box and an XOR operation the RSL style was analyzed. The results revealed that the toggle counts in each cycle are related to the mask value. In [ST07] the authors extended their investigations and evaluated toggle count traces from an MDPL test circuit. Their results based on the probability density function (PDF) and showed that the toggle counts have a significant relation to the mask value in the simulated MDPL circuit. Mulder et al. also performed experiments based on the PDF on an MDPL prototype chip [MGPV09]. They generated PDF histograms from real measurements of an MDPL circuit supplied with a random mask bit the authors separated the power traces into subsets. Correlation-based PA attacks on these subsets showed that the effect of masking can be significantly reduced.

Gathering Mask Information in iMDPL

The GRANDESCA chip provides the feature to observe the mask bit during operation (cf. Section 5.4.1). Furthermore, the 8051-compatible microcontroller in combination with our PRNG functions allows a cycle accurate control of the mask bit in the iMDPL circuit. This way we were able to perform a thorough investigation of the masking technique in iMDPL based on real measurements and to verify our findings by the comprehensive knowledge of the actual mask value active in the circuit.

Although we took additional measures to minimize the effect of the activated mask observation feature, we exercised caution during our experiments and performed the collection of mask data and the actual power measurements in two separated steps. First, we executed MOV operations in the MCcore iMDPL with randomly chosen byte values and measured the instantaneous power consumption as well as the mask output. In a second step we performed MOV operations with exactly the same byte values but with a deactivated mask observer. With this approach we obtained unaffected power traces besides having total knowledge about the active mask values in the iMDPL circuit from the first measurement we performed.

In order to demonstrate the effect of the mask observer on the power consumption, in a first experiment we compared the power traces obtained from the GRANDESCA chip in two cases: (1) activated mask observer and (2) deactivated mask observer. We extracted the information about the mask value directly from the observed data and compared the power traces. In order to improve the visibility of the differences we reduced the noise in the traces by averaging 100 traces for the two cases (activated and deactivated mask observer), in each of the cases we averaged $m = 0$ and $m = 1$ traces. Figure 5.19 (left) shows a small section of the power traces recorded with deactivated mask observer, the two traces corresponding to $m = 0$ (gray) and $m = 1$ (black) can barely be distinguished. On the other hand, Figure 5.19 (right) shows the same section of power traces recorded with activated mask observer. In this case the power traces show significant differences depending on the mask value m.

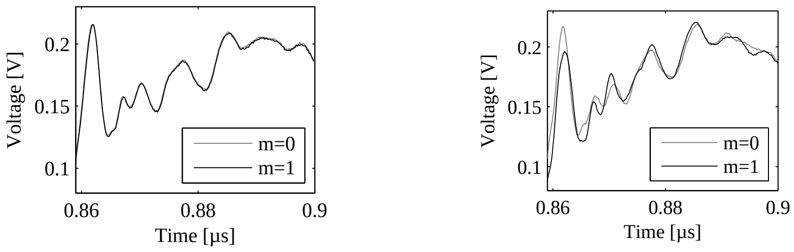


Figure 5.19: Comparison of averaged traces, each generated from 100 power traces; left: deactivated mask observer; right: activated mask observer; the power traces in case mask $m = 0$ are plotted in gray, traces for $m = 1$ are plotted in black.

Based on these findings we completely discarded the power measurements recorded during the mask observer was active. Analyzing these traces would definitely have resulted in severely falsified results of our following experiments. These results indicated that the usage of the mask reader module would also most probably have significant influences on the power measurements, and hence, we did not perform any experiments with the module.

Extracting Mask Information Directly from Power Traces

In the following experiment we tried to verify that the single mask bit in an iMDPL circuit has a traceable effect on the power consumption of our chip as stated in [TS07, ST07, MGPV09]. It is assumed that the power consumption depends on the mask value to a certain degree: a mask value of $m = 0$ causes a power consumption γ, a mask value of $m = 1$ causes a power consumption δ, and it is further assumed that $\gamma \neq \delta$. Based on this assumption we tried to extract information about the mask value directly from the measured power traces. In order to verify the success rate of the information extraction we compared the extracted mask values to the actually observed mask values obtained from the mask observer measurement. We published practical results dealing with this topic in [KP09]. During the work for this thesis we revised our techniques and evaluations of extracting information about the mask value directly from power traces. The results of our work are presented in the following.

The first step for extracting mask information was to profile the power consumption in the time range where the correlation peak occurred. We performed the profiling by generating histograms of several different sections in the traces averaged over many different window lengths, i.e. for each recorded trace we calculated the mean value of several sequential data points over the measurement time. Several window lengths resulted in promising histograms, in the end we chose a window length of 1.6 ns and averaged the data points starting from 2.86 µs, as depicted in Figure 5.20. The result of the profiling based on 40 000 averaged power traces is shown in Figure 5.21.

In a next step we separated the 40 000 traces based on the averaged values in the histogram into two groups: traces having an average value $v <= 0.0457$ and traces having an average value $v > 0.0457$. We then assumed that these two groups are related to the actual mask value that was active in the iMDPL circuit. We then checked the accordance of the two groups with the mask information obtained from the mask observer and obtained astonishing results: 39 073 out of 40 000 traces matched, i.e. the extraction of mask information on the MCcore iMDPL has a success rate of > 97%. It also turned out that an average value $v <= 0.0457$ corresponds to a mask value $m = 1$ and $v > 0.0457$ corresponds to $m = 0$.

This experiment shows that the mask value in an iMDPL circuit can be revealed by analyzing averaged power traces. Hence, the effect of masking can almost be completely neutralized, or we can at least introduce a significant mask bias by separating the traces. We further examined the impact of a bias in the mask value by performing PA attacks on defined subsets of the power traces.

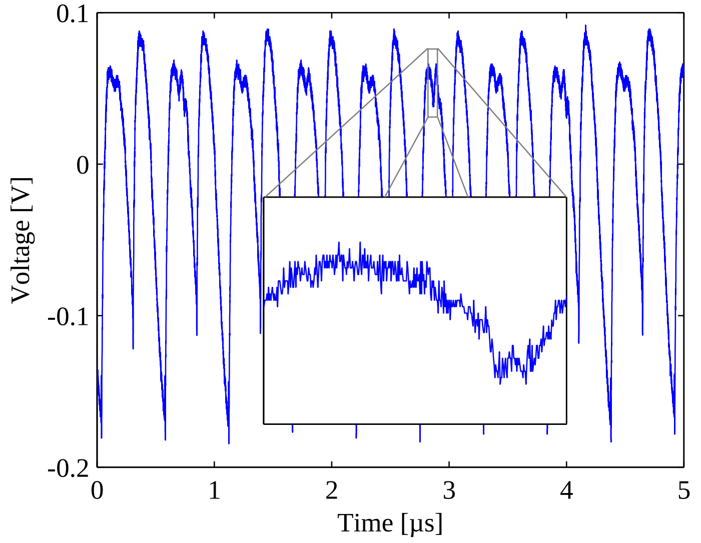


Figure 5.20: 100 power traces of the MOV operation in the MCcore iMDPL, the zoomed area depicts the averaged part of the traces to generate the histogram shown in Figure 5.21.

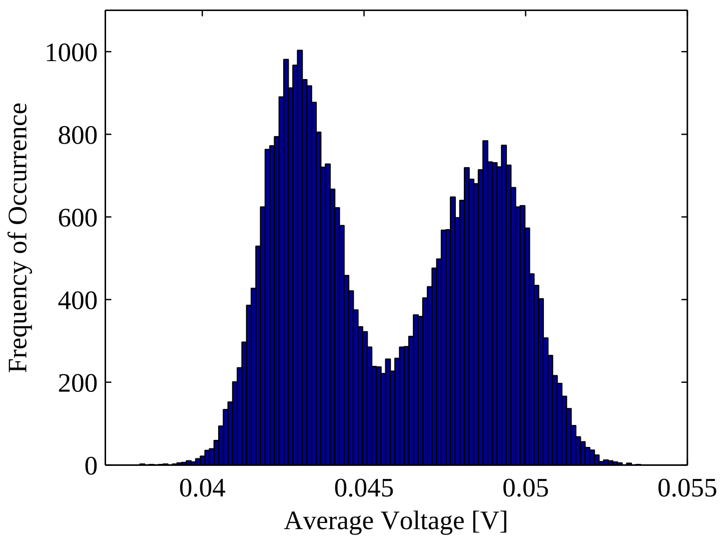


Figure 5.21: Histogram of 40 000 averaged power traces using a window length of 1.6ns, starting at 2.86 µs as shown in Figure 5.20.

Exploiting Mask Information to Improve PA Attacks

In a first step we showed that the knowledge of the mask value can be used to improve an attack on an iMDPL implementation. In one case we used a standard set of 10 000 mixed traces, in the other case we selected a subset of power traces for the attack according to a mask value $m = 0$ in the clock cycle where the correlation peak occurs. Figure 5.22 (left) shows the result of a correlation attack on the standard set of 10 000 mixed traces. The correlation value is 0.0386, which corresponds to 19 000 required power traces, i.e. the attack was not successful. We can see that the correct hypothesis cannot be distinguished from the incorrect ones. Figure 5.22 (right) shows the result on a subset of traces where $m = 0$: the correlation peak of 0.0624 is significantly higher, resulting in 7 200 required traces. As we can see, in this case the correct hypothesis is clearly distinguishable, the attack was successful.

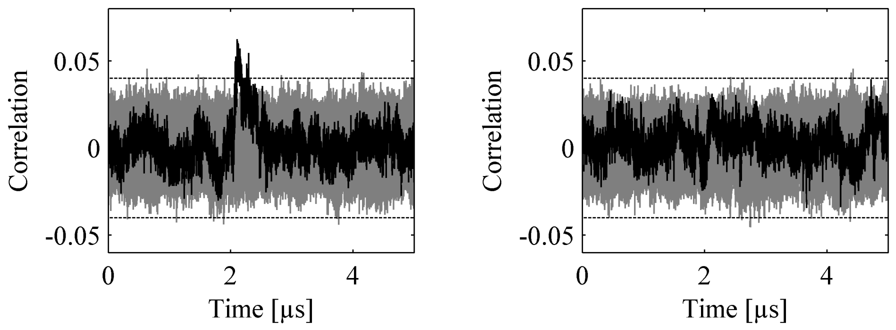


Figure 5.22: Result of the PA attacks on the GRANDESCA MCcore iMDPL: internal MOV operation in the IRAM, 10 000 traces, HW power model; left: standard set of traces with mixed mask value; right: subset of traces with mask $m = 0$; 40 correlation traces, 39 traces represent randomly chosen byte values; the trace for the correct hypothesis is plotted in black.

We performed further PA attacks on various subsets of power traces having a different mask bias. Figure 5.22 (left) corresponds to a mask bias of 0.5, i.e. 50% of the mask values in the traces were 0 and the other 50% were 1. For example: Figure 5.22 (right) corresponds to a mask bias of 1: 100% of the mask values were 0. We performed the attacks using a constant amount of 10 000 power traces, manually introducing different mask biases of 0.7, 0.8, 0.9, and 0.97. The results of our attacks based on the obtained correlation values are summarized in Table 5.6. A mask bias of > 0.7 already significantly reduces the required power traces to perform a successful attack. Approximately 7 600 traces were required in case the mask bias is 0.97, which represents our mask extraction success rate of 97%. Hence, a measurement of 15 200 traces and the separation of the traces in two subsets according to the generated histograms results in a subset containing 7 370 traces where $m = 0$ and 230 traces where

$m = 1$. Based on this subset we can perform a successful PA attack as our results showed. A comparison to an attack based on non-preprocessed traces which required approximately 19 000 traces, the 15 200 required traces represent a reduction in measurement effort of 20%.

Table 5.6: Results of PA attacks on the internal MOV operation in the MCcore iMDPL with different mask biases: 10 000 traces in each case; traces with mask $m = 0$ represent the majority in case of mask bias > 0.50.

mask bias	0.50	0.70	0.80	0.90	0.97	1.0
$\rho_{ck,ct}$	0.0386	0.0502	0.0497	0.0565	0.0602	0.0624
n_{min}	19 000	11 100	11 300	8 800	7 600	7 200

This experiment has shown that the filtering of power traces based on the generation of histograms may introduce a significant mask bias in an iMDPL implementation. The generation and the evaluation of countless histograms was quite an expensive task, compared to the only slight reduction of required power traces. However, the results clearly show that the unbalanced mask trees represent a major source of the leakage in iMDPL and that the mask values in an iMDPL circuit can be extracted directly from the measured power traces with a rather high confidence of 97%.

Conclusions

Starting from evaluations of the original MDPL style we presented an improved version called iMDPL that significantly reduced the effect of early propagation. The number of required power traces to perform a successful PA attack is a good measure to determine the effectiveness of a logic style. As a reference we used an unprotected microcontroller implementation in CMOS logic, where we required approximately 230 traces. The same microcontroller implementation in the original MDPL style increased the number of required traces only to 700 (factor 2-3), due to the occurrence of early propagation in the logic style. The evaluation of our proposed iMDPL style showed that the effect of early propagation could be significantly reduced, resulting in a number of required power traces of approximately 17 000 (factor >70 compared to CMOS, 25 compared to MDPL). Certainly, the increase in security against PA attacks comes at a price: a significant increase in area requirement (MDPL 5x, iMDPL 20x compared to CMOS) and a reduced maximum clock frequency.

Our detailed evaluations of the iMDPL style revealed that imbalances between complementary wires are the source of the leakage in iMDPL. Also the mask signal trees show significant differences in the electrical characteristics, which results in the detectability of the mask value directly from the measured power traces. The results presented in this chapter also showed that a combination of different evaluation techniques is a very powerful tool to completely reveal weaknesses in a PA-resistant logic style.

6 Integration of PA-Resistant Logic Styles in an Embedded Processor

In case of the GRANDESCA chip presented in the previous chapter we implemented a whole microcontroller and an AES module in a secure logic style. The main drawbacks of this approach are a significant increase of the required area and a lack of flexibility. In this chapter we present a different approach: a combination of cryptographic instruction-set extensions (ISEs), a secure logic style, and an architectural masking technique. This approach provides a certain level of resistance against PA attacks for modern processor platforms and limits the area overhead by implementing only a small part of the processor in a costly secure logic style. Furthermore, this approach provides a high degree of flexibility when implementing a cryptographic algorithm due to the employment of ISEs.

In this chapter, we first give an introduction to ISEs and point out the advantages compared to both pure hardware approaches and pure software approaches. Then, we present the *secure-zone* approach, where all critical operations are executed within a single module in a processor. The so-called secure zone is implemented in a secure logic style and all data outside of the secure zone is strictly masked.

Subsequently, we present the implementation of the secure-zone approach in a 32-bit SPARC V8 Leon3 processor which was a joint work together with Stefan Tillich and Alexander Szekely and which was published in [TKS10] and in [TKS11]. We present our prototype chip which was designed in the course of the POWER-TRUST [POW10] project. The chip contains three secure zones implemented in different logic styles. We discuss the architecture of the chip and the implemented AES ISEs and present the implemented logic styles (CMOS logic, iMDPL, and DWDDL) in the secure zones. In doing so, we first introduce the original WDDL style and justify our decision for implementing an advanced version of WDDL, called double WDDL (DWDDL), on the POWER-TRUST chip. Further, we provide details for implementing the DWDDL style on an ASIC and present the results of the fabricated chip. It turns out that the secure-zone approach is able to significantly reduce the overall area overhead compared to a pure hardware implementation realized in a secure logic style.

We further present a security evaluation of different AES implementations on the POWER-TRUST chip by comparing different AES implementations: (1) a pure software AES implementation, (2) an implementation based on basic AES ISEs (without a secure logic style and architectural masking), and (3) an AES implementation in each of the three secure zones. The results show that the secure-zone approach significantly increases the resistance against PA attacks even if the secure zone itself is implemented in plain CMOS logic. Further, it turns out that this approach reduces the area overhead caused by a secure logic style.

In the last part of this chapter we elaborate on the results obtained from the executed PA attacks on the different AES implementations. As the evaluation of the POWER-TRUST chip is ongoing research, we also discuss remaining open questions and planned future work.

6.1 Instruction-Set Extensions

Contrary to a pure AES hardware module working as a coprocessor attached to a microcontroller, a cryptographic algorithm can also be implemented purely in software and executed on a processor. The main advantage of a software implementation is flexibility: once the design of an AES hardware module is fixed and the module is fabricated in real silicon, there is no chance to execute any new operation or to change anything in the sequence of the performed operations. The hardwired combinational blocks implementing core operations of the algorithm (e.g. an S-box or a MixColumns transformation) and the fixed state machine do not allow any variations. Hence, in case of a hardware module it is impossible to extend the functionality of the module afterwards, e.g. add support of different key sizes or a new mode of operation [TG06]. In contrast, a pure software implementation can easily be adapted in order to fulfill changed or additionally introduced requirements, it is only a matter of adding some additional lines of code.

The code size, the amount of memory required, and the associated computation time represent major disadvantages of a pure software implementation, especially when destined for constrained devices like cell phones or sensor nodes. Limited computational power may further increase the computation time of a cryptographic algorithm implemented in software, which results in a significantly decreased total operating time due to a limited amount of energy. In a worst case, memory limitations may even completely inhibit the computation of more complex operations. Actually, there were proposed quite fast software AES encryptions based on high-end NVIDIA GTX 295 architectures achieving up to

30 Gbps [OBSC10], which is very fast but also tremendously far away from the computational power of embedded processors. However, a software implementation running on a processor can never outrun a matching pure hardware implementation: in [HV04] an AES processor capable of up to 70 Gbps throughput was presented. Note the significant difference in the process technologies: the AES processor is only based on 180 nm whereas the GTX 295 is based on a 55 nm CMOS technology, i.e. implementing the AES hardware module from [HV04] in a 55 nm technology would result in a further significant increase in throughput.

Hence, a hardware module seems more appropriate for adding cryptographic functionality to an embedded system. The lack of flexibility could be compensated by implementing a cryptographic hardware module capable of many additional kinds of features like different key sizes. Unfortunately, this would significantly increase the required chip area and the associated production costs, which is intolerable in case of systems produced in extremely high volumes.

The implementation of ISEs combines the advantages of both pure hardware and pure software approaches. On the one hand, individual core operations of a cryptographic algorithm can efficiently be implemented in terms of area or speed. Different implementations of each operation favoring either high speed or low area can be implemented, thus leaving space for adaption to the respective overall requirements of the system. On the other hand, the flexibility of a software approach persists as the custom instructions utilizing the operations implemented in hardware can be executed in an arbitrary order. The program flow can easily be adapted, and the elimination of implementing complex core operations in software significantly reduces the code size.

With regard to implementing hardware countermeasures like PA-resistant logic styles, the only option to protect a pure software implementation is to implement the whole processor in a secure logic style, similar to the implementation presented in the previous chapter where we implemented the whole 8051-compatible microcontroller in the iMDPL style. In case of rather complex processors this approach would result in tremendous area requirements due to the increased area demands of most secure logic styles. The usage of ISEs enables us to implement only small parts of the processor, namely the hardware modules implementing the core operations of a cryptographic algorithm, in a costly secure logic style. During the last years, Tillich and Großschädl were working on several proposals of optimized AES ISEs, also with regards to PA attacks. In the next section we present a combined approach for implementing AES ISEs protected by a secure logic style.

6.2 The Secure-Zone Approach

The *secure-zone* approach was first introduced by Tillich and Großschädl [TG07]. Detailed work on the implementation of the secure-zone approach in a 32-bit SPARC V8 Leon3 processor was published by Tillich et al. in [TKS10]. In the following we introduce the implementation of this approach in combination with secure logic styles.

A secure-zone module is integrated into a processor as conventional functional unit (FU), which enables the execution of additional cryptographic instructions. Cryptographic operations actually transforming critical data (i.e. data that is related to a secret key) is confined to the secure zone, outside of the secure zone the data is just forwarded and stored. In order to ensure a certain level of resistance against PA attacks, the secure zone itself needs to be implemented in a secure logic style. All critical data values outside of the secure zone are protected by architectural masking. The architecture of the secure zone is depicted in Figure 6.1: the secure zone receives the masked operands $op1_m/op2_m$ and their register addresses $addr_{op1}/addr_{op2}$. Based on the register addresses the *mask unit* identifies the corresponding masks m_{op1}/m_{op2} in the mask storage and removes the mask from the operands by an XOR operation. All operations in the FU are performed on the unmasked operands $op1/op2$, which yield the result res. The *mask generator* generates a new mask for the result m_{res}, the mask unit stores information about the mask using the register address of the result $addr_{res}$, and the masked result res_m leaves the secure zone.

The overall security of the secure-zone approach is primarily based on the PA resistance of the secure logic style. First, each operation on the unmasked values within the secure zone is protected by the secure logic style in order to impede or even completely prevent PA attacks. Second, the handling and the storage of the masks is also protected by the secure logic style. Third, each particular architectural mask is only used once, which protects the masks from being attacked outside of the secure zone by higher-order PA attacks. The main advantage of the secure-zone approach is that only a relatively small part of the whole processor has to be implemented in a costly (in terms of area requirement and energy consumption) secure logic style.

6.3 Implementation of the Secure-Zone Approach in a SPARC V8 Leon3 Processor

In the course of the POWER-TRUST project [POW10] we implemented the secure-zone approach in a 32-bit SPARC V8 Leon3 embedded processor [Gai10], combined with secure logic styles as well as architectural masking. The main purpose of the project was the realization of a PA-resistant modern processor platform and to prove the practicality and the advantages of the secure-zone approach, namely an economical approach for implementing secure logic styles besides providing a certain level of security against PA attacks. In the following we present the main features of the POWER-TRUST chip as well as the implemented secure logic styles.

6.3.1 Architecture of the Secure Processor

In order to provide increased comparability, we implemented AES using different approaches. First of all, the 32-bit processor itself provides the possibility to execute a pure software implementation of AES. The software implementation

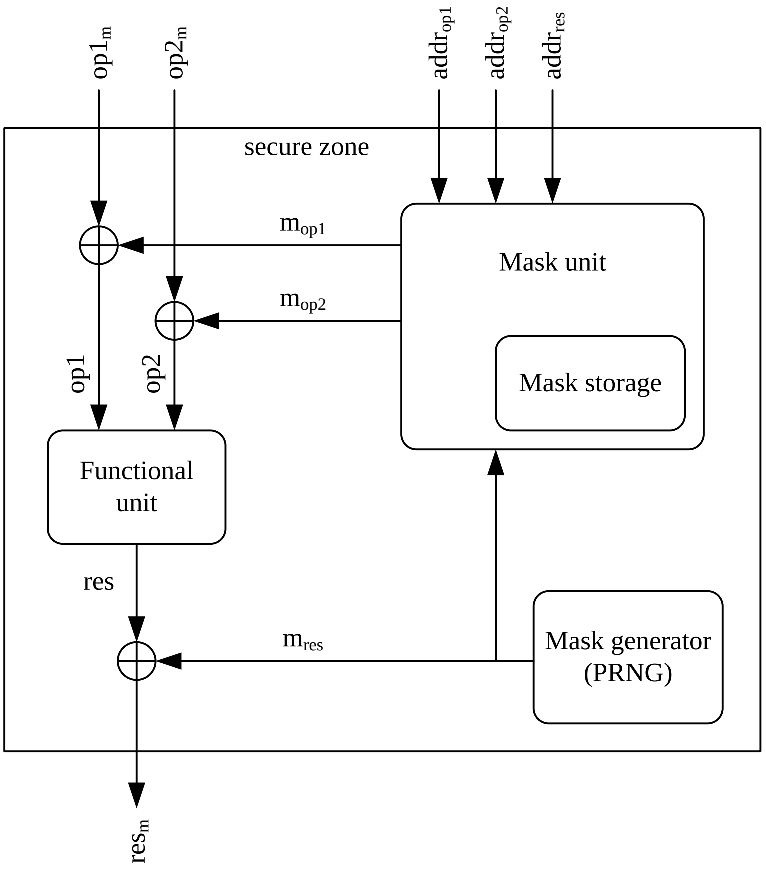


Figure 6.1: Architecture of the secure zone; the mask is removed from the input data, critical operations are performed in the functional unit protected by a secure logic style, and the masked result leaves the secure zone.

we used for evaluations and for comparison with the instruction-set approaches is based only on native SPARC V8 instructions and uses lookup tables for the S-box operation. Further, we implemented basic ISEs according to [TG06] in a standalone FU called *AESREF*, i.e. without combining the FU with a secure logic style or architectural masking. The ISEs eliminate the need of lookup tables for the S-box operation. As shown in [Man03a], the utilization of S-box lookup tables poses a significant threat with regard to PA attacks. The approach based on basic ISEs shall demonstrate that the elimination of table lookups already has a positive effect on the PA resistance of an implementation.

The third and main part of our secure processor is represented by three secure-zone modules implemented in different logic styles: CMOS logic, iMDPL, and DWDDL. Details on the implemented logic styles will be given in Section 6.3.2. The three secure zones are contained in a single wrapper module which provides additional custom instructions to select the active secure zone (by writing to a particular configuration register) and to control the PRNG (load seed, start, stop) that provides one mask bit per cycle for the secure zone implemented in iMDPL. At most one secure zone can be active at a time, while the other two secure zones are deactivated: the clock signal is gated and all inputs are kept at zero in order to prevent any interference. It is also possible to deselect/deactivate all secure zones to be able to solely investigate the Leon3 processor.

Implemented Instruction-Set Extensions

We implemented the following AES ISEs in the FUs of AESREF and the three secure zones according to the "Advanced Word-Oriented AES Extensions with Implicit ShiftRows" proposed in [TG06]: *sbox4s*, *mixcol4s*, *sbox4r*, *isbox4s*, and *imixcol4s*. We also implemented some additional instructions in the FU (XOR, OR, AND, NOT) in order to provide additional functionality. Thus, our instruction set provides all necessary functions to securely implement the AES algorithm.

Mask Generation in the Secure Zone

For the generation of a fresh 32-bit mask per cycle for masking the result of the secure-zone instructions we implemented a maximum-length LFSR with an internal state of 128 bit. The single mask bit per cycle for the secure zone implemented in iMDPL is provided by an additional 64-bit maximum-length LFSR according to [Alf96].

6.3.2 Implemented Logic Styles

One of the secure zones was implemented in static complementary CMOS logic and serves as a reference. Although this secure zone is implemented in CMOS logic, which is well known to be highly vulnerable to PA attacks, we expected to encounter already a significant increase in resistance against first-order PA attacks due to the confinement of critical operations to the relatively small FU within the secure zone. Furthermore, the architectural masking approach reduces the leakage of data values outside of the secure zone.

Based on our first experiments with the GRANDESCA chip we knew that iMDPL increases the resistance against PA attacks to a certain degree compared to unprotected CMOS logic. Furthermore, iMDPL can easily be implemented using our netlist converter (cf. Section 3.3) and standard cell libraries. Hence, we decided to implement one of the secure zones in iMDPL. As the implementation of the POWER-TRUST chip was performed before we finished our detailed evaluation of the GRANDESCA chip (cf. Section 5.5.3), we did not implement any further improved version of iMDPL.

The third logic style we chose for implementation is double wave-dynamic differential logic (DWDDL) proposed by Yu and Schaumont [YS07]. The DWDDL style is an enhancement of the WDDL style proposed by Tiri et al. [TV04a] with the goal of solving its inherent wire balancing issues, especially when implemented on an FPGA.

6.3.3 Wave Dynamic Differential Logic

The WDDL style proposed by Tiri et al. [TV04a] follows the dual-rail precharge (DRP) principle and tries to keep the power consumption of a circuit constant in each clock cycle. A basic WDDL AND cell consists of one CMOS AND gate and one CMOS OR gate as shown in Figure 6.2. If all inputs a, b, $\overline{a}$, and $\overline{b}$ of the compound cell are 0 (i.e. the inputs of the AND or OR gate are 0), also the outputs q and $\overline{q}$ of the cell are 0. As these outputs represent the inputs to the next combinational stage in the circuit, a so-called *precharge wave* automatically propagates through the whole WDDL circuit and ensures that every compound WDDL cell in the circuit has exactly one switching event in the precharge phase and one switching event in the evaluation phase. This way the occurrence of glitches is prevented in WDDL. Furthermore, the PA resistance of WDDL relies on balancing the electrical characteristics of the complementary wires.

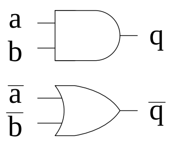


Figure 6.2: Schematic of a WDDL AND cell: it consists of one AND and one OR gate.

The evaluation of a prototype chip implemented in WDDL [THH+05] showed a significant increase of the resistance against PA attacks compared to an unprotected CMOS implementation (consider cautionary notes about such comparisons in Section 4.4). Certainly, the resistance against PA attacks comes at a price: the required area of a hardware implementation is increased approximately by a factor of 3 and the maximum clock frequency is around one half compared to an implementation in CMOS logic. Furthermore, in order to achieve the same data rate in a WDDL circuit as in a plain CMOS circuit the clock frequency has to be doubled due to the master-slave flip-flop structure in WDDL, which is necessary to achieve a reset of the WDDL compound register structure in every clock cycle. This circumstance increases the effort of implementing a WDDL circuit if only some sensitive parts of a design shall be secured against PA attacks. In this case, two clock domains are needed: one clock domain for the WDDL circuit and another clock domain running at half frequency (e.g.

derived from the faster domain) for the surrounding plain CMOS part of the design. Evaluations of Suzuki et al. showed that WDDL also suffers from early propagation [SS06]. Like in MDPL, different arrival times of the input signals at the compound WDDL cells result in a temporary data-dependent switching behavior of the circuit, which results in a significant reduction of the resistance against PA attacks. Table 6.1 shows an example of early propagation by means of a basic WDDL AND cell. If one of the input signals $a/\overline{a}$ and $b/\overline{b}$ arrives early (E), the evaluation time of the outputs $q/\overline{q}$ (E …early, L …late) correlates with the timing of the early input signal: if inputs a and $\overline{a}$ arrive earlier than b and $\overline{b}$, $\overline{q}$ evaluates early in both cases where $a = 0$. In case $a = 1$, q and $\overline{q}$ evaluate late with signal $b/\overline{b}$. The same conditions apply in case the signals $b/\overline{b}$ arrive early at the WDDL cell.

Table 6.1: Transition times of the outputs q and $\overline{q}$ of a basic WDDL AND cell in case of different arrival times (E …early, L …late) of the input signals a and b.

				a: E		b: E	
a	b	$\overline{a}$	$\overline{b}$	q	$\overline{q}$	q	$\overline{q}$
0	0	1	1	0	1_E	0	1_E
0	1	1	0	0	1_E	0	1_L
1	0	0	1	0	1_L	0	1_E
1	1	0	0	1_L	0	1_L	0

6.3.4 Double Wave Dynamic Differential Logic

One important requirement of WDDL is the balancing of complementary wires, which represents a significant effort in current VLSI design flows. Yu et al. have proposed an idea called double WDDL (DWDDL) [YS07] to overcome placement and routing issues when implementing WDDL on an FPGA in the first place. DWDDL basically duplicates the layout of a conventionally routed WDDL circuit and inverts the input signals of the duplicated WDDL block. Furthermore, the logic gates within the compound WDDL cells in the duplicated WDDL circuit are exchanged.

We adapted this idea to implement DWDDL on an ASIC and hence to minimize the overhead for meeting the placement and routing requirements. Furthermore, a first closer look revealed that a DWDDL circuit also counteracts the existing early propagation problem in WDDL. Unfortunately, at this point we made a mistake that remained undetected for a long time. We performed a theoretical investigation based on an incorrectly designed DWDDL AND cell as shown in Figure 6.3. We correctly inverted the input signals of the complementary WDDL cell ($a_c = \neg a$, $b_c = \neg b$, $\overline{a_c} = \neg\overline{a}$, and $\overline{b_c} = \neg\overline{b}$), but we missed to exchange the logic gates AND and OR of the complementary WDDL cell. Our mistake disguised the vulnerability of DWDDL to early propagation and resulted in an incorrect behavior of the complementary WDDL circuit: the circuit does not compute the complementary values as intended. Actually, a secure zone implemented this way would still have functioned correctly, as the global outputs of the complementary WDDL circuit are discarded during the transition from the secure logic domain back to the single rail CMOS domain.

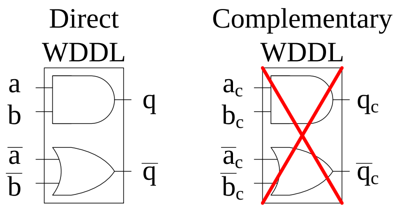


Figure 6.3: Schematic of an **incorrectly** designed DWDDL AND cell: the logic gates in the complementary WDDL cell are not exchanged.

Table 6.2 shows the transition times of the logic gates within each WDDL cell in case the input signals a and b arrive at different moments in time. Regardless of whether a arrives earlier than b or vice versa, one of the logic gates switches its output early (E) and one late (L), independent of the input signals. In this case early propagation is prevented, but the power consumption may show a data dependency due to the non-complementary signal transitions in the duplicated WDDL cell.

Table 6.2: Transition times of the outputs q, $\overline{q}$, q_c, and $\overline{q_c}$ of an **incorrect** DWDDL AND cell in case of different arrival times (E …early, L …late) of the input signals a and b.

				a: E				b: E			
$a/\overline{a_c}$	$b/\overline{b_c}$	$\overline{a}/a_c$	$\overline{b}/b_c$	q	$\overline{q}$	q_c	$\overline{q_c}$	q	$\overline{q}$	q_c	$\overline{q_c}$
0	0	1	1	0	1_E	1_L	0	0	1_E	1_L	0
0	1	1	0	0	1_E	0	1_L	0	1_L	0	1_E
1	0	0	1	0	1_L	0	1_E	0	1_E	0	1_L
1	1	0	0	1_L	0	0	1_E	1_L	0	0	1_E

The correct version of a DWDDL AND cell is shown in Figure 6.4, where the AND and OR gates in the complementary WDDL cell are swapped. The theoretical investigation based on the correct version of a DWDDL AND cell clearly shows that early propagation occurs in the cell: depending on the input signals of the cell the logic gates switch at different moments in time, which potentially decreases the resistance against PA attacks. Table 6.3 shows the transition times of the outputs q, $\overline{q}$, q_c, and $\overline{q_c}$ of a DWDDL AND cell in case one of the input signals a or b arrives earlier at the cell. In case $a = 0$ (i.e. $\overline{a_c} = 0$, $\overline{a} = a_c = 1$) and a arrives earlier than b (a:E), the outputs of the OR gates $\overline{q}$ and q_c evaluate early on arrival of a. If $a = 1$ the outputs of the OR gates and the AND gates evaluate only on arrival of b (late). The same observations apply to the case where b arrives earlier than a (b:E). Similar to MDPL (cf. Section 5.2), the data-dependent evaluation times of the logic gates may cause a significant information leakage in a DWDDL circuit.

We recognized our mistake very shortly before the tape-out date, we still had the time to exchange the logic gates in the complementary WDDL circuit on our chip, but we did neither have enough time to implement an improved version of DWDDL to prevent early propagation nor to switch to another secure logic style.

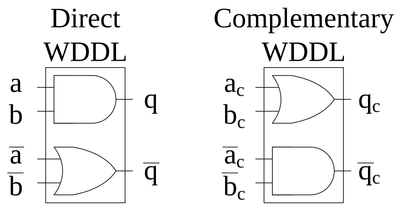


Figure 6.4: Schematic of a **correctly** designed DWDDL AND gate: the logic gates in the complementary WDDL cell are exchanged.

Table 6.3: Transition times of the outputs q, $\overline{q}$, q_c, and $\overline{q_c}$ of a **correct** DWDDL AND cell in case of different arrival times (E ... early, L ... late) of the input signals a and b.

				a: E				b: E			
$a/\overline{a_c}$	$b/\overline{b_c}$	$\overline{a}/a_c$	$\overline{b}/b_c$	q	$\overline{q}$	q_c	$\overline{q_c}$	q	$\overline{q}$	q_c	$\overline{q_c}$
0	0	1	1	0	1_E	1_E	0	0	1_E	1_E	0
0	1	1	0	0	1_E	1_E	0	0	1_L	1_L	0
1	0	0	1	0	1_L	1_L	0	0	1_E	1_E	0
1	1	0	0	1_L	0	0	1_L	1_L	0	0	1_L

In fact, we produced a prototype chip containing two not perfectly secure logic styles. Our PA attacks on the GRANDESCA chip showed a leakage in iMDPL and the theoretical investigation of the correct DWDDL gates has revealed the occurrence of early propagation. However, our primary goal in the POWER-TRUST project was the implementation of the secure-zone approach in an embedded processor. We expected that the utilization of the AES ISEs in a secure zone would already increase the PA resistance to a certain degree compared to an equivalent AES software implementation, even if no secure logic style would have been applied at all. Our second goal was to investigate the practicality and the economy (in terms of area requirements) when actually implementing the secure-zone approach with a secure logic style compared to a naive approach where the whole processor would have to be implemented in a secure logic style.

Special DWDDL Cells

In the following we discuss some special issues we encountered during the implementation of the DWDDL style on our ASIC prototype. We first address the master-slave flip-flop structure proposed by Tiri et al. [TV04a] and the implementation of the consequently required two clock domains in our design. Further, we present our solution for converting the input signals from non-precharged single-rail CMOS to precharged WDDL signals and back to CMOS.

Master-slave flip-flop structure: In our WDDL implementations we followed the approach proposed by Tiri et al. using master-slave flip-flops. During operation, one of the flip-flop stages stores the precharge value and the other stage stores the actual complementary values. In order to achieve the same data rate of a comparable CMOS circuit, the flip-flops have to be clocked with a doubled clock frequency. While the conventional CMOS and iMDPL circuits outside the WDDL secure zone run through one clock cycle (in iMDPL one precharge and one evaluation phase), a WDDL circuit completes two clock cycles: one precharge cycle and one evaluation cycle. Hence, we implemented two clock domains in the whole design:

1. The chip is only fed with one double-speed clock signal *clk2x* which runs through a flip-flop producing the standard speed clock signal *clk*. The *clk* signal is used in all CMOS circuits and in the iMDPL secure zone, additionally it serves as a precharge/evaluation signal for the inputs to the WDDL circuit.

2. The WDDL secure zones are solely operated with the double-speed clock in order to provide the same data rate as the CMOS and iMDPL circuits.

Single rail to dual rail conversion: In a DWDDL circuit the input signals to both WDDL sub-circuits need to have identical timings. Differences between the input signals would counteract the fundamental idea of implementing duplicated WDDL circuits. We realized this requirement by clocking the conversion modules from single-rail CMOS to dual-rail WDDL as shown in Figure 6.5: the single rail data signal d is converted to the dual rail complementary signals by the inverter and the NOR stage introduces the precharge and evaluation phase. The *clk* signal controls the beginning of each phase, hence the timing of the input signals depends on the clock skew. During the synthesis of the clock tree it has to be taken care that the clock signals have as little skew as possible. The same conversion cell is used for the complementary WDDL circuit, where the input signal d is simply inverted.

Dual rail to single rail conversion: When leaving the WDDL domain, the precharge phase has to be removed from the data signals as shown in Figure 6.6 (left): the NOR latch removes the precharge phase.

Furthermore, we need to combine the complementary outputs d and d_c of the two WDDL circuits to one single CMOS signal q as shown in Figure 6.6 (right): the data signal d_c from the complementary WDDL circuit is simply discarded.

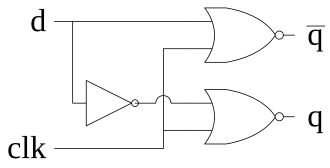
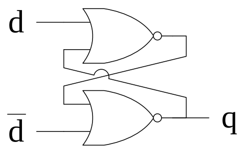


Figure 6.5: Schematic of a CMOS to WDDL conversion cell; the precharge and the evaluation phase is introduced by the clock signal.



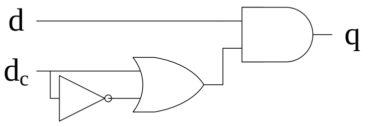


Figure 6.6: Schematics of (D)WDDL to CMOS conversion cells; left: WDDL to CMOS, the NOR latch removes the precharge phase; right: DWDDL to CMOS, the data signal from the complementary WDDL circuit d_c is simply discarded.

Flawless Exchange of Logic Functions and Duplication of the WDDL Circuit

In order to implement DWDDL correctly, the logic functions of the complementary WDDL circuit need to be exchanged, e.g. swapping an AND and an OR gate. The implementation of DWDDL on an FPGA can be performed rather easily by substituting each standard cell in a gate-level netlist with two lookup tables (LUTs). One of the LUTs implements the functionality of the substituted standard cell and the other LUT implements the complementary function, e.g. an AND cell is replaced by a LUT implementing the AND function and by a second LUT implementing the OR function.

After synthesizing, placing, and routing the netlist to the target FPGA architecture, the resulting *direct* WDDL circuit is copied. The coordinates of all LUTs of the copied *complementary* circuit are consistently moved, i.e. the relative placement and the electrical routes between all LUTs persist. Then, the LUTs are logically modified (exchanged) to implement the complementary function of their counterpart in the direct circuit. Finally, special care has to be taken when connecting the inputs and outputs of the two WDDL circuits. Timing differences or imbalances between the signals going to the two WDDL circuits would significantly reduce the resistance against PA attacks due to a temporal offset in the performed operations.

In case of an ASIC implementation the exchange of gates can not be performed that easily, because of structural differences between CMOS gates. The logic gates available in many standard cell libraries have the input and output pins of the gates located at different coordinates, i.e. a simple exchange of the logic gates most probably results in faulty contacts and hence in a non-functioning integrated circuit. Re-routing the affected wires would result in significant differences between the two WDDL circuits, which would be counter-productive. We have developed a technique to overcome this issue: most standard cell libraries contain a logic gate that implements the majority (MAJ) function. The output of a MAJ gate is 1 if at least two of the three inputs are 1, otherwise the output remains 0. The truth table of a MAJ gate shown in Table 6.4 reveals that a three-input MAJ gate can be used as programmable two-input AND/OR gate. The input c determines the functionality of the MAJ gate: if $c = 0$ the output q follows the rules of an AND function of the inputs a and b, if $c = 1$ the output q is produced according to an OR function.

Table 6.4: Using a MAJ gate as configurable AND / OR gate.

AND				OR			
a	**b**	**c**	**q**	**a**	**b**	**c**	**q**
0	0	**0**	0	0	0	**1**	0
0	1	**0**	0	0	1	**1**	1
1	0	**0**	0	1	0	**1**	1
1	1	**0**	1	1	1	**1**	1

We utilized this fact for implementing DWDDL on an ASIC: instead of using AND and OR gates we used MAJ gates and statically connected one of the three inputs (used as configuration signal) either to 1 or to 0, depending on the target function. Similar to implementing DWDDL on an FPGA, in a first step each conventional logic gate (AND, OR) in a netlist has to be replaced by two MAJ gates: one MAJ gate implements the direct functionality and the other one implements the complementary functionality. The configuration signals of all MAJ gates in the WDDL circuit were connected to the respective signal net, and the WDDL circuit was equipped with two additional configuration ports. The basic conversion steps from conventional CMOS logic to WDDL were performed using a netlist converter introduced in Section 3.3. After the conversion process, we had a fully functional netlist containing three secure zones: one implemented in CMOS logic, one in iMDPL, and one in WDDL.

The next important step was the duplication of the WDDL circuit. After we performed the netlist conversion of the secure zone we created a second instance of the direct WDDL circuit. In this step also the conversion cells between CMOS and DWDDL were integrated: the inputs to the complementary circuit were inverted, and the outputs of both WDDL circuits were combined in a way that the final output of the DWDDL circuit corresponds to the output of the direct WDDL circuit. Furthermore, the configuration ports of the complementary WDDL circuit were inverted in order to realize the exchange of the logic functions: this way the MAJ gates in the WDDL cells simply switched their functionality from AND to OR and vice versa. Note, at this point two instances of one physical WDDL circuit were present in the gate-level netlist. In the course of a conventional design phase the physical circuit would have been duplicated and bound to the second instance. This way the placement and routing of the two WDDL circuits would have been performed independently of each other, which would have resulted in different placement and routing results. In order to achieve an identical placement and routing of the two WDDL circuits, we used the partitioning feature of Cadence SoC Encounter (cf. Section 3.3.2) to physically duplicate the WDDL circuit. After defining bounding boxes for both module instances in the floorplan we defined one of the instances as master partition, the other instance was automatically defined as slave partition. In the following, a rough placement of the cells within the partition was performed in order to determine the coordinates of the partition pins. Then the partition information was saved and we obtained two partitions: (1) the WDDL partition and (2) the top level partition containing the rest of the chip. The next step was the conventional placement and routing of the WDDL partition, followed by the placement and routing of the remaining modules of the chip in the top level partition. In the end the WDDL partition blocks and the top level partition were assembled and the final chip data was created. This way we obtained two identically placed and routed WDDL circuits.

We are aware that such a regional separation of two WDDL circuits on a chip might increase the vulnerability to localized EM attacks. However, in the POWER-TRUST project we focussed on the design and the implementation of the secure-zone approach to determine the practicality and the impact of such a method on the overall performance of an embedded processor. The vulnerability to EM attacks could be reduced by utilizing a more advanced approach to implement balanced complementary wires.

6.3.5 Production of the POWER-TRUST Chip

The POWER-TRUST chip was produced in a 180nm CMOS process technology from UMC [Uni], using standard cell libraries from Faraday [Far04]. Originally, we wanted to insert memory macros for the register file and the cache memories, but despite our best efforts we were unable to flawlessly integrate the memory macros in our automated design flow within a reasonable period of time. Hence we synthesized the memory structures using conventional flip-flops. The overall area of the chip die including the I/O pads is approximately $21\,mm^2$. The floorplan of the POWER-TRUST chip is shown in Figure 6.7. The secure zone CMOS (sz_CMOS) is located in the bottom right corner, the secure zone iMDPL (sz_iMDPL) occupies the left side of the floorplan, and the two rectangularly shaped secure-zone modules implemented in WDDL are located near the center. The integer unit (IU) of the Leon3, the memory management unit (MMU), the debug support unit, the register file, and the cache memories occupy the rest of the area on the die.

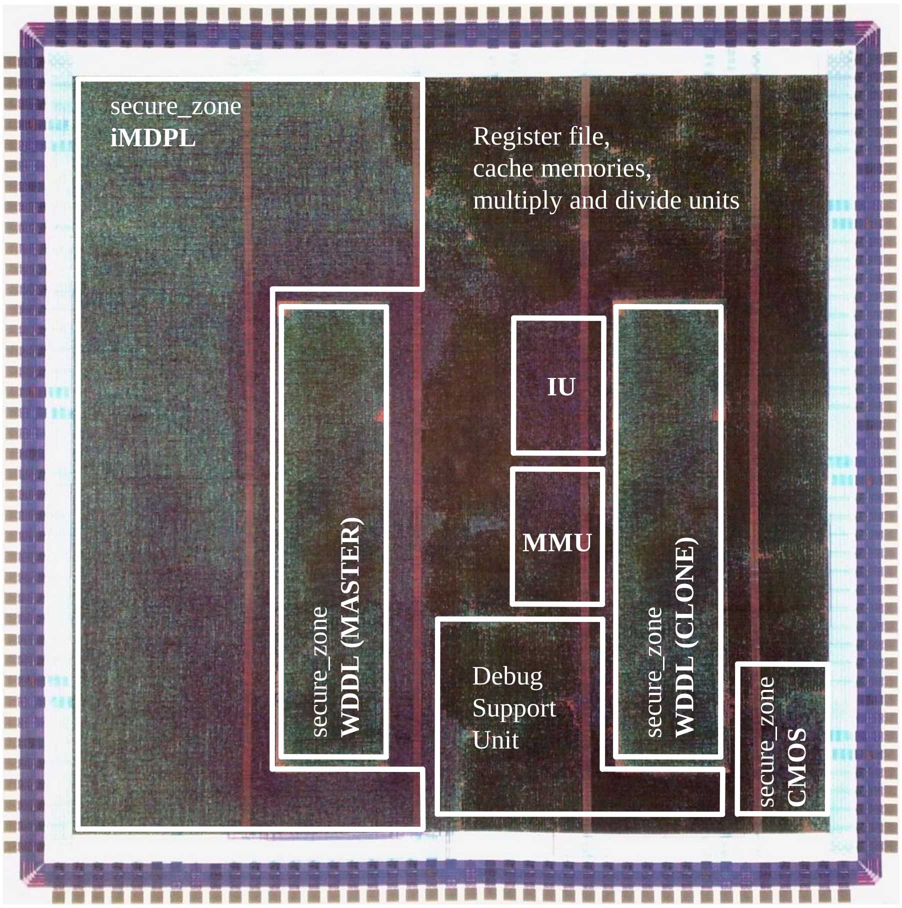


Figure 6.7: Floorplan of the POWER-TRUST chip.

Table 6.5 (upper part) shows the circuit complexity of the main modules on our prototype chip. The secure zone implemented in CMOS logic has a complexity of 19 kGEs. The complexity of an iMDPL implementation is usually increased by a factor around 20 (cf. Section 5.4.3), in our case the secure zone iMDPL has 406 kGEs, which corresponds to a factor of 21 compared to CMOS. A single WDDL secure zone has 130 kGEs (equals a factor of 7), the DWDDL implementation (260 kGEs) thus requires approximately 14 times the area of a corresponding CMOS implementation. The bare Leon3 processor excluding memories (488 kGEs), the secure zones (692 kGEs), and the conventional AES ISEs (6 kGEs) has 88 kGEs.

Based on the size of the Leon3 processor including memories and a secure zone implemented in CMOS (in total 595 kGEs) we evaluated the benefit of the secure-zone approach in terms of area overhead compared to a processor completely implemented in a secure logic style. The results are summarized in

Table 6.5 (lower part). A Leon3 processor including a secure zone implemented in iMDPL requires 982 kGEs which corresponds to an overhead of only 65%. In case of DWDDL the overhead even shrinks to 40%. These values represent a moderate increase in area in return for a possibly significantly increased resistance against PA attacks. Contrary, implementing the bare Leon3 processor (with insecure memories) in iMDPL or DWDDL would result in an area overhead of approximately 400% and 290%, respectively.

Table 6.5: The main components and their complexity of the Leon3 processor (upper part); the complexity of the Leon3 processor in case of different secure-zone configurations and the overhead caused by the logic styles (lower part).

Module	kGEs
sz_CMOS	19
sz_iMDPL	406
sz_WDDL (single instance)	130
sz_DWDDL	260
Convent. AES extensions	6
Leon3 processor	88
Memories	488

Secure-zone configuration	Total kGEs	Overhead
Leon3 + memories + sz_CMOS	595	0%
Leon3 + memories + sz_iMDPL	982	65%
Leon3 + memories + sz_DWDDL	836	40%
Leon3_iMDPL + memories	2 400	400%
Leon3_DWDDL + memories	1 700	290%

6.3.6 Identifying a Bug on the POWER-TRUST Chip

During the first functionality tests of the POWER-TRUST chip we encountered a strange behavior when operating the secure zones implemented in iMDPL and in DWDDL. In some cases the instructions performed in one of the secure zones yielded incorrect results. By means of executing several elaborated test routines and investigating the processes at the interface between the secure zones and the Leon3 IU by means of logic simulations, we figured out that the secure zones behave incorrectly in case of cache misses. If a cache miss occurs either in the instruction cache or the data cache of the Leon3 processor the secure zones should be put on *hold*, i.e. the secure zones should maintain their current register contents. Unfortunately, during the implementation of the secure zones and the logic styles we missed to check the correct functionality of this feature.

However, we discovered a workaround that enables us to fully use each of the secure zones: if we execute the same instruction a few times in a row, the cache misses do not occur during the last execution of the instructions and the secure zones yield the correct results. More important, this workaround does not influence the security of the secure zones, because at the time the hold signal is missed the data coming out of the secure zone is already masked. Furthermore, the consecutive execution of identical operations with identical data has no influence, as both secure zones are based on precharging logic styles, and hence, no data remains stored anywhere in the secure zones. The operations performed within the secure zones are protected by the logic styles anyway.

6.4 Evaluation of the Secure-Zone Approach on a Prototype Chip

We performed correlation-based PA attacks on various AES implementations on the POWER-TRUST chip in the order of their expected PA resistance. First, we evaluated a standard software AES implementation that uses only native SPARC V8 instructions and that is based on a lookup table for the S-box transformation (SWAES). The second AES implementation uses the basic AES ISEs we integrated in the Leon3 processor, which eliminate the need for table lookups (AESREF). We also implemented AES in each of the secure zones (sz_CMOS, sz_iMDPL, and sz_DWDDL). The practical results of our PA attacks were published in [TKS11]. The results are presented and discussed in the section after next.

6.4.1 POWER-TRUST Evaluation Board

In the course of the POWER-TRUST project we designed a versatile evaluation board in order to provide various application possibilities of the POWER-TRUST chip. The board was built with respect to performing power measurements of the chip in the first place. We integrated adjustable voltage regulators for both the I/O and the core power supply of the chip as well as the possibility to easily measure the chip's power consumption by inserting measurement shunts in the V_{DD} and GND lines of both power supplies. For the basic operation of the Leon3 processor we added Flash, SRAM, and SDRAM modules on the board as well as two RS232 interfaces and an Ethernet microcontroller that is operated memory mapped in the Leon3 processor. Figure 6.8 depicts the layout of the evaluation board. In the right part of the board layout it can be seen that we expended major effort on carefully routing the address and the data bus to and from the external memory modules. The board and the chip were also successfully utilized in the *System-on-Chip Architectures and Modelling* course [Ins11] at our institute.

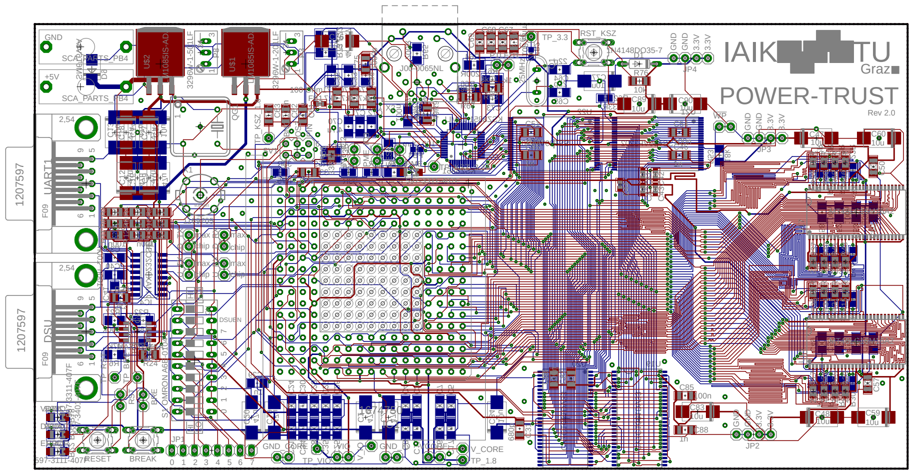


Figure 6.8: Layout of the POWER-TRUST evaluation board.

The power measurements of the POWER-TRUST chip were performed using a 2.2 Ω resistor in the V_{DD} line of the core supply, and we utilized a highly optimized measurement setup using a LeCroy WP725Zi oscilloscope. Table 6.6 summarizes the most important parameters of our power measurements.

Table 6.6: Summary of the parameters for the measurements of the POWER-TRUST chip.

Oscilloscope	LeCroy WP725Zi
Probe	LeCroy D320
Probe Coupling	1 $M\Omega$ AC
Clock frequency	11.0592 MHz
Measurement shunt	2.2 Ω resistor in V_{DD} line
Sampling rate	5 GS/s

6.4.2 PA Attack Results

For the evaluation of the PA attacks on the different AES implementations executed on the POWER-TRUST chip we used the same attack success metrics as in case of the GRANDESCA chip: we defined a PA attack as succeeded if at least 9 of the 16 key bytes could be distinctly revealed by an attack. We also used the *rule of thumb* [MOP07] (Chapter 6.4.1) to calculate the number of required power traces based on the correlation value of the correct key hypothesis.

In a first step we performed a correlation-based PA attack on the software AES (SWAES) implementation. The attack was performed using the HW of the S-box output to generate the power hypothesis. We performed 10 000 measurements of SWAES and the ninth-highest correlation value we obtained is 0.315, which corresponds to a number of required traces $n = 263$. In case of the SWAES implementation we omitted attacks based on the HD of two consecutive intermediate values due to the involvement of the MixColumns operation. The HD attacks would hence have resulted in expensive attacks based on 4 key bytes.

The AESREF implementation we evaluated was based on basic AES ISEs. It turns out that the avoidance of S-box lookup tables already has a significant effect on the PA resistance of an implementation. We first used the HW of the S-box output to generate the power hypotheses, and the ninth-highest correlation value we obtained is 0.0319, which corresponds to a number of required power traces of 27 200. In case of the AESREF implementation the PA attacks based on the HD power model resulted in significantly higher correlation values around 0.2, which corresponds to 700 required power traces. The HD leakage most probably originates from storing the unmasked values in the register file of the Leon3 processor.

We received very interesting results from the sz_CMOS, on which we performed 2 000 000 measurements. It turned out that the utilization of a secure zone already has a significant effect on the PA resistance, even if the secure zone is not implemented in a secure logic style. Due to the strict masking of the values outside of the secure zone, the HD leakage is almost completely defeated, except for two byte transitions which require 30 000 and 70 000 traces for disclosure. The PA attacks based on the HW resulted in a correlation coefficient of 0.0146, which corresponds to 130 000 required traces.

The PA attacks on the sz_iMDPL showed a further increase in the PA resistance. We performed 5 000 000 measurements and obtained a ninth-highest correlation value of 0.0103 (for HW based hypotheses), which corresponds to 260 000 traces. Similar to the results obtained from the GRANDESCA chip (cf. Section 5.5.2), we were not able to perform any successful attack using up to 5 000 000 traces using HD based hypotheses.

On the sz_DWDDL we performed 10 000 000 measurements. It turned out that the lower 16 bits of each of the four 32-bit words processed could not be clearly distinguished in a HW based attack. The detailed investigation of this circumstance is a topic of our current research work. We were able to reveal 8 of the 16 key bytes. In this case, the eighth-highest correlation value we obtained was 0.0064, which corresponds to 675 000 required traces. By means of HD based hypotheses we were able to reveal all 16 key bytes in the sz_DWDDL at a high price: between 5 200 000 and 8 000 000 power traces were required to distinctly reveal the correlation traces of the correct key hypothesis.

6.4.3 Summary of the Performed PA Attacks

The correlation results of all HW based PA attacks are summarized in Table 6.7, the results of the HD based PA attacks are summarized in Table 6.8.

The iMDPL and the DWDDL style are precharging logic styles, i.e. all internal buses are precharged in every clock cycle. Hence, the HD of two consecutively processed byte values is not a suitable power model for sz_iMDPL and sz_DWDDL. In case of sz_CMOS, no intermediate value remains stored in the secure zone: after an instruction was executed, the internal buses of the sz_CMOS also return to zero. Hence, the very low HD leakage in case of the three secure zones makes sense, except for two byte transitions 11 → 15 and 16 → 4 in case of sz_CMOS, which keeps unclear at this stage but is marked for future work. Another point that remains unclear is the regularity in the HW correlation results of the secure zones: in case of sz_CMOS and sz_DWDDL every second byte-pair (3,4,7,8,11,12,15,16) yielded a significantly higher correlation value than the others (1,2,5,6,9,10,13,14). A similarly strange behavior can be observed in case of sz_iMDPL where every second byte (2,4,6,8,10,12,14,16) yielded a significantly higher correlation value than the other bytes (1,3,5,7,9,11,13,15).

Table 6.7: Summary of the correlation-based PA attacks performed on different AES implementations on the POWER-TRUST chip; Hamming weight power model; correlation values marked with an asterisk indicate unsuccessful attacks; correlation values in bold represent the ninth-highest value, in case of sz_DWDDL the eighth-highest value.

Target	**SWAES**	**AESREF**	**sz_CMOS**	**sz_iMPDL**	**sz_DWDDL**
Byte 1	0.227	0.0164*	0.0111	0.0066	0.0028*
Byte 2	0.298	0.162	<0.003*	0.0177	0.0025*
Byte 3	0.574	0.0244	0.0264	**0.0103**	0.0121
Byte 4	0.135	0.177	0.0177	0.0289	0.0243
Byte 5	0.230	0.0334	**0.0146**	0.0067	0.0024*
Byte 6	0.697	0.0162*	0.0074	0.0195	0.0018*
Byte 7	0.672	0.0878	0.0261	0.0080	**0.0064**
Byte 8	0.178	0.0294	0.0213	0.0289	0.0120
Byte 9	**0.315**	0.0222	0.0085	0.0059	0.0026*
Byte 10	0.564	0.0355	0.0067	0.0166	0.0018*
Byte 11	0.511	0.0240	0.0228	0.0078	0.0091
Byte 12	0.556	0.1296	0.0192	0.0268	0.0114
Byte 13	0.409	0.145	0.0050	0.0062	0.0028*
Byte 14	0.294	0.0201	0.0045*	0.0181	0.0019*
Byte 15	0.452	0.168	0.0237	0.0084	0.0155
Byte 16	0.113	**0.0319**	0.0240	0.0279	0.0098

The AES ISEs in the FU of the AESREF implementation are identical to the ISEs in the FU of the sz_CMOS, and hence, no intermediate values remain stored in the AESREF module neither. As a consequence, the relatively high HD leakage in case of AESREF is most probably caused by storing the unmasked results in the register file of the processor and by overwriting the previously computed results, which were also unmasked. We assume that the lower HW leakage of the AESREF implementation (compared to HD AESREF) mostly originates from the relatively small hardware modules realizing the basic AES ISEs in the AESREF FU.

Table 6.8: Summary of the correlation-based PA attacks performed on different AES implementations on the POWER-TRUST chip; Hamming distance power model; correlation values marked with an asterisk indicate unsuccessful attacks.

Target	**AESREF**	**sz_CMOS**	**sz_iMPDL**	**sz_DWDDL**
Byte 1 → 5	0.184	<0.003*	<0.002*	0.0019
Byte 3 → 7	0.211	<0.003*	<0.002*	0.0019
Byte 5 → 9	0.184	<0.003*	<0.002*	0.0020
Byte 6 → 10	0.200	<0.003*	<0.002*	0.0023
Byte 7 → 11	0.212	<0.003*	<0.002*	0.0021
Byte 8 → 12	0.221	<0.003*	<0.002*	0.0023
Byte 9 → 13	0.184	<0.003*	<0.002*	0.0017
Byte 10 → 14	0.201	<0.003*	<0.002*	0.0019
Byte 11 → 15	0.214	0.0314	<0.002*	0.0021
Byte 12 → 16	0.221	<0.003*	<0.002*	0.0021
Byte 14 → 2	0.201	<0.003*	<0.002*	0.0017
Byte 16 → 4	0.222	0.0239	<0.002*	0.0021

Our evaluations on the POWER-TRUST chip showed that we cannot resolve every open question solely by means of PA attacks on the prototype chip. Similar to iMDPL implemented on the GRANDESCA chip presented in the previous chapter, we need to further investigate the POWER-TRUST chip by means of simulations and PA attacks on toggle-count traces in order to pinpoint the remaining issues in our implementation. The results of the prototype chip illustrated that the secure-zone concept enables an economic implementation of secure logic styles in modern processor platforms in terms of the required area overhead. Furthermore, the technique of utilizing configurable logic gates to implement a secure logic style seems promising. More complex gates could be utilized, for example, to realize advanced masking techniques working with more than one mask value.

7 Conclusions and Future Work

In this thesis, we presented methods and techniques to design, to implement, and to evaluate PA-resistant logic styles. Further, we presented two different approaches to implement hardware modules in a secure logic style and provided practical results based on two prototype chips for both approaches.

Before we presented details on our evaluation techniques and the implemented prototype chips, we gave an introduction to the power-consumption behavior and the vulnerability to PA attacks of modern digital integrated circuits based on CMOS logic in Chapter 2. We further discussed countermeasure approaches against PA attacks that emerged during the last decades. In the last part of this chapter we presented practical examples of PA and EMA attacks on different hardware devices. The examples demonstrated the vulnerability of modern sensor nodes as well as the impact of an imperfectly implemented countermeasure on the security of a cryptographic device. Further, we presented a powerful technique to investigate digital circuits in every detail based on EM measurements combined with surface scanning.

In Chapter 3 we focused on cell-level countermeasures against PA attacks and presented a brief overview of logic styles that were proposed during the last decades. We pointed out the two categories of full-custom based and standard-cell based logic styles. Further, we discussed the implementation of PA-resistant logic styles with a semi-custom VLSI design flow and pointed out the main steps necessary to realize such logic styles on an ASIC.

Chapter 4 discussed various methods to investigate the effectiveness of PA-resistant logic styles on different levels. We elaborated on the theoretical investigation, the practical verification by means of simulations, as well as the verification by means of implementing hardware circuits on FPGAs and ASICs. Regarding the latter, we also discussed the necessity of a fast and reliable measurement and evaluation environment and presented techniques we developed and optimized during our work. In the last part of this chapter we discussed exemplary applications of the presented evaluation methods in order to point out the effectiveness and the significance of these approaches.

In Chapter 5 we started with a brief summary of the effect of early propagation on PA-resistant logic styles and presented evaluation results of an MDPL prototype chip. Based on these findings we presented an improved version of the MDPL style, called iMDPL, and performed a theoretical investigation of the improvements. Then, we discussed the architecture of our iMDPL prototype chip which contains a microcontroller core with an AES coprocessor as well as a standalone AES core; each of the cores was implemented in both CMOS logic and in iMDPL. Following, we presented a comprehensive security evaluation of the iMDPL style based on PA attacks on measurements and on back-annotated logic-level simulations. It turned out that iMDPL significantly reduces the effect of early propagation, which comes at the price of a significantly increased complexity. Compared to CMOS logic, an iMDPL implementation requires approximately 18-20 times the area and the maximum clock frequency is reduced by 80%. Further investigations showed that iMDPL suffers from imbalanced complementary wires, which results in a remaining data leakage as well as in the detectability of the mask value in an iMDPL circuit. We showed that information about the active mask value can directly be extracted from the power traces. This way, PA attacks on an iMDPL implementation can be simplified to a certain degree.

In Chapter 6 we introduced the secure zone approach where AES ISEs are combined with secure logic styles. All operations on critical data are performed within the secure zone module, i.e. all transformations of the data are protected by the secure logic style. Outside of the secure zone the data is just forwarded/stored and protected by architectural masking. We presented our implementation of this approach in a Leon3 processor combined with two different secure logic styles: iMDPL and DWDDL. Subsequently, we evaluated the area overhead compared to a secure zone implemented in CMOS logic and to a processor implemented entirely in one of the secure logic styles. It turned out that the overall area overhead is only 65% in case of a secure zone implemented in iMDPL and 40% in case of DWDDL, which represents quite moderate values. The security evaluation of the prototype chip showed that the secure zone approach is able to significantly increase the resistance against PA attacks. Unfortunately, at this stage there are several questions unanswered which are marked for future work. During our evaluations it turned out that we are not able to fully resolve the open questions solely by means of measurements. It further turned out that we cannot perform logic simulations of the whole Leon3 processor within a reasonable time. Hence, we plan to perform simpler simulations of individual submodules in order to be able to investigate the processes within the Leon3 processor and the secure zones in more detail.

In general it turned out that many countermeasure proposals are not perfect and suffer from various shortcomings. Most countermeasures are not able to completely prevent but to significantly complicate PA and EMA attacks. It is likely that the effectiveness of many countermeasures, especially of some proposals on the cell level, is strictly narrowed due to physical limitations that arise from the fabrication processes.

It also turned out that a fair comparison of various countermeasure proposals is almost impossible to perform as the investigation of most countermeasures proposed so far was based on significantly different hardware structures and on different evaluation metrics. The joint implementation of a few prototype chips that contain identical hardware structures and that are fabricated using different process technologies, e.g. 180 nm and 40 nm, would represent a great opportunity to perform a thorough comparison of different countermeasure techniques. Further, this would also resolve the uncertainties about the impact of steadily shrinking process technologies on the vulnerability of digital circuits to PA attacks.

Bibliography

[Ach05] Stefan Achleitner. A Strong Authentication Module Resistant against Side-Channel Attacks. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, October 2005.

[Alf96] Peter Alfke. Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators. Available online at http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf, July 1996. Application Note.

[AMM$^+$06] Manfred Aigner, Stefan Mangard, Francesco Menichelli, Renato Menicocci, Mauro Olivieri, Thomas Popp, Giuseppe Scotti, and Alessandro Trifiletti. Side channel analysis resistant design flow. In *International Symposium on Circuits and Systems (ISCAS 2006), 21-24 May 2006, Island of Kos, Greece*, pages 2909–2912, 2006.

[BGL$^+$11] Marco Bucci, Luca Giancane, Raimondo Luzzi, Giuseppe Scotti, and Alessandro Trifiletti. Delay-Based Dual-Rail Precharge Logic. *Very Large Scale Integration (VLSI) Systems, IEEE*, 19:1147–1153, July 2011.

[BGLR08] L. Batina, B. Gierlichs, and K. Lemke-Rust. Comparative Evaluation of Rank Correlation Based DPA on an AES Prototype Chip. In *Proceedings of the 11th international conference on Information Security, 2008, Taipei, Taiwan*, 2008.

[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.

[Cad11] Cadence Design Systems, Inc. The Cadence Design Systems Website. http://www.cadence.com/, 2011. San Jose, California, United States.

[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.

[Far04] Faraday Technology Corporation. Faraday FSA0A_C 0.18 μm ASIC Standard Cell Library, 2004. Details available online at http://www.faraday-tech.com.

[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.

[Gai10] Gaisler Research. GRLIB IP Library User's Manual. Available online at http://www.gaisler.com/products/grlib/grlib.pdf, October 2010. Version 1.1.0 B4100.

[GRA07] GRANDESCA project. Generating RANDom values for Encryption in the presence of Side Channel and other Attacks: Project Webpage. http://www.iaik.tugraz.at/grandesca, 2007.

[HBGL11] Norman Hendrich, Johannes Bitterling, Manfred Grove, and Lars Larsson. Hamburg Design System - interactive simulation framework. =http://tams-www.informatik.uni-hamburg.de/applets/hades, 2011.

[HHL$^+$05] Alireza Hodjat, David D. Hwang, Bo-Cheng Lai, Kris Tiri, and Ingrid Verbauwhede. A 3.84 Gbits/s AES crypto coprocessor with modes of operation in a 0.18-um CMOS Technology. In John Lach, Gang Qu, and Yehea I. Ismail, editors, *15th ACM Great Lakes Symposium on VLSI 2005, Chicago, Illinois, USA, April 17-19, 2005, Proceedings*, pages 60–63. ACM, ACM Press, April 2005. Available online at http://portal.acm.org/ft_gateway.cfm?id=1057677&type=pdf&coll=GUIDE&dl=GUIDE&CFID=50585284&CFTOKEN=38947284.

[HV04] Alireza Hodjat and Ingrid Verbauwhede. Minimum Area Cost for a 30 to 70 Gbits/s AES Processor. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2004), Emerging Trends in VLSI Systems Design, 19-20 February, 2004, Lafayette, LA, USA*, pages 83–88. IEEE Computer Society, 2004.

[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.

[Ins11] Institute for Applied Information Processing and Communications. System-on-Chip Architectures and Modelling: Master Course. http://www.iaik.tugraz.at/content/teaching/master_courses/system_on_chip/, August 2011.

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

[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.

[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.

[KKT06] 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.

[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*, Proceedings of 25th ACSAC, pages 43–50, 2009.

[KPP$^+$06a] Sandeep S. Kumar, Christof Paar, Jan Pelzl, Gerd Pfeiffer, Andy Rupp, and Manfred Schimmler. How to Break DES for 8,980. Workshop on Special-purpose Hardware for Attacking Cryptographic Systems - SHARCS 2006, April 3-4, Cologne, Germany, 2006.

[KPP$^+$06b] Sandeep S. Kumar, Christof Paar, Jan Pelzl, Gerd Pfeiffer, and Manfred Schimmler. Breaking Ciphers with COPACOBANA – A Cost-Optimized Parallel Code Breaker. 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 101–118. Springer, 2006.

[KS11] Mario Kirschbaum and Jörn-Marc Schmidt. Learning from Electromagnetic Emanations - A Case Study for iMDPL. In *Second International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2011), 24-25 February 2011, Darmstadt, Germany*, Workshop Proceedings COSADE 2011, pages 50–55, 2011.

[LeCa] LeCroy. LeCroy LC564 / LC584 / LC684 Series Oscilloscope. http://www.lecroy.com/tm/products/Scopes/LCSeries/LC684DXL/default.asp.

[LeCb] LeCroy. LeCroy WavePro 725Zi Series Oscilloscope. http://www.lecroy.com/Oscilloscope/OscilloscopeModel.aspx?modelid=1536.

[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.

[Man03a] Stefan Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2003.

[Man03b] Stefan Mangard. Exploiting Radiated Emissions - EM Attacks on Cryptographic ICs. In Timm Ostermann and Christoph Lackner, editors, *Proceedings of Austrochip 2003, October 3, 2003, Linz, Austria*, pages 13–16, October 2003. ISBN 3-200-00021-X.

[MBKP11] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the Vulnerability of FPGA Bitstream Encryption against

Power Analysis Attacks – Extracting Keys from Xilinx Virtex-II FPGAs. Cryptology ePrint Archive (http://eprint.iacr.org/), Report 20011/390, 2011.

[MGPV09] 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*, pages 191–195. IEEE, 2009. ISBN 978-1-4244-5279-8.

[MKEP] Amir Moradi, Mario Kirschbaum, Thomas Eisenbarth, and Christof Paar. Masked Dual-Rail Precharge Logic Encounters State-of-the-Art Power Analysis Methods. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems.*

[MKP11] Amir Moradi, Markus Kasper, and Christof Paar. On the Portability of Side-Channel Attacks – An Analysis of the Xilinx Virtex 4 and Virtex 5 Bitstream Encryption Mechanism. Cryptology ePrint Archive (http://eprint.iacr.org/), Report 20011/391, 2011.

[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.

[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.

[Nat99] National Institute of Standards and Technology (NIST). FIPS-46-3: Data Encryption Standard, October 1999. Available online at http://www.itl.nist.gov/fipspubs/.

[NXP11] NXP Laboratories UK Ltd. JN5148 Wireless Microcontroller Modules. http://www.jennic.com/products/modules/jn5148_modules, August 2011.

[OBSC10] Dag Arne Osvik, Joppe W. Bos, Deian Stefan, and David Canright. Fast Software AES Encryption . In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption, 17th International Workshop, FSE*

2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers, volume 6147 of *Lecture Notes in Computer Science (LNCS)*, pages 75–93. Springer, 2010.

[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.

[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.

[POW10] POWER-TRUST project. Low POWer & Energy Relevant techniques Targetting Robust Universal Security in deep sub-micron Technologies: Project Webpage. http://www.iaik.tugraz.at/powertrust, 2010.

[Rab96] Jan M. Rabaey. *Digital Integrated Circuits – A Design Perspective*. Electronics and VLSI Series. Prentice Hall, 1st edition, 1996. ISBN 0-13-178609-1.

[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.

[SCA05] SCARD project. Side Channel Analysis Resistant Design: Project Webpage. http://www.iaik.tugraz.at/scard, 2005.

[SPK+10] Jörn-Marc Schmidt, Thomas Plos, Mario Kirschbaum, Michael Hutter, Marcel Medwed, and Christoph Herbst. Side-Channel Leakage Across Borders. In Dieter Gollmann and Jean-Louis Lanet, editors, *Smart Card Research and Advanced Applications 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010, April 13-16, 2010, Passau, Germany, Proceedings*, Lecture Notes in Computer Science, pages 36–48. Springer, April 2010.

[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.

[SS08a] 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.

[SS08b] Daisuke Suzuki and Minoru Saeki. An Analysis of Leakage Factors for Dual-Rail Pre-Charge Logic Style. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E91-A(1):184–192, 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.

[SSI07] 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.

[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.

[SV02] Patrick Schaumont and Ingrid Verbauwhede. Domain Specific Tools and Methods for Application in Security Processor Design . *Design Automation for Embedded Systems*, 7(4):365–383, November 2002.

[Syn] Synopsys. The Synopsys Website. http://www.synopsys.com/.

[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.

[TG06] Stefan Tillich and Johann Großschädl. Instruction Set Extensions for Efficient AES Implementation on 32-bit Processors. 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 270–284. Springer, 2006.

[TG07] Stefan Tillich and Johann Großschädl. Power-Analysis Resistant AES Implementation with Instruction Set Extensions. 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 303–319. Springer, September 2007.

[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.

[TKS10] Stefan Tillich, Mario Kirschbaum, and Alexander Szekely. SCA-Resistant Embedded Processors – The Next Generation. In Carrie Gates, Michael Franz, and John P. McDermott, editors, *26th Annual Computer Security Applications Conference (ACSAC 2010), 6-10 December 2010, Austin, Texas, USA*, pages 211–220. ACM Press, 2010.

[TKS11] Stefan Tillich, Mario Kirschbaum, and Alexander Szekely. Implementation and Evaluation of an SCA-Resistant Embedded Processor. In *Proceedings of the Tenth Smart Card Research and Advanced Application Conference, CARDIS 2011, September 15-16, 2011, Leuven, Belgium, Proceedings*, 2011.

[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 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.

[TV04b] Kris Tiri and Ingrid Verbauwhede. Charge Recycling Sense Amplifier Based Logic: Securing Low Power Security IC's against DPA. In *30th European Solid-State Circuits Conference - ESSCIRC 2004, Leuven, Belgium, September 21-23, 2004, Proceedings*, pages 179–182. IEEE, September 2004.

[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.

[TV05a] Kris Tiri and Ingrid Verbauwhede. A VLSI Design Flow for Secure Side-Channel Attack Resistant ICs. In *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, pages 58–63. IEEE Computer Society, 2005. ISBN 0-7695-2288-2.

[TV05b] 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.

[TV06] Kris Tiri and Ingrid Verbauwhede. A Digital Design Flow for Secure Integrated Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(7):1197–1208, July 2006. ISSN 0278-0070.

[Uni] United Microelectronics Corporation. The United Microelectronics Corporation Website. http://www.umc.com/.

[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.

[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.

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)