Physical Unclonable Functions

Christoph Boehm, Maximilian Hofer

Physical Unclonable Functions

Dipl.-Ing. Christoph Boehm

Dipl.-Ing. Maximilian Hofer

Submitted as thesis to attain the academic degree “Dr. techn.”
at the

Graz University of Technology

Institute of Electronics

Supervisor: .....
Univ.-Prof. Dipl.-Ing. Dr. techn. Wolfgang Pribyl

Graz, May 2013

Erklärung zur gemeinsamen Dissertation

Die folgende Dissertation ist das Ergebnis der dreijährigen Forschungstätigkeit zweier Dissertanten auf dem Gebiet der Physical Unclonable Functions. Die Arbeit wurde im Rahmen des FIT-IT Projekts PUCKMAES, einer Kooperation des Instituts für Elektronik der Technischen Universität Graz und der Infineon Technologies Austria AG, ausgeführt. Aufgrund der bisher fehlenden Expertise der beteiligten Partner auf dem Gebiet der Physical Unclonable Functions waren, mit dem Ziel etwaige Synergieeffekte ausnutzen zu können, von Anfang an zwei Dissertanten für das Projekt vorgesehen.

Die Arbeit im Team hat sich im Laufe des Projekts als sehr produktiv herausgestellt. So sind in deren Folge nicht nur neun nationale und internationale Veröffentlichungen und zwei Erfindungsmeldungen, sondern auch zwei Testchips entstanden, die als Grundlage für zukünftige Produkte von Infineon Technologies Austria AG dienen.

Auch wenn die wissenschaftliche Arbeit in zwei separaten Dissertationen dargestellt werden könnte, haben wir uns für eine gemeinsame Version entschieden. Durch die Zusammenführung der beiden Arbeiten konnte der gesamte Themenkomplex in einem einzigen Dokument abgebildet und dem Leser so ein roter Faden erhalten werden.

Um die Beurteilung der Arbeit möglichst übersichtlich zu gestalten, wurden die einzelnen Kapitel mit dem jeweiligen Autor gekennzeichnet.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die mit meinem Namen gekennzeichneten Kapitel vollständig verfasst habe. Die mit Maximilian Hofer gekennzeichneten Kapitel wurden von diesem zu 100% selbstständig verfasst. Andere als die angegebenen Quellen/Hilfsmittel wurden nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen habe ich als solche kentlich gemacht.

I declare that I have authored the parts of this thesis independently that are marked with my name. All of those parts that are marked Maximilian Hofer were authored by him. I declare that I have not used other than the declared sources / resources, and that I have explicitly marked all material which is quoted either literally or by content from the used sources.

.....	.....
Date	Christoph Boehm

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die mit meinem Namen gekennzeichneten Kapitel vollständig verfasst habe. Die mit Christoph Böhm gekennzeichneten Kapitel wurden von diesem zu 100% selbstständig verfasst. Andere als die angegebenen Quellen/Hilfsmittel wurden nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen habe ich als solche kentlich gemacht.

I declare that I have authored the parts of this thesis independently that are marked with my name. All of those parts that are marked Christoph Böhm were authored by him. I declare that I have not used other than the declared sources / resources, and that I have explicitly marked all material which is quoted either literally or by content from the used sources.

.....	.....
Date	Maximilian Hofer

Abstract

Physical unclonable functions (PUFs) are a quite new approach to secure number generation and storage. PUFs use fabrication variability to generate device specific numbers. Such a PUF generated number can be seen as the fingerprint of a device. The PUF output can be used in different ways. Today, PUFs are used for identification, authentication, and cryptographic key generation purposes. PUFs are mainly built out of microelectronic components. Those PUFs are also called silicon PUFs. Even if already extensive research has been done on PUFs, still a number of problems remain. The most dominate problem is the high error rate of the PUFs' output when it comes to environmental changes like temperature shifts.

After a general introduction, this thesis covers the different steps that can be undertaken to reduce the high error rates. The first part is on design and simulation issues related to PUFs. It covers the different mismatch sources that are available in microelectronic circuits and their usability in PUF circuits. Another focus is put on the adaptation of transistor mismatch modeling to be able to estimate future PUF error rates already during Montecarlo mismatch simulation. Furthermore, different pre-processing steps are suggested that help to avoid the usage of error-prone PUF cells and thus provide a powerful way to reduce the error rates during PUF usage. The second part introduces three PUF test chips including implementation details and measurement results that focus on different design issues: reduction of noise induced errors, area reduction, pre-selection of stable PUF cells. Additionally, a SRAM PUF microcontroller implementation is presented that includes error correction capabilities and provides very small error rates. The small complexity of the implementation and the good performance make the microcontroller PUF a very attractive implementation.

Kurzfassung

Physical Unclonable Functions (PUFs) sind ein eher neuer Ansatz zur Generierung und Speicherung von sicherheitsrelevanten Daten. Diese Funktionen nutzen Schwankungen des Herstellungsprozesses aus, um elementspezifische Daten zu generieren. Der PUF-Ausgang kann auf verschiedene Art und Weise genutzt werden: Identifizierung, Authentifizierung und Schlüsselgenerierung für kryptographische Anwendungen. Die meisten PUFs werden aus mikroelektronischen Komponenten hergestellt. Auch wenn in den letzten Jahren bereits viel im Bereich der Physical Unclonable Functions geforscht und publiziert wurde, gibt es bis heute offene Probleme. Eines davon ist die hohe Fehlerrate, die vornämlich bei Temperaturänderungen auftritt.

Nach einer allgemeinen Einführung, beschäftigt sich die vorliegende Arbeit hauptsächlich mit den verschiedenen Schritten, die dazu nötig sind, die Fehlerrate so stark wie möglich zu reduzieren. Der erste Teil beschäftigt sich mit den verschiedenen Design- und Simulationsproblemen, die beim Entwickeln von PUFs auftreten können. Ein Fokus liegt auf der Anpassung der Transistor Mismatchmodelle, die für die Montecarlosimulation verwendet werden. Die Adaptierung macht es möglich, schon während der Schaltungssimulation Aussagen über künftige Fehlerraten zu treffen. Des Weiteren werden verschiedene Ansätze vorgestellt, die helfen sollen, die Verwendung von fehleranfälligen PUF-Zellen zu vermeiden. Der zweite Teil ist den Testchips gewidmet. Es werden drei Implementierung vorgestellt, deren Designfokus auf verschiedenen Gebieten liegt: Reduzierung der durch Rauschen verursachten Fehler, Flächenreduzierung und Aussortierung von fehleranfälligen Zellen. Darüber hinaus wird ein so genannter SRAM PUF präsentiert, der auf einen herkömmlichen Mikrocontroller implementiert wurde. Die einfache Implementierung und die gute Performance machen diesen Ansatz besonders attraktiv.

Acknowledgement

In the following I want to thank all the people who were involved in any kind in the work. Especially, I want to thank Maximilian Hofer, Professor Pribyl and the team of IFE, Holger Bock, Raimondo Luzzi and Marco Bucci of Infineon Technologies Austria and of course my family, my wife Nicole and my son Vincent.

CHRISTOPH BOEHM
GRAZ, MAY 2013

Acknowledgement

In the following I want to thank all the people who were involved in any kind in the work. Especially, I want to thank Christoph Böhm, Professor Pribyl and the team of IFE, Holger Bock, Raimondo Luzzi and Marco Bucci of Infineon Technologies Austria and of course my family, my wife Karin and my sons Peter and Anton.

MAXIMILIAN HOFER
GRAZ, MAY 2013

Contents

List of Figures

List of Tables

List of Abbreviations

6T-SRAM	6 Transistor SRAM
ADC	Analog To Digital Converter
AES	Advanced Encryption Standard
APSK	Amplitude And Phase-Message
ARQ	Automatic Repeat Requests
BCH	Bose-Chaudhuri-Hocquenghem
BER	Bit Error Rate
BSC	Binary Symmetric Channel
BSIM	Berkeley Short-Channel IGFET Model
BTI	Bias Temperature Instability
BL	Bit Line
CAM	Conditional Access Module
CD	Compact Disc
CDF	Cumulative Distribution Function
CI	Common Interface
CI	Confidence Interval
CMOS	Complementary Metal–Oxide–Semiconductor
CORR	Correlation
CSA	Common-Scrambling-Algorithm
DES	Data Encryption Standard
DIBL	Drain Induced Barrier Lowering
DNA	Deoxyribonucleic Acid
DSP	Digital Signal Processor
DVB	Digital Video Broadcasting
DVD	Digital Versatile Disc
E.G.	Exempli Gratia
ECC	Error-Correcting Code
ECM	Entitlement Control Message
EN	European Norm
EPC	Electronic Product Code
ERP	Electronic Road Pricing
FAR	False Alarm Rate OR False Acceptance Rate
FDR	False Detection Rate
FEC	forward error correction
FPGA	Field-Programmable Gate Array
FRR	False Rejection Rate
H	Hydrogen
H2	Hydrogen Molecule
HCI	Hot Carrier Injection
HD	Hamming Distance
HD	Helper Data
HDTV	High Definition Television

HeNe	Helium–Neon Laser
HF	High Frequency
HfO2	Hafnium Dioxide
IAP	In Application Programming
IC	Integrated Circuit
ID	Identification
KT1	Temperature Coefficient for Threshold Voltage (BSIM)
KT1L	Channel length dependence of KT1 (BSIM)
LER	Line Edge Roughness
LF	Low Frequency
MC	Micro-Controller
MIM	Metal-Insulator-Metal
MIT	Massachusetts Institute of Technology
MOS	Metal–Oxide–Semiconductor
MOSFET	Metal–Oxide–Semiconductor Field-Effect Transistor
MPEG	Moving Picture Experts Group
MTF	Median Time To Failure
NBTI	Negative Bias Temperature Instability
NEA	Number of Enroll Attempts
NFC	Near Field Communication
NFR	Number of False Rejections
NMOS	N-Channel Metal–Oxide–Semiconductor
NMOST	NMOS Transistor
NUF	Not-Useful
NVM	Non-Volatile Memory
OTA	Operational Transconductance Amplifier
PBTI	Pegative Bias Temperature Instability
PC	Personal Computer
PDF	Probability Density Function
PGG	Polygate Granularity
PLD	Programmable Logic Device
PMOS	P-Channel Metal–Oxide–Semiconductor
PMOST	PMOS Transistor
PSK	Phase-Shift Keying
PUF	Physical Unclonable Function
QPSK	Quadrature Phase-Shift Keying
R	Redundancy
R_{xx}	Auto-Correlation Coefficient
RAM	Random-Access Memory
RDD	Random Discrete Doping
RFID	Radio-Frequency Identification
RMS	Root Mean Square
RO	Ring Oszillator
ROM	Read-Only Memory
RSA	Ron Rivest, Adi Shamir and Leonard Adleman
RTN	Random Telegraph Noise
SA	Sense-Amplifier
SCM	Supply-Chain-Management
Si	Silicon

Si-H	Silicon-Hydrogen
SiO2	Siliciumdioxid
SOI	Silicon On Isolator
SRAM	Static Random-Access Memory
TDDB	Time Dependent Dielectric Breakdown
UF	Useful
UHF	Ultra-High Frequency
USB	Universal Serial Bus
UTE	Mobility temperature Exponent (BSIM)
UWB	Ultra-Wide Band
WL	Word Line

Part I.

Theories

1. Introduction

by Christoph Boehm

1.1. Preface

During the last 10 years physical unclonable functions (PUFs) have became a fixed part of security hardware. They are meant to provide a cheap and secure alternative to random number storage on devices. Especially in microelectronic circuits different approaches to PUFs were developed which performance with respect to size, error rate, and power dissipation now reach acceptable levels. Nevertheless, there is still a need for further development in the different fields related to PUFs. The following disquisition concentrates on chip-level PUF cell design including the issues a designer has to cope with during the development. This work provides theoretical and practical analysis of the design and testing of different types of silicon PUFs. Besides, an overview on physical unclonable functions and their applications will be given.

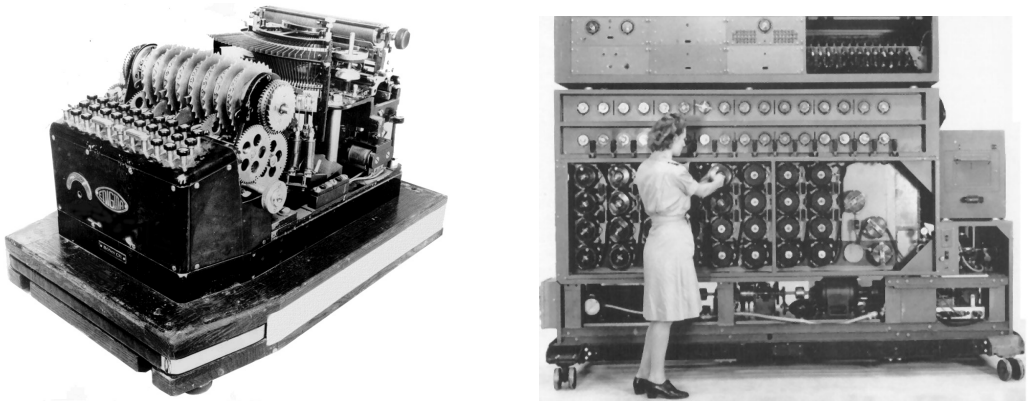
This work is divided into three major parts. The first part deals with the theoretical background. The second part covers issues related to the development and design of PUFs in integrated circuits (i.e. silicon PUFs). Part three is on four different realizations implying the measurement results of silicon PUFs. The work comprises 16 chapters. Chapter 1 provides the state-of-the-art in PUF design. It includes the motivation, an overview over different PUF approaches, a detailed analysis of an exemplary PUF circuit (SRAM PUF), a list of PUF-related patents, and finally an introduction into three PUF related topics: RFID, cryptography, and biometrics. Chapter 2 shows use cases of PUFs to give a deeper understanding of the commercial ideas behind them. Chapter 3 concentrates on the basic applications of PUFs: Identification, key generation, and authentication. Since no common procedures have been established in literature so far, Chapter 4 provides PUF cell testing and specification. Due to the high bit error rates that occur in PUF circuits, error correction is an important topic. Since common error correction codes get very demanding for such high error rates, extra effort has to be spend in order to find accurate solutions. Chapter 5 is dedicated to error correction. Chapter 6 gives the fundamentals of PUF design by analyzing the mismatch properties of microelectronic components. To be able to estimate the future error rate of PUF circuits it is very important to have valid Montecarlo mismatch models. Unfortunately, the temperature behavior of MOS transistors is not modeled sufficiently. Chapter 7 describes ways to adapt the Montecarlo mismatch model of MOS transistors to fulfill the needs of PUF design. Chapters 8 to 11 introduce different ways to decrease the bit error rates of PUFs from the circuit perspective. These techniques are important since they help to reduce the complexity of post-processing. The last part deals with the different practical realizations of PUFs: Chapter 12 describes the measurement results of a two-stage PUF, Chapter 13 of a shared amplifier PUF, and Chapter 14 of a PUF with pre-selection. Chapter 15 is a SRAM PUF which was implemented in a microcontroller. Chapter 16 concludes this work.

1.2. Motivation

Globalization and the rise of information technology simplify the interchange of goods and information on a daily base. Nevertheless, the underlying processes become more and more complex.

In consequence of the outsourcing of production to foreign countries and the shipping of goods around the world it is getting more and more difficult to control all single steps related to a product. Thus, potential adversaries have many alternative points of attacks. Furthermore, to handle the vast number of goods, these have to be identified. This is usually done by assigning identification numbers (IDs) to the single products which can be pricey.

As a side effect, the increasing usage of information technologies increases the amount of sensitive data which should be kept invisible for any third party. There are many ways to attack the communication channel to receive actual secret data. For these reasons, the demand for security applications has increased in an inflationary way during the last years. Critical data are stored in different databases constantly: Governmental data, health care data, information about consumer behavior, emails, calendar data, chat records, data at banking institution, and access control information are only few examples. Attackers of all sorts try to get access to such data for different reasons. Many cases are known where confidential data were hacked and published. Even if security systems are steadily improved, attackers find ways to break them. One popular historical example of such a case is the cryptographic machine called *Enigma*. This machine was developed by the Germans in 1920 and was used during the World War II for encrypting military messages. Over the years, the functionality of Enigma was improved to raise the security standard of the machine [132]. Nevertheless, UK scientists in cooperation with Polish colleagues developed the so-called *bombe* which was a machine that helped to break an enigma encrypted cypher text. Both machines are shown in Figure 1.1.



(a) (b)

Figure 1.1.: Cryptography in the World War II. (a) Enigma; (b) Bombe.

The Enigma, as an example, shows the ongoing conflict between people who try to create secure systems and the attackers that try to break them. Since the struggle has not ended up to now, the demand for further improvements of information security exists.

On account of the globalization of trade and the usage of information technologies it is necessary to develop technologies that help to reduce or solve the identification and security issues. One approach to that problem are the so-called physical unclonable functions (PUFs). These functions use production variability to generate fingerprint-like data of physical devices. By means of PUFs, chips can be identified, authenticated, and also used as key generators for cryptographic purposes. All three applications can be used to reduce the above mentioned issues. Therefore, using PUFs, the intrinsic properties of devices can be utilized in communication channels to authenticate the counterpart and also to encrypt the transmitted messages. Furthermore, goods can be identified due to their individual properties and even be authenticated. Thus, not only producers can monitor the production chain but also the customers may verify the origin of sensitive articles like pharmaceutical products, security-relevant spare parts, or luxury goods. For that reasons, physical

unclonable functions have become increasingly popular during the last years. Nevertheless, PUF circuits still exhibit high bit error rates. To assure error-free functionality complex post-processing has to be used. For that reason, the main focus of this work lies on finding ways to reduce the error rate of PUF chips.

1.3. Physical Unclonable Functions

1.3.1. What is a Physical Unclonable Function?

Different definitions of a Physical Unclonable Function (sometimes physically unclonable functions) are given in the literature ([13, 19, 149]). In general, a physical unclonable function is an entity that uses production variability to generate a device specific output which usually is a binary number. This output can be seen as the *fingerprint* of a device (Figure 1.2). A PUF is made of several components defined by local parameter variations. The differences between the components are called local mismatches. Depending on the PUF approach these local parameters are combined, compared or directly read out to generate the binary output. Since the variation of the components cannot be controlled from the outside, a PUF cannot be replicated. This fact makes it unclonable. Depending on the application the PUF output depends on an input signal. Hence, a PUF is a function. To what extend the input signal influences the output differs between the various PUF approaches. The input (challenge) may alter the internal combination of the mismatching components which changes the output (response). The input may also define which of the components should be used to generate the output. There are also alternatives like the use of HASH functions to combine the input and a PUF output to generate an input specific number. From a theoretical perspective, the single words in PUF have the following meaning:
Physical means a physical entity, in contrast to an algorithm or a similar function. If *physically* is used, the meaning changes since now it becomes an adverb to *unclonable* which means that the function is clonable in general but not in a physical way.
Unclonable means that a thing cannot be replicated. For PUFs this is true in practice. Theoretically, PUFs are clonable.
Function means in terms of mathematics that an input value is associated to one specific output value. Since the output of a PUF is usually noisy it happens that an input produces different outputs. And often PUFs are used without any input in literature, for example, if it is used for key generation. Thus, in general a PUF is not a function in the mathematical meaning.

Therefore, we define a PUF as follows:

A PUF is a physical entity which produces an output value at least in dependence of physical structures which are hard to clone.

1.3.2. The Missing Component

As mentioned above, the growing degree of interaction between different people but also between things and all combinations increases the demand for secure and reliable solutions. These include solutions for identification, authentication, and cryptographic purposes [8, 20]. All these applications need unique or unpredictable numbers as a basis. In most cases these numbers are generated outside the purpose device and then saved in a non-volatile memory (NVM). Furthermore, if cryptographic keys are transfered to the device this has to be done in a secure environment which produces extra costs. Example devices are radio frequency identification (RFID) devices, smartcard, near field communication (NFC) devices, and all kind of security hardware.

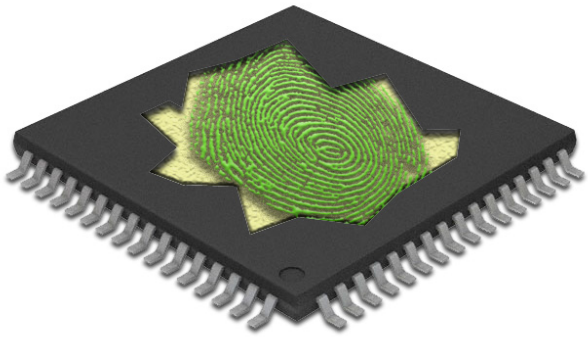


Figure 1.2.: PUFs: The fingerprint of devices.

Due to the fact that PUFs generate a number from the intrinsic properties of a device, number generation and storage has not to be done outside the chips which can reduce the costs significantly. In some cases, this makes a technology competitive after all.

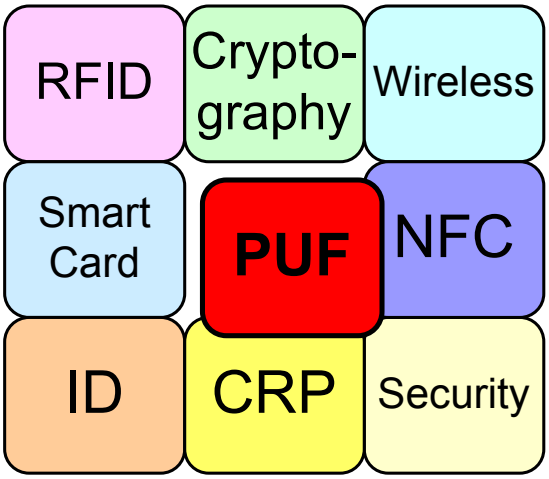


Figure 1.3.: PUF the missing element.

RFID tags are a popular example: RFID is considered to be an alternative to the simple bar code but with extended features like SCM (Supply-Chain-Management), supermarket checkout systems and good management (see section 2.1). Unfortunately, the RFID tag has to be produced including an NVM on which the EPC (Electronic Product Code) is stored. NVMs are expensive to produce which makes RFID uncompetitive. Using a PUF instead would reduce the costs remarkably, and would shift the price into regions which makes an investment interesting. A schematic overview over relevant applications is given in Figure 1.3.

1.3.3. The Advantages: Security and Costs

There are two main advantages of PUFs over common techniques (Figure 1.4): cost reduction and an increase of security. Depending on the application one of the advantages can be a good reason to include a PUF into the product.

Reducing Costs: If a microchip is used for identification purposes, a unique ID must be available. Common systems are storing IDs in NVMs. A chip without NVM is usually cheaper to

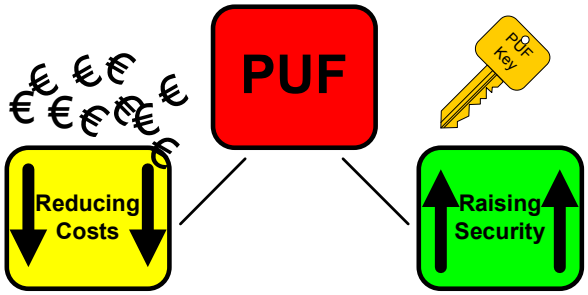


Figure 1.4.: Basic advantages of PUFs.

produce, because additional processing steps are not required. Moreover, the unique ID has to be generated outside the chip and later transfered to it. This causes additional costs. If a PUF is used instead, the ID is inherently available on the chip. If the number of output bits is large compared to the number of required IDs, the PUF output can be assumed to be unique.

Raising Security: Firstly, PUFs are very hard to attack by reverse engineering methods. Thus, the probability of reading out the PUF output can be assumed to be very low. Secondly, a PUF makes it unnecessary to generate confidential data (e.g. cryptographic key) outside the chip and transfer it inside a secure environment. High security levels make these steps very costly. By means of PUFs the confidential data is generated directly on the chip. Furthermore, it does not have to be stored inside NVMs. Using public-key cryptography, the private-key does never have to leave the chip. With such systems unprecedented security standards can be reached.

1.3.4. The Problem: Bit Errors

It is needless to say that PUFs are not perfect. PUFs do not provide exactly the same output each time. PUFs show bit errors. These errors appear randomly and deterministically. The random errors are generated by circuit noise. The deterministic errors are generated by mismatch between parameters of the involved components, e.g. mismatches of temperature coefficients or aging effects. Thus, if no errors are allowed by the application (e.g. key generation), error correction data has to be generated and stored inside NVMs on the chip or outside the chip in data-bases. Both is costly and reduces the advantages of PUFs. For identification purposes, errors may be no problem as long as the distance between the IDs is large enough. Accordingly, even if errors occur, the devices can be identified correctly.

Since the error rate is the crucial disadvantage of PUFs, a big part of this work is dedicated to this problem.

1.4. Different Approaches

During the last decade, many different approaches to PUFs have been published. In the following section an overview of different approaches to PUFs is given. This should not be understand as a complete list, but it should give an idea of the different ways a PUF can be realized. Figure 1.5 shows a rough time-line of the history of PUFs.

For further reading the following texts are recommended: [22] provides a short introduction. In [94] a detailed survey is given. An overview over different transistor-based PUFs was also published by Hirase et al. already in 2005 [61], by Tuyls in 2005 [151] and 2006 [150].

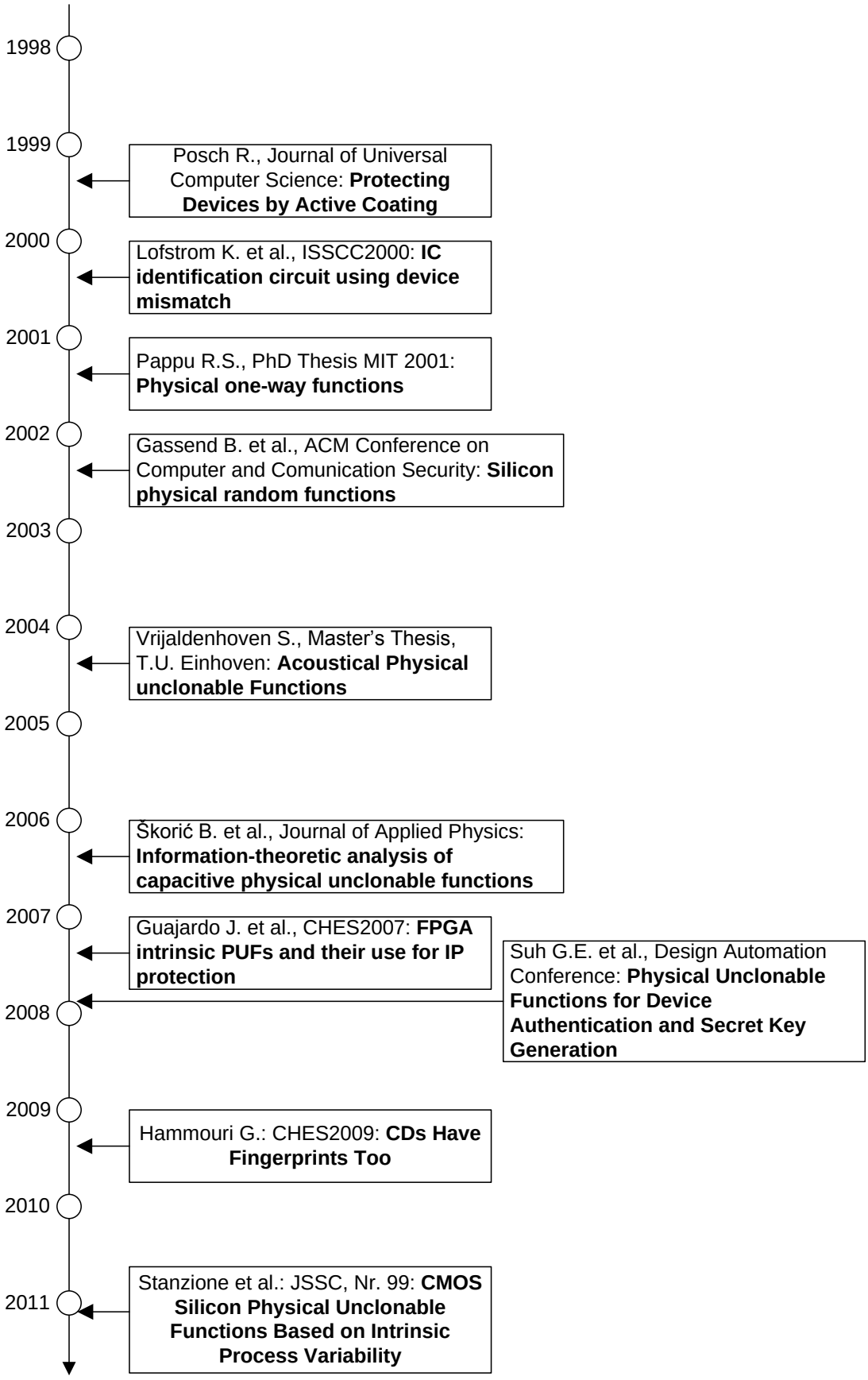


Figure 1.5.: History of PUFs.

1.4.1. First Identification Schemes

In the year 2000 Lofstrom [87–89] presented a way to extract chip-specific data from manufacturing variations by comparing the drain currents of two nominally identical transistors. He used an autozeroing comparator to measure the potential difference between different transistors connected to a resistor. The circuit is depicted in Figure 1.6.

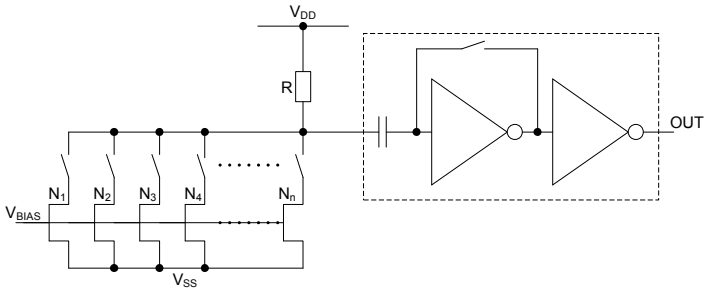


Figure 1.6.: Approach from Lofstrom (see [89]).

To compare two transistors the switch of the first transistor is closed and the amplifier consisting of the two inverters is aligned. Afterwards the switch is opened and another one is closed. Due to the different drain current caused by the mismatch between the transistors the voltage drop at the resistor changes. Depending on the mismatch, the output of the amplifier will either move towards V_{DD} or towards V_{SS}.

1.4.2. Optical PUF by Pappu

The first explicit PUF was introduced by Pappu in 2002 [113]. In this approach a unique output is produced by evaluating the interference pattern of a transparent optical medium. Its high complexity makes this approach difficult to implement. An illustration of the approach can be seen in the Figure 1.7.

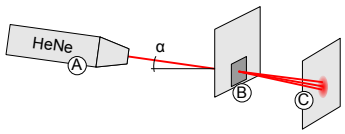


Figure 1.7.: Optical PUF by Pappu (see [113]).

At point Ⓐ a HeNe laser beam is generated which hits an optical medium Ⓑ under a pre-defined angel. Here, the angel of the laser beam is the challenge to that optical PUF. The beam on the other side of the optical medium creates a speckle pattern at an indicator Ⓒ. From this speckle pattern a Garbor-hash is generated. This can be seen as the response of the PUF. Through the various possibilities of different angels a huge number of challenge response pairs can be derived.

1.4.3. Ring-Oscillator PUF

The ring-oscillator PUF was presented by Gassend et al. in 2002 [45,46]. He suggested measuring the differences in frequency of ring-oscillators that are caused by manufacturing variations. In Figure 1.8 the approach is depicted schematically [139].

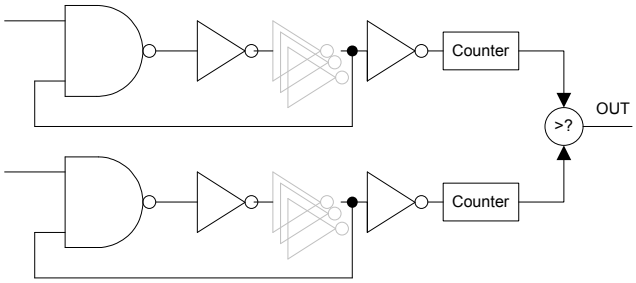


Figure 1.8.: Ring oszillator PUF (see [46]).

In this approach only standard logic gates are used and thus the approach can be implemented on an FPGA. Since no extra processing steps are needed for this kind of PUF these PUFs are called silicon PUFs or intrinsic PUFs. Today, in literature the vast majority of PUFs are silicon PUFs.

An improvement using configurable ROs is suggested in [96]. A characterization is given in [95, 97].

1.4.4. Arbiter PUF

An approach that compares the delay between two digital paths was proposed by Lee et al. in 2004 [81]. The approach is called arbiter PUF since an arbiter circuit decides which of the two delay lines provides the smaller delay [84]. Additionally, the path can be controlled by an input which allows generating several outputs from one PUF circuit. The approach is depicted in Figure 1.9.

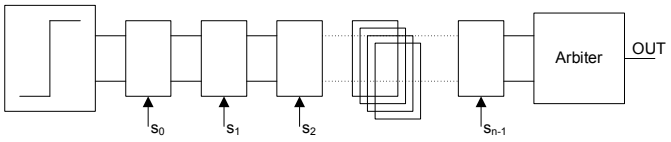


Figure 1.9.: Arbiter PUF (see [81]).

The design of a switch box can be seen in the Figure 1.10.

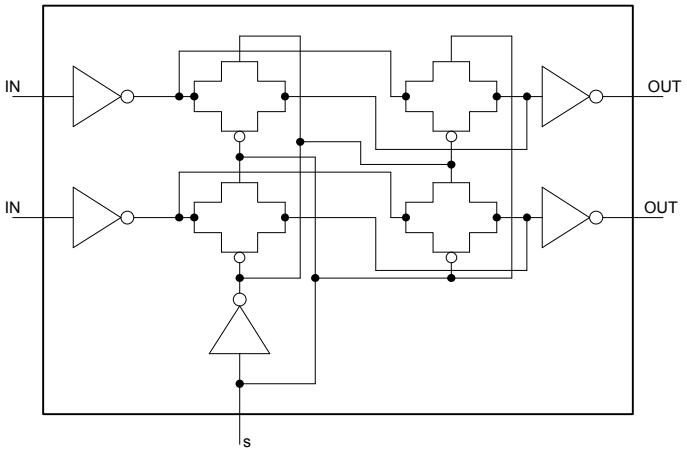


Figure 1.10.: Circuit of a switch box.

Like the ring-oscillator PUF, the arbiter-based PUF can be implemented on a FPGA.

1.4.5. Coating PUF

The idea to use chip-specific data was already proposed by Posch in 1998 [116]. He suggested adding a coating to the chip which is made of a mixture of isolating and conducting material. This coating provides the basis for a chip-specific code. Since physical modification of the coating is followed by a difference in the code, attacks can be detected easily. Later in 2005, Tuyls et al. suggested the so-called coating PUF which uses the device specific data extracted from the coating for authentication and identification purposes [124, 133]. The principle is shown in Figure 1.11.

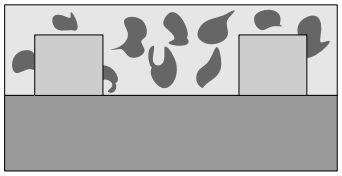


Figure 1.11.: Coating PUF (see [133]).

1.4.6. SRAM PUF

SRAM PUFs use existing SRAM blocks to generate chip specific data. After powering-up the circuit the cells stabilize at a state which is defined by the mismatches between the involved transistors. Thus, each SRAM cell provides one bit of output data. The SRAM PUF was first proposed by Guajardo and Holcomb in 2007 [52, 65, 66] . The SRAM approach can also be implemented by using FPGAs or Microcontroller [18]. The problem with this approach is that not every SRAM implementation is suitable for PUF purposes.

So, SRAM PUF-like circuits were developed. One approach including reset circuitry was proposed by Su in 2008 [138] (see Figure 1.12).

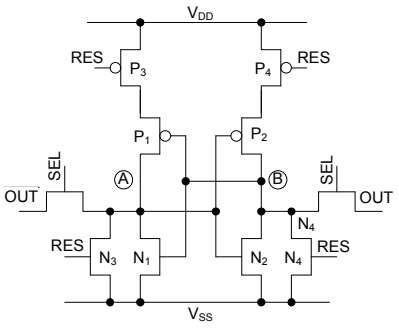


Figure 1.12.: Resetable SRAM PUF (see [138]).

The transistors N_1, N_2, P_1 and P_2 are building a common SRAM cell. The transistors N_3, N_4, P_3 and P_4 are used to reset the cell. If the RES signal is ON, P_3 and P_4 cut the SRAM cell from the power supply, N_3 and N_4 are forcing the nodes Ⓐ and Ⓑ to V_{SS}.

Since the SRAM PUF is a simple approach the underlying circuit is well-known and the circuit is used as a basis for the PUFs that are designed and analysed in this work, the SRAM PUF will be introduced more detailed in Section 1.6.

The SRAM PUF can also be used in sub-threshold region [74]. The advantage of this approach is the high relative mismatch in sub-threshold region. Nevertheless, the absolute mismatch and the signal itself is very small so that noise can become a severe problem.

1.4.7. The Inverter-Based Approach

A further approach is an inverter-based one which was introduced by Puntin et al. in 2008 [117]. An extended version was published by Stanzione et al. in 2011 [137]. In the circuit several NMOS transistors share one PMOS transistor (see Figure 1.13). The circuit is similar to the approach in section 1.4.1.

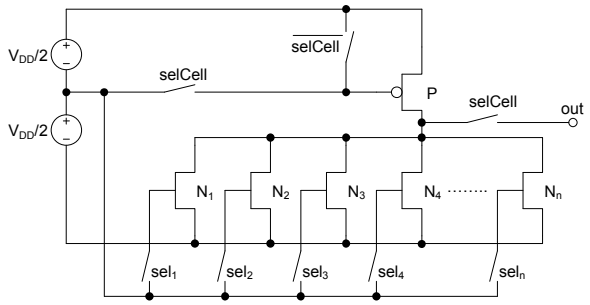


Figure 1.13.: Inverter based PUF (see [117]).

During usage one of the NMOS transistors is connected to the PMOS. In this approach the NMOS transistors are designed for high mismatch (i.e. small size). The PMOS transistor should have little influence on the decision. To minimize mismatches, the PMOS transistors have to be chosen sufficiently big. The input of the inverter is set to $V_{DD}/2$. The inverter works in the maximum gain region. The output voltage of the inverter is compared to the output voltage of another NMOS/PMOS pair which is different due to the parameter variation. An auto-zeroing comparator is used in this step. If the first pair shows a higher voltage than the second pair a '1' appears at the output. Otherwise a '0' shows up. To compensate for variations in environmental conditions a process regulation circuit regulates the voltage $V_{DD}/2$ at the input of the inverters. This reduces the bit error rate. Additionally, a decision margin is introduced which is used to select those pairs which show voltage differences that exceed a certain value. This is important in order to reduce the BER due to noise and operation point shifts. The cell selector is depicted in 1.14.

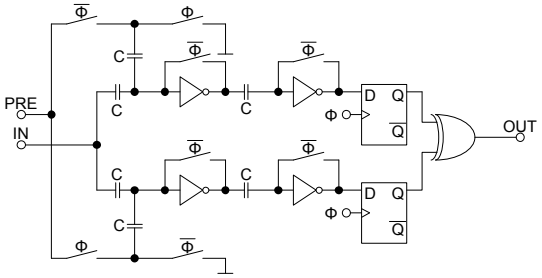


Figure 1.14.: Stable cell detector (see [137]).

The pre-selection of stable PUF cells is a basic step to reduce the error rate significantly. More about the theory of pre-selection schemes is shown in Section 10.

The inverter-based PUF approach achieves good results in terms of BER, area, and energy consumption.

1.4.8. Other Approaches

In addition to the above mentioned PUFs different other approaches were published in the last few years. The butterfly PUF was proposed by Kumar et al. in 2008 [79]. Choi et al. suggested a PUF using differential amplifiers [27]. In 2011 Majzoobi et al. published a ultra-low power current-based approach [98]. Ganta et al. published a leakage-based approach [44] in 2011, too. An approach that utilizes system equivalent resistance variations was published by Helinski et al. in 2009 [59]. A current starved inverter chain PUF is suggested by Kumar et al. in [78]. Reconfigurable PUFs are described in [80]. PUFs based on D flip-flops were published in [152] and [93]. A PUF that is based on device aging was introduced in [100]. A tristate buffer PUF was published by Ozturk et al. in [112] in 2008. Sensor-based PUFs were introduced in 2010 [123]. Selimis et al. work on SRAM PUF-based key generation in wireless sensor nodes [127]. PUFs that use lithographic variations are described in [135]. A delay PUF based on glitch shapes was published by Suzuki et al. in 2010 [141]. A RO-PUF that uses not only process parameters but also takes variations like temperature dependence or dependence on V_{DD} into account was suggested by Wang et al. in 2010 [156]. A RO PUF implemented on an FPGA was published by Yu at al. in 2009 [161].

1.4.9. Summarizing of Different Approaches

Table 1.1 shows the performance of some of the published PUF approaches. The intra-chip (HD_{intra}) and the inter-chip (HD_{inter}) Hamming distances are used as performance parameters. A detailed description of the performance parameters can be found in Chapter 4. Unfortunately, information on the area and power consumption is not given for all of the presented approaches, therefore these parameters are not taken into account in this work. Furthermore, the temperature range for which the intra-chip Hamming distances were determined vary between the various publications. To compare the approaches in terms of that parameter the presented results have to be inter- or extrapolated. Additionally, in most cases $V_{DD}$was varied during the analyses which also leads to an increase of HD_{intra}. Since temperature shifts were the dominant source of error, the results from the V_{DD}variations are not presented here.

1.5. Theoretical Analysis

In addition to the different approaches many theoretical papers were published concerning mathematical models of PUFs, modeling attacks of PUFs from their challenge response pairs and on further security issues. Since this is not in the scope of this work, the reader is referred to the literature.

1.6. Exemplary PUF Circuit: The SRAM PUF

One of the simplest circuits in silicon which can be used for PUF purposes is the SRAM cell. The SRAM consists only of a few transistors and is easy to understand. Since in an SRAM cell minimum size transistors are used, the local mismatch between the involved transistors is large. This can be a problem in common SRAM design [3] but it turns into an advantage as soon as it is used for PUF purposes. Different PUF circuits are based on the SRAM circuit. Furthermore, the

Table 1.1.: Performance of different approaches.

Approach	Ref.	Temp. (°C)	HD_{intra}(%)	HD_{inter}(%)
ID Cell	[89]	-25..125	5	50
Ring-Oscillator PUF1 [a]	[139]	20..120	0.48	46.15
Ring-Oscillator PUF2 [a]	[96]	25..65	<2	47.31
Arbiter PUF	[139]	20..120	9	23
SRAM PUF1	[52]	-20..80	12	50
SRAM PUF2	[65]	room	5	bias to 1
Latch PUF	[138]	0..80	5.5	50
Inverter-Based PUF [a]	[117]	20..125	0.4	50
Butterfly PUF	[79]	-20..80	6	50
D-Flip-Flop PUF1 [b]	[93]	room	<5	50
D-Flip-Flop PUF2	[152]	-40..80	10	35
Glitch-Based PUF	[141]	0..80	<8	40

[a] With kind of pre-selection (see Chapter 10).
[b] After bias compensation.

SRAM PUF is used in the following chapters to describe fundamental properties of PUFs. To get an idea of the functionality, the SRAM PUF is analyzed in more detail in the next section.

1.6.1. Fundamentals

An regular SRAM cell usually contains 6 transistors. In Figure 1.17a such a cell is shown. A SRAM cell is a bistable circuit. To explain the basic functionality of a bistable system the physical model of Figure 1.15 is used.

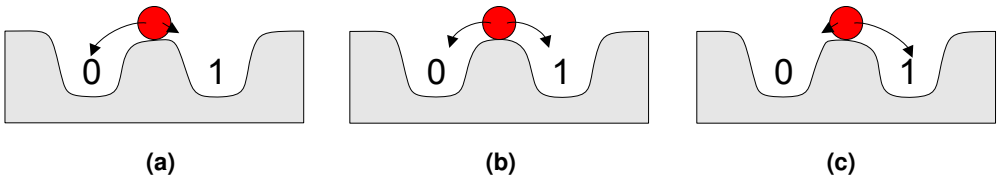


Figure 1.15.: Bistable system: (a) Bias towards '0'; (b) 50:50 chance; (c) Bias towards '1'.

In Figure 1.15b a totally balanced system is shown. The probability that the ball moves to the left side is the same as the probability that the ball moves into the right direction. The systems in Figure 1.15a and Figure 1.15c are not balanced. In the case of Figure 1.15a, the system is biased towards '0'. In the case of Figure 1.15c the ball will probably fall to the right side. The same behavior can be observed when looking at the power-up voltage curves of SRAM cells (Figure 1.16).

The bias of a SRAM cell is defined mainly by the mismatch of the PMOS transistors. This bias defines the state of the SRAM cell after powering up the circuit. Since the bias cannot be controlled during production, a symmetrical designed SRAM cell can be utilized as a PUF circuit.

A 6T-SRAM (6 transistor SRAM) circuit and the equivalent circuit consisting of two inverters can be seen in Figure 1.17a.

During powering-up the circuit V_{DD} increases towards its final value (e.g. 1.35 V). As soon as the first PMOS transistor (either P_1 or P_2) starts to provide much current (i.e. V_{TH} of the tran-

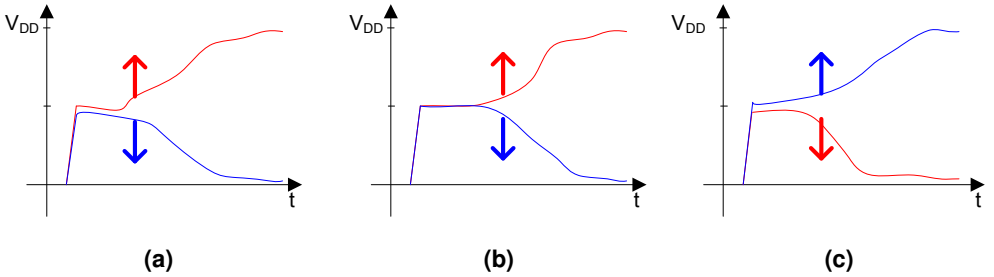


Figure 1.16.: Bistable electronic circuit (e.g. SRAM): (a) Bias towards '0'; (b) 50/50 chance; (c) Bias towards '1'.

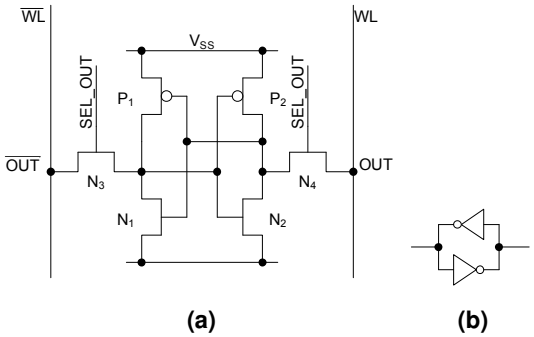


Figure 1.17.: (a) Regular 6T-SRAM cell; (b) Equivalent circuit.

sistor is reached) the corresponding node Q/Qn starts to increase voltage which at the same time decreases V_{GS} of the second PMOS transistor and increases V_{GS} of the corresponding NMOS. Thus a latching effect occurs and the cell either outputs V_{SS} or V_{DD}. If the difference between the two transistors P_1 and P_2 is very small, both start providing current nearly at the same time. Thus, Q and Qn increase in voltage until the V_{TH} of one of the transistors N_1 and N_2 is reached. Now, the latching effect is defined by the NMOS transistors. It can be seen that in most cases the output of an SRAM PUF cell is defined by the threshold voltage difference ΔV_{THP} of the PMOS transistors. Only in cases of very small ΔV_{THP} NMOS transistors' threshold voltage difference ΔV_{THN} dominates the decision. The voltage characteristics at Q and Qn of three different Montecarlo simulation runs are shown in Figure 1.18. It can be seen that one branch always dominates the decision by faster increase in voltage. As soon as V_{GS} reaches V_{TH} of the dominant transistor the cell starts to latch.

In Table 1.2 the threshold voltage values (BSIM: V_{TH0}) of the transistors of the three Montecarlo runs are shown. The decision is always defined by the V_{TH} values of the PMOS transistors. The dominant transistor is printed bold. At the second run the V_{TH} values of the PMOS transistors are very close which affects the power-up procedure: for the second run, the latching happens late (see Figure 1.18).

1.6.2. Discrete Implementation

To get a feeling for the behavior of SRAM PUFs a discrete version was built from a dual N-channel and a dual P-channel matched MOSFET pair (ALD1103PB from Analog Devices). The threshold voltage of these transistors are typical around 0.7 V. The power-up behavior of that bistable circuit was analyzed. The results are depicted in Figure 1.19. The blue curve shows the power supply, the red and the yellow curve show the nodes Q and Qn. The horizontal resolution is

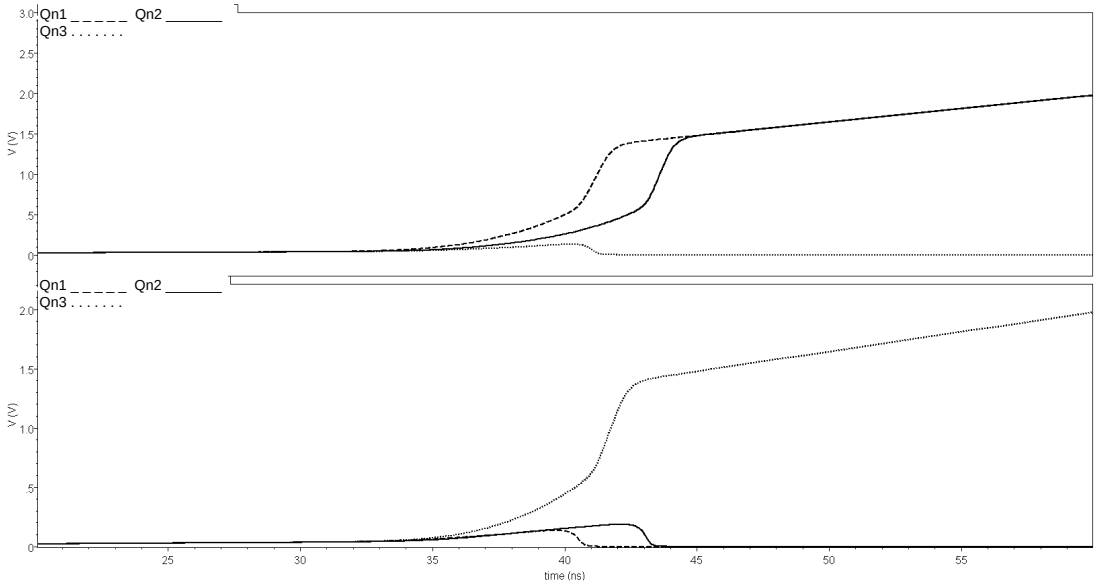


Figure 1.18.: Three different (Montecarlo simulation) SRAM cells are powered-up (slow increase of V_{DD}). The latching takes place as soon as the dominant transistor reaches V_{TH}.

Table 1.2.: V_{TH0} values of mismatching SRAM transistors.

Transistor	V_{TH0}/V, run 1	V_{TH0}/V, run 2	V_{TH0}/V, run 3
N_1	0.467	0.494	0.508
N_2	0.483	0.494	0.525
P_1	**-0.648**	**-0.695**	-0.740
P_2	-0.696	-0.704	**-0.670**

set to 10 ms. As the simulation shows the decision process of the SRAM PUF can be seen nicely. The process is depicted for different power supply voltages. Since the decision time significantly depends on the power supply voltage, it varies between 100 ms (Figure 1.19a, V_{DD}=0.95 V) and 1 ms (Figure 1.19h, V_{DD}=1.3 V).

The SRAM PUF behavior can also be explained using a simple equivalent circuit where the gate capacitances are depicted (Figure 1.20).

The capacitances C_1 and C_3 and the capacitances C_2 and C_4 build capacitive voltage dividers respectively. As soon as the power is switched on the capacitances are charged. As shown in the Figure 1.19 the voltage in between the capacitances settles around 0.2 V_{DD}. This is an indicator that the capacitances of the NMOSTs (C_3, C_4) are larger than the capacitances of the PMOSTs (C_1, C_2). Subsequently, the voltage at the two nodes sinks a little bit. This behavior indicates that the current in the NMOST is higher than the current in the PMOS transistor. The two voltages between the nodes increase slowly and finally the circuit settles at one of its stable points.

1.6.3. Analysis of SRAM Chips When Used as PUFs

In addition to the analysis of a SRAM PUF from discrete components (see Section 1.6.2) five SRAM chips where tested towards their usability as PUFs. In this section the experimental results and an analysis of the occurring problems is given. Table 1.3 shows the analyzed memory chips.

Table 1.3.: SRAM Chips.

Model	Manufacturer	Size
CDP1822R	Intersil	256 x 4 bit
HM6116	HMI	2k x 8bit
MB8416	Fujitsu	2k x 8bit
SRM20256	S-MOS Systems	32k x 8bit
PCF8570P/F5,112	NXP	256 x 8bit

The following paragraphs presents the test results:

CDP1822R

The initial value of this SRAM depends on the speed of the powering-up process. A sharp edge causes an initial output value of '0' a flat ramp causes an initial value of '1'. Due to the dependence on the power supply this chip is not suitable for PUF purposes. It turns out that the asymmetrical layout of this SRAM cell causes the problem (see Figure 1.21).

HM6116, SRM20256

The output bits of this SRAM changes for the same chip at different runs. Thus, it is not useful for SRAM PUFs. It seems that the decision is strongly effected by noise. This property is useful for random number generators but counterproductive for PUFs.

MB8416

The initial states of this SRAM always exhibit the same pattern for each byte. The pattern is 10011001. The internal design of the chip is not known but it is assumed that the pattern is generated due to asymmetries in design or layout. This chip is not useful for SRAM PUF purposes as well.

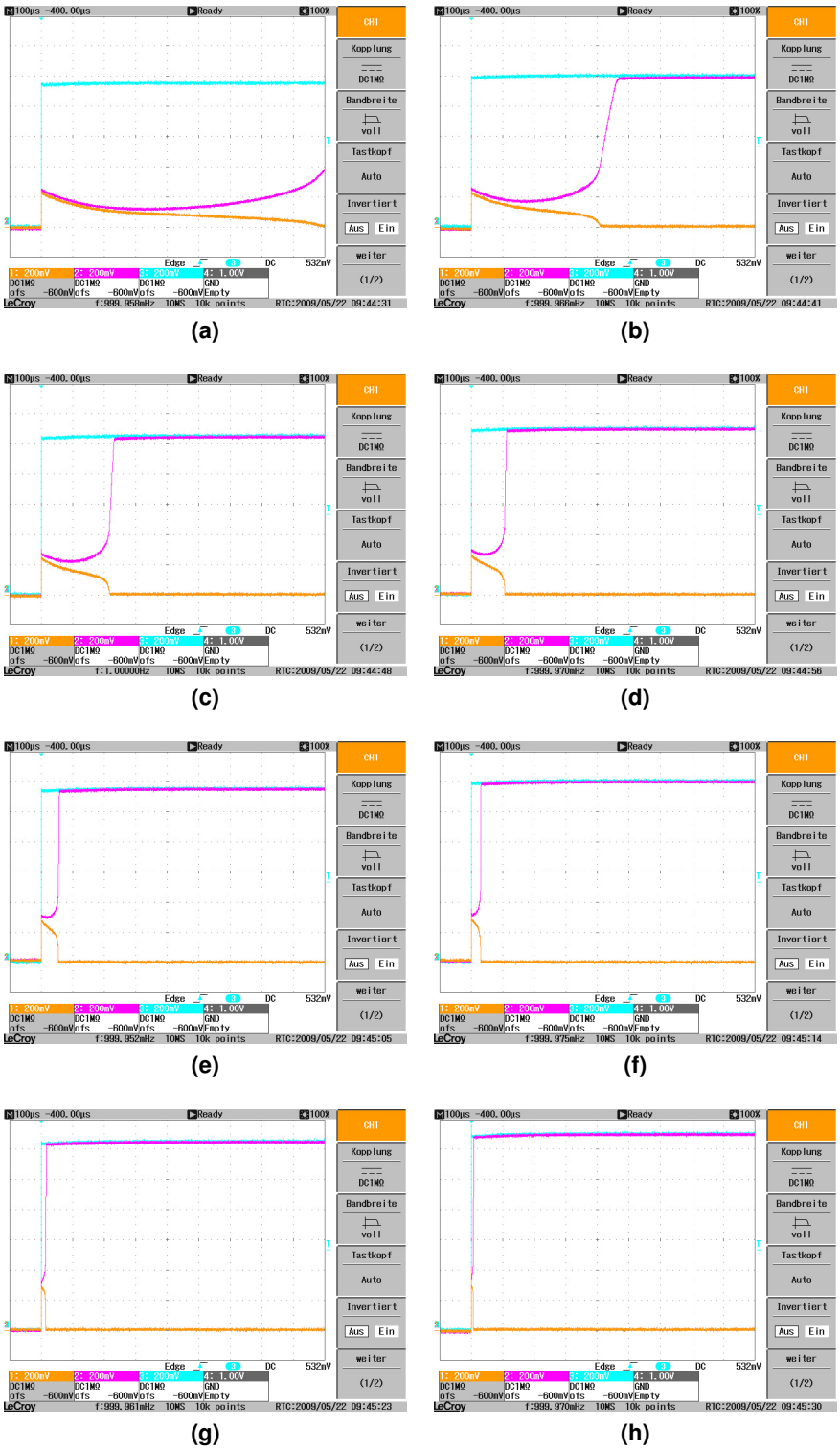


Figure 1.19.: The figure shows the decision process of a discrete implemented SRAM cell applying different supply voltages: (a) V_{DD}=0.95 V; (b) V_{DD}=1 V; (c) V_{DD}=1.05 V; (d) V_{DD}=1.1 V; (e) V_{DD}=1.15 V; (f) V_{DD}=1.2 V; (g) V_{DD}=1.25 V; (h) V_{DD}=1.3 V.

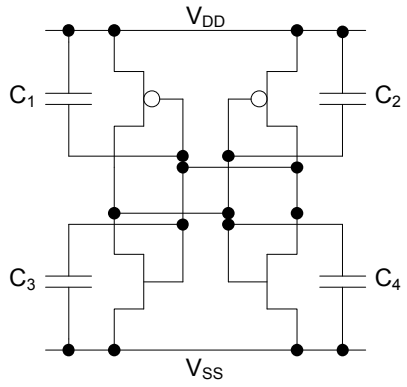


Figure 1.20.: SRAM cell including the gate capacitances.

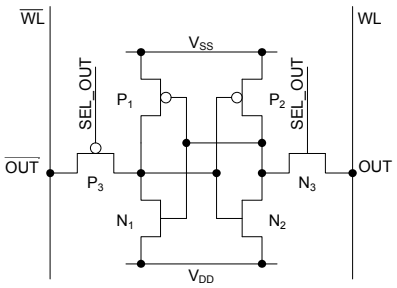


Figure 1.21.: Asymmetrical layout (see P_3 and N_3) of the CDP1822R SRAM cell.

PCF8570P/F5,112

The best results were achieved with this chip. The PCF8570P/F5,112 is a 2k SRAM with I2C interface. The initial states are not predictable but stable. Unfortunately, the two stable states are not uniformly distributed. The probability p_1 of occurrence of '1' is around 0.13, the probability p_0 of occurrence of '0' is about 0.87. Therefore, even this chip cannot be recommended without limitations.

As a conclusion to this analysis it is important to state that by far not all SRAM circuits fulfill all the requirements of PUFs (see Section 4.3). It is important to check SRAMs carefully before using them for PUF purposes. Of course, this statement is also true for SRAMs of FPGAs or Microcontroller (see Chapter 15).

1.7. Patents

The following section is on patents concerning PUFs that were filed in the last years. It is by far not a complete listing of available patents. It should rather give an overview of the variety of companies that are involved into the topic. The listed patents are all US patents and applications.

1.7.1. Identification Using Parameter Variability

Using parameter variability to gain fingerprint-like data from manufacturing variability is the basic idea behind PUFs. Even though PUFs may exploit such data for further applications and also make the output depend on an input signal, all PUFs base on manufacturing variability. The first patents

that take advantage of mismatches between components suggest to generate unique numbers to identify devices. Some of them are presented below.

ICID 1999

The first patent that utilizes random parameter variability was filed in 1999 by Lofstrom and his company named ICID [86]. The idea is to use the random parameter variations of different cells that are compared to each other to generate a unique ID from the intrinsic properties of a micro-electronic device. This block can be included into any chip to provide a ID without spending extra effort for ID generation and storage on the chip. An example circuits can be seen in Figure 1.22

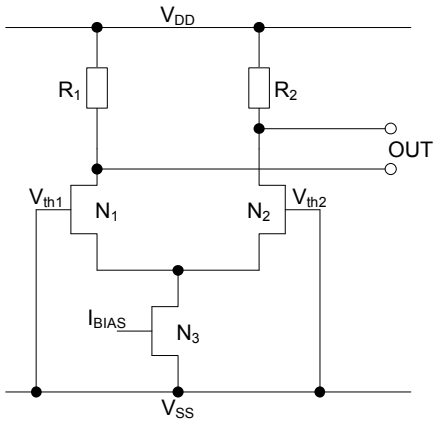


Figure 1.22.: Example of identification cell from [86].

Microvision 2000

In the year 2000 Microvision filed a patent that was finally issued in 2009. In this patent an identification scheme is proposed that uses an intrinsic ID. To do so the ID is read out several times to identify instable bits. Only stable bits are being used for identification later.

Hitachi 2001

Also Hitachi filed a patent concerning identification by means of parameter variations [99]. This was done in 2001. The patent was issued in 2005. As an example circuit inverters are used where the output is connected to the input. Each inverter settles at its specific voltage. These voltages are compared by a measurement circuit. An example can be seen in 1.23.

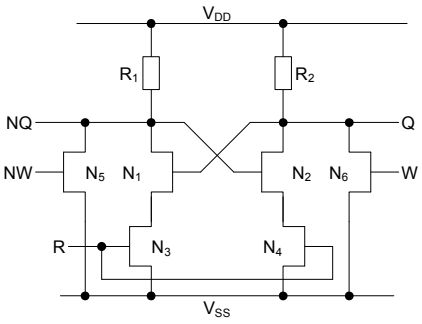


Figure 1.23.: Example of identification cell from [99].

STM 2003

In 2003 ST Microelectronics filed a patent on a resistor-based identification device. Based on process variability of two resistors a binary value is generated. The patent was issued in 2004 [159]. An example circuit can be seen in Figure 1.24.

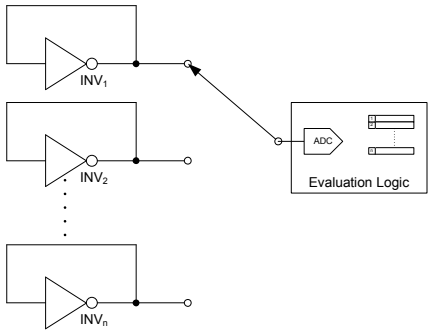


Figure 1.24.: Example of identification cell from [159].

PIXIM 2004

In 2004 PIXIM filed a patent that also uses manufacturing variability of all kind of circuit elements. This includes bipolar and MOS transistors, resistors, capacitors and inductors or light detecting pixel elements. But also memory cells and amplifiers are included. The elements' properties are evaluated by comparing the cells with a reference voltage or by digitizing the output signal. The SRAM PUF is covered by this patent. The patent was issued in 2007 [15].

Infineon 2007

In 2007 [76] Heiko Koerner from Infineon Technologies filed mainly the same idea as Lofstrom including more circuit details. Via random variations, a circuit can be identified using a certain evaluation unit. The patent was issued in 2011.

National Semiconductor 2008

In 2008 Elroy Lucero, employee of National Semiconductor, filed a patent that somehow expanded his patent described in 1.7.4. This patent describes an SRAM PUF that is used to generate a unique number without going into application details. The patent was issued in 2009 [91].

National University Corporation Tohoku University, Advantest Corporation 2008

Another chip identification procedure, based on process variability, was filed by Okayasu et. al in 2008. The received data are used to detect degradation of the device. As an example, the threshold voltage variation over time can be detected by adequate circuits. The patent was issued in 2010 [111].

1.7.2. Key Generation From Parameter Variability

On-chip key generation is another important PUF application. There exist several patents on key generation utilizing process variability.

STM 2002

In 2002 ST Microelectronics filed a patent that utilizes process variations to generate a secret quantity. This quantity can then be used by an application. Furthermore, methods using shift registers and non-volatile memories are suggested to be used in addition to the secret generator. The patent was issued in 2010 [160].

LSI 2005

In 2005 David Barr employed at LSI Corporation filed a patent that uses a PUF output to generate a key that will never leave the system and therefore is never visible to the outside. In this patent he also suggests error correction that ensures bit error free keys. This patent was issued in 2009 [10].

Xilinx 2005

In 2005 Stephen Trimberger employed Xilinx filed a patent that includes the usage of PUF output data. The data is used as a key for en- and de-cryption of configuration data that is stored outside the device. As a result, the data is protected against undesired copying. The patent was issued in 2011 [143].

MIT 2006

In 2006 the Massachusetts Institute of Technology filed a patent that describes the use of PUFs for accessing data stored in an integrated circuit or using the data for key generation for cryptographic purposes. The generated data depends on a digital input. The patent was issued in 2010 [33].

Irdeto 2009

In 2009 Irdeto Access B.V. filed a patent that utilizes PUFs to prevent the cloning of pay-TV receivers for example. To do so, challenge-response pairs of a PUF that is part of the receiver are stored in a data-base at the transmitter side. The responses of the PUF are used as keys to encrypt a message. At the receiver the message is decrypted. The patent was not issued yet [30].

1.7.3. PUF Authentication Schemes

PUF-based device authentication is also covered in several patents. Some of them are listed below.

ICID 2002

In 2002 Chi-Song Horng employed at ICID filed a patent about a authentication procedure for random binary ID codes like the ones proposed by Lofstrom. Since errors are produced due to noise, some of the bits do not always return the same value. The position of the unstable bits itself are unique for different ICs. The patent proposes to use this unique positions of unstable bits for the authentication process. The patent was issued in 2004 [67].

MIT, Intrinsic ID 2003

In 2003 the MIT and Intrinsic ID filed a patent about a PUF authentication scheme. In this patent a multi bit input to a PUF is used to get a device specific output. Since this kind of authentication is a main application of PUFs, it is a major patent which was finally issued in 2010 [28].

Philips 2004

In 2004 Pim Tuyls et al. filed a patent for authentication of physical objects utilizing their physical properties. This is exactly what PUFs do, if used for authentication. The patent was issued in 2011 [147].

1.7.4. Others

The following patents did not match the preceding sections:

Verayo 2005

In 2005 Verayo Inc. filed a patent on a PUF that is implemented on a field configurable device (i.e. FPGA). It utilizes the process variabilities to generate a device specific output depending on a configuration input. An example is a arbiter-based PUF which is implemented on an FPGA. The patent was issued in 2010 [35].

National Semiconductor 2003

In 2003 Elroy Lucero of National Semiconductor filed a patent that explains the layout steps that have to be undertaken to build a highly symmetrical SRAM cell. This is done to maximize the randomness in the power-up state. The only meaningful application for such a circuit is to create fingerprint-like data. The patent was issued in 2009 [90].

LSI 2006

In 2006 the LSI Corporation filed a patent that uses multiple read-outs of the same cells to detect unstable cells. The unstable cells are not used to generate the output anymore. The patent was issued in 2010 [154].

LSI 2006

In 2006 the LSI Corporation filed a patent that uses fuses to mark unstable (soft) cells. The patent was issued in 2009 [54].

Tuyls, Schrijen 2009

In 2009 Tuyls and Schrijen filed a patent that describes how to lower the error rate by using biasing the circuit in dependence of the first output. By using this technique the output of a PUF can be stabilized. The biasing leads to an increase of the mismatch and thus to a reduced error rate. The patent was not issued yet [148].

1.8. Related Topics

In this section three PUF-related topics are covered more detailed. This should help the reader to get a deeper insight into the fundamentals of PUF applications. The three topics are RFID, cryptography and biometrics.

1.8.1. RFID

A wide field of PUF applications are RFID (radio-frequency identification) related. In many cases the PUF is an integral part of the RFID tag. Due to this fact this section takes a closer look at RFID systems.

During the last years RFID (radio-frequency identification) became more and more popular [119]. As the name suggests RFID tags were originally meant for identification purposes. Therefore, many identification applications are already around. Example are book identification in libraries, the electronic road pricing (ERP) in Singapore as well as electronic door keys. In Figure 1.25 my-d tags from Infineon Technologies are depicted.

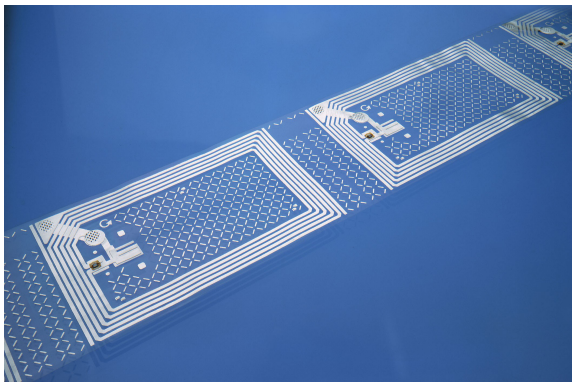


Figure 1.25.: My-d tags from Infineon Technologies (carrier frequency = 13.56 Mhz).

RFID tags often include NVMs with capacities between a few bits and several kilobytes. In addition to identification tags there are also tags which include security modules for authentication and other purposes [69].

The different RFID systems can be classified by their frequency. A listing of the frequencies is shown in in Table 1.4 [158].

Table 1.4.: Frequency range RFID.

Frequency range	Frequencies	Distance
Low frequency (LF)	120-140 KHz	10-20 cm
High frequency (HF)	13.56 MHz	10-20 cm
Ultra-high frequency (UHF)	868-928 MHz	3 meters
Microwave	2.45 & 5.8 GHz	3 meters
Ultra-wideband (UWB)	3.1-10.6 GHz	10 meters

A further classification can be done by the power supply. On the one hand, there exist passive tags which receive the required energy over the electromagnetic field coming from the reader (Figure 1.26a). Simple passive tags can be very cheap. The price can be as small as a few cents per item and the size can be as small as 0.05 mm x 0.05 mm (chip by Hitachi [24]). On the other hand, active tags are available (Figure 1.26b). Those tags are more expensive because they use on-board batteries for providing the energy. Also combinations of active and passive tags are receivable. For example, wireless car keys can start the car even if their internal battery is empty. For this purpose 3D transponders are developed that can fulfill the requirements [75].

More information on RFID can be found in various literature, for example in [42].

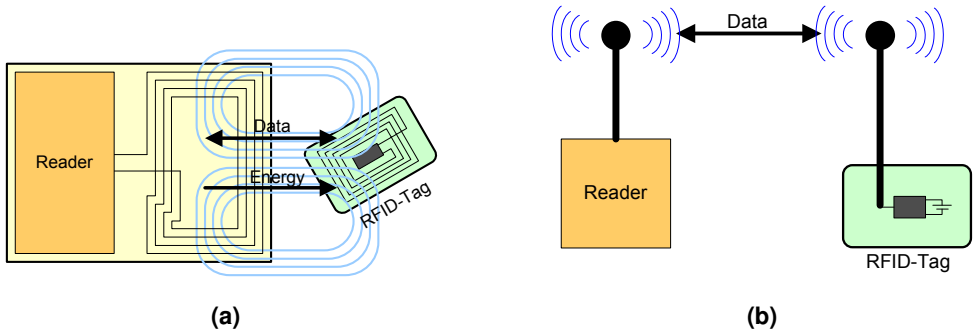


Figure 1.26.: (a) Passive RFID Tag; (b) Active RFID Tag.

1.8.2. Cryptography

As mentioned above PUFs are used for cryptographic key generation and storage purposes [51]. For this reason, some cryptographic basics are covered in the following section.

In general, there are two different systems of encryption: the first one is based on symmetric-key algorithms and the second one on public-key cryptography. Both are introduced bellow.

Symmetric-Key Algorithm

Symmetric-key algorithms use the same key for encryption and decryption of messages. Therefore, the key has to be kept secret on both sides of the communication channel: the sender and the receiver. The basic idea of symmetric-key cryptography is shown in Figure 1.27.

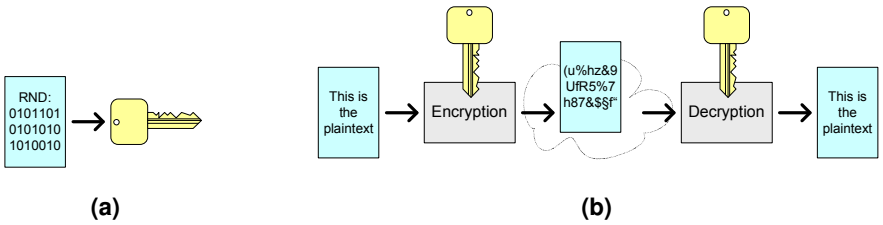


Figure 1.27.: Symmetric-key cryptography: (a) Key generation; (b) Encryption.

The key generation is very simple. No special preconditions for the key exist. Any secret random number can be used (see Figure 1.27a). The key can be used to encrypt a plain text and decrypt the cipher 1.27b. Popular algorithms of symmetric-key cryptography are the data encryption standard (DES) [101] and the advanced encryption standard (AES) [73]. During the last years the AES has become more and more popular for its higher security level. The DES which has a key length of only 56 bits can be attack by brute force attacks [136]. In contrast, AES can be used with variable key length of 128, 192 and 256 bits. Up to now and presumably in the near future even the 128 bit version cannot be cracked using brute force attacks.

Public-Key Cryptography

The first public-key algorithm was published by Rivest, Shamir and Adleman in the year 1978 [121]. Public-key, or asymmetric-key cryptography uses two different keys for encryption and decryption. As the name suggests one of the keys is typically public: anyone may access the key

without lowering security. The second key is private, which means that only the owner may have access to it.

In Figure 1.28b the private key is used to encrypt the plain text and the public key is used to decrypt the cipher text. Such systems can be used for signing a text. Only the owner of the private key is able to encrypt the plain text in such a way that the owner of the corresponding public key can decrypt the message. Therefore, the sender of the message can be verified by the receiver using the public key. Such cryptographic systems are also used for authentication.

Alternatively the public key can be used for message encryption. Here everyone who knows the public key can send encrypted messages over a public channel like the Internet to the private key owner. Only the owner of the private key is able to decrypted this message.

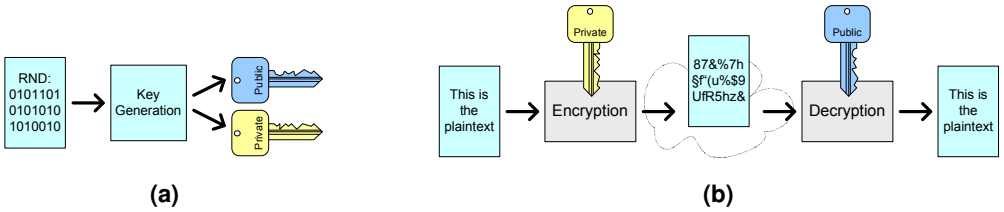


Figure 1.28.: Public-key cryptography: (a) Key generation; (b) Encryption.

In contrast to the symmetric-key algorithms not every key can be used for public-key cryptography. A private and a public key have to be generated using special algorithms.

Popular public-key algorithms are the RSA algorithm and elliptic curve cryptography (ECC) [53]. A disadvantage compared to symmetric-key algorithms is the higher complexity of the en- and decryption processes and thus the higher energy demand [155]. This might be a problem in RFID applications when passive tags are used. ECC has to be preferred for low-power applications [108].

1.8.3. Biometrics

Biometrics is the authentication of human beings using physiological and behavioral characteristic data [72]. Some examples are shown in Figure 1.29. Physiological data for example are fingerprints (Figure 1.29a), DNA, iris recognition(Figure 1.29b), face recognition (Figure 1.29c) and hand geometry (Figure 1.29d). Behavioral characteristics for example are voice and gait. [136]

There is a wide range of applications for biometrics. As an example, it is used for authentication of persons when access control to a safety area is needed. Another example is forensic science where DNA profiling is being used [2].

Certain parallels can be drawn between biometrics and PUFs: a PUF can be seen as the fingerprint of an microchip. Thus, the example *fingerprint* is used for comparison purposes.

First of all, a reference fingerprint is necessary which is also called exemplar print. During an enrollment phase the fingerprint is collected (Figure 1.30a). At the identification phase the received fingerprint (e.g. from an access control system) is compared with the exemplar prints (Figure 1.30b). If the correlation is high enough, the fingerprint will be assigned to a certain ID.

The identification using PUFs is very similar. During an enrollment phase a reference vector will be collected (Figure 1.30c). Later, during the identification phase, the PUF vector is collected again and will then be compared to the entries of a ID database (Figure 1.30d). If the correlation to the reference vectors is high enough, the PUF will be identified correctly.

To provide a high number of correct identifications, it is very important to get a good reference object for the database. In the case of a fingerprint identification the contours of the considered area of the clean finger should be sharp. In the case of a PUF identification the reference ID should

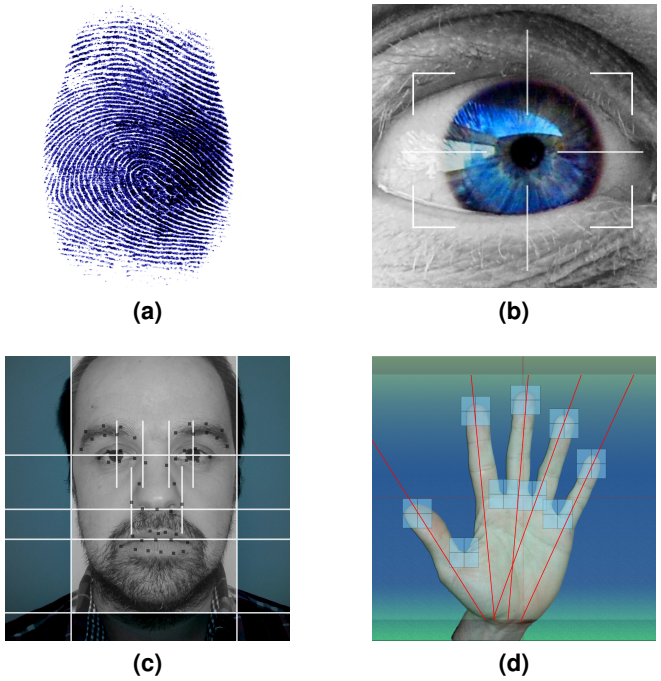


Figure 1.29.: Biometric methods: (a) Fingerprint; (b) Iris recognition; (c) Face recognition; (d) Hand recognition.

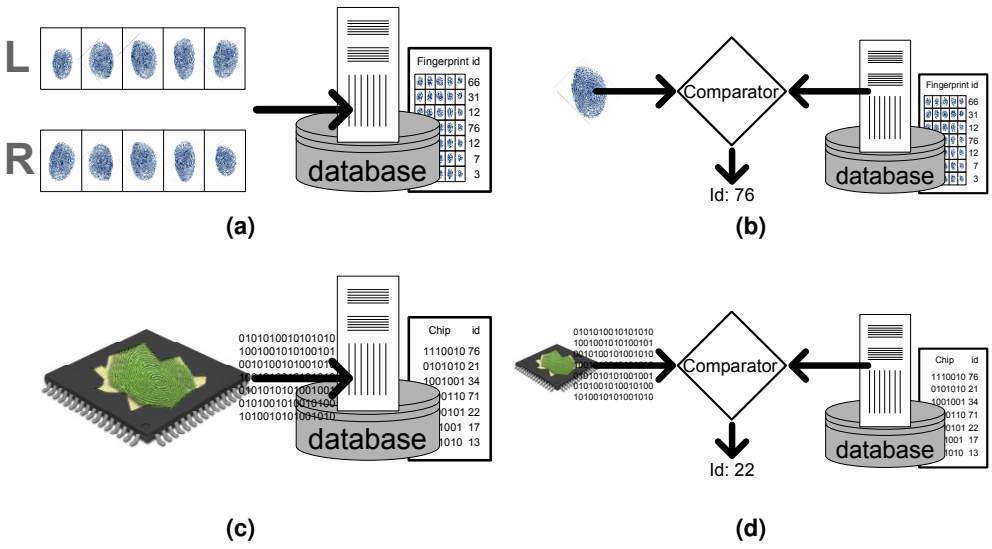


Figure 1.30.: Comparison of Biometric and PUF identification: (a) Enrollment fingerprint;(b) Identification fingerprint; (c) Enrollment PUF; (d) Identification PUF.

be measured several times or even at different environmental conditions to receive reliable data. Another approach to increase reliability, in the case of the fingerprint only significant characteristics are used for identification. There are also concepts where only characteristic PUF cells are used. These approaches will be discussed in Chapter 8 and the following chapters. The following similarities between PUFs and biometrics exist:

- Both can be used for identification and authentication purposes.
- Measurements of both are noisy.
- Both are hard to clone.

2. Use Cases

by Maximilian Hofer

In this chapter some use cases of PUFs are shown. As already mentioned in Chapter 1 PUFs are introduced to reduce costs, to increase the security level or sometimes both. A wide range of applications exist. Technical realizations are covered in the following chapters. This chapter highlights some prominent problems and in what way they could be solved using physical unclonable functions. In section 2.1 commerce related examples are discussed. Section 2.2 is on encryption of pay television. In section 2.3 the night-shift problem and possible solutions are discussed. Section 2.4 covers different anti-counterfeiting approaches. In the last section a technical approach to the FPGA code protection using PUFs is shown.

2.1. Commerce (Supply-Chain-Management)

A typical example of identification using a PUF is shown in [7]. Here, a PUF is an integrated part of an RFID tag. One RFID tag is attached to every product package. Thus, every good is inseparably bounded to an RFID tag and therefore also to the unique PUF number of the tag. Such RFID tags can be seen as upgrade of common bar codes. Common bar codes are not used to identify products but to assign the product to a group of products.

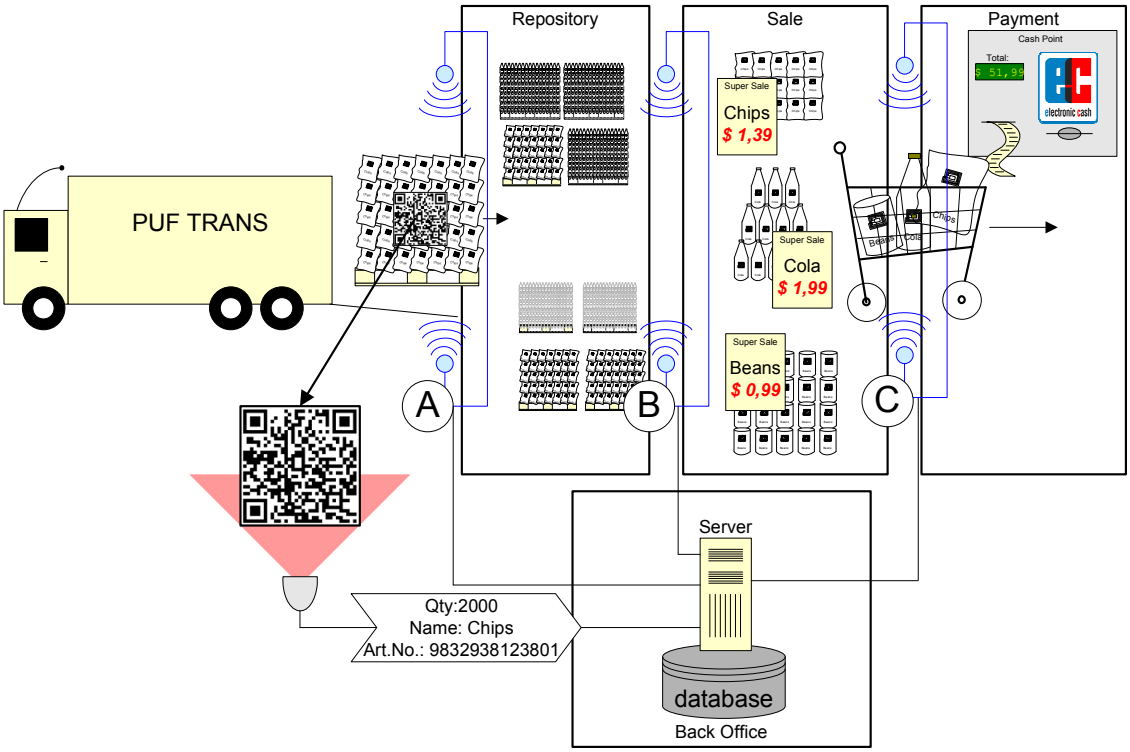


Figure 2.1.: Practice and commerce.

However, the advantages of RFID tags compared to bar codes have to exceed the additional costs. Especially in commerce cost pressure is extremely high. Bar codes are printed on the packaging together with the residual printing. Therefore, the bar code comes nearly for free. RFID tags have to be produced and later assigned to the package which is an additional production step. Hence, RFID tags have to be produced as cheap as possible to get competitive. There are two crucial reasons why PUFs can reduce the costs of such RFID tags:

- No RAM/ROM is required on the tag to store the ID.
- The ID is inherently available on the tag. So, no enrollment phase is necessary to write the ID on the tag.

In Figure 2.1 the principle handling of an PUF-RFID tag used for product identification is shown: at some point at the beginning of the producer-costumer chain the goods have to be registered. The data is stored in a database. The PUF-ID of every good has to be assigned to an article-ID. The article-ID helps to receive informations about the properties of the respective good. Additionally, good specific properties can be stored into the database. The best before date, the product manufacture or charge numbers can be stored, as an example. In the example of Figure 2.1 the registration and the transfer of the data into the database is done at point Ⓐ. At this location all delivered goods are registered. A whole pallet of goods - in this case 2000 pieces of potato chips - is scanned at once. On the pallet itself a bar code gives the general information about the incoming goods. This bar code is read and assigned to the 2000 IDs from the RFID-tags. The data is stored into the database. From the warehouse the goods will be delivered to the stores. There the goods can be easily registered by reading out the RFID-tags. This is depicted at Ⓑ of Figure 2.1. In the store the consumer puts the goods into the shopping cart. At the check-out (Ⓒ), the tags are read out again and the bill is generated automatically.

As an addition, the fridge at the consumer's home is also featured with an RFID-reader (see Figure 2.2). Such a feature would allow for many different services: recipe proposals, best before date warnings, shopping list proposals, nutritionist functions, and many others. This feature alone could result in completely new business models.

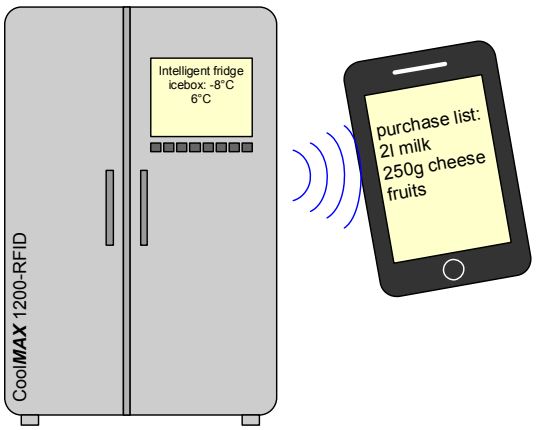


Figure 2.2.: Fridge with RFID reader.

The list below shows the advantages (+) and the disadvantages (-) of RFID-based good identification:

+ Staff savings.
+ Exact traceability of transport chain from manufacturer to consumer, e.g. the cooling chain must be unbroken for special goods.

+ Easy and exact inventory.
+ The number of human errors can be reduced, e.g. at the check-out.
+ Additional features like automatic shopping list generation, etc.
+ Less queuing at the check-out.
- RFID tags are more expensive than bar codes.
- Fast, long distance RFID readers are required.
- Used materials must be RFID conform (package, cans, carts, palettes, etc.).
- Consumer acceptance can be a problem. Privacy and health concerns could occur.
- Technical issues concerning readers and tags have to be solved.

Summing up, for sure a market for such approaches exists. However, the initial costs are high which might discourage investors.

2.2. Pay Television

In pay-TV, also referred to as premium televisions or premium channels, the video stream is encrypted to allow only authorized users to watch the program. In Digital Video Broadcasting (DVB) receivers a common interface (CI) module is used for this purpose. In practice, CAMs (Conditional Access Modules) are used which are able to carry a smartcard that contains the key for the used system [39–41]. One of these systems, as an example, is the Irdeto's content management solution which is used from British Sky Broadcasting [70].

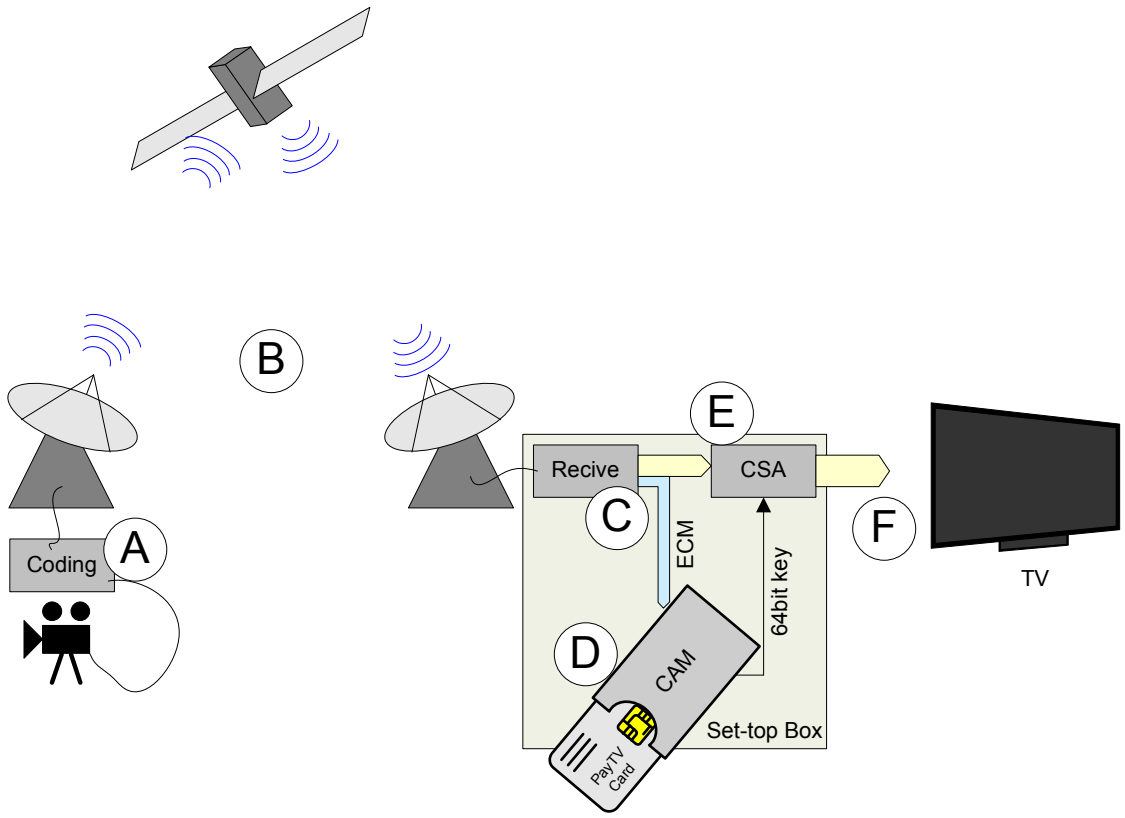


Figure 2.3.: Transmission system and conditional access system.

In Figure 2.3 the principle of the transmission system is shown. In point Ⓐ a DVB signal is generated from a video source. The MPEG (moving picture experts group) video signal is encrypted by the CSA (common-scrambling-algorithm) using a 64 bit key. Usually the key is changed every few seconds. A ECM (Entitlement Control Message) stream is also transmitted within the DVB signal. The signal is transmitted to the consumer. In Figure 2.3, a satellite is used to transmit the signal Ⓑ. Usually QPSK is used for DVB-S (EN 300 421 [39]) and QPSK, 8PSK, 16APSK or 32APSK are used for DVB-S2 (HDTV) (EN 302 307 [41]). Of course, there are also other transmission paths available like transmission over the Internet or over terrestrial channels. The receiver (Ⓒ) in the consumer set-top-box must be capable to interpret the signal correctly, though. In the case of pay-TV the output of the receiver consists at least of a CSA encoded MPEG stream and an ECM (entitlement control message). By means of the ECM, a CAM (conditional access module), and a smartcard the key for the decryption of the video system is generated. In the figure, this process can be seen in Ⓓ. Finally, a chip in the set-top-box which handles the common-scrambling-algorithm is decrypting the video-stream. The data are transmitted to the screen.

Since people are not always willing to pay for the services pirate decryption is a well known issue for pay-TV provider. There are various methods to get illegal access to encrypted TV signals. In Figure 2.4 a procedure of a potential attack is shown. The architecture and a valid key are necessary to hack the system.

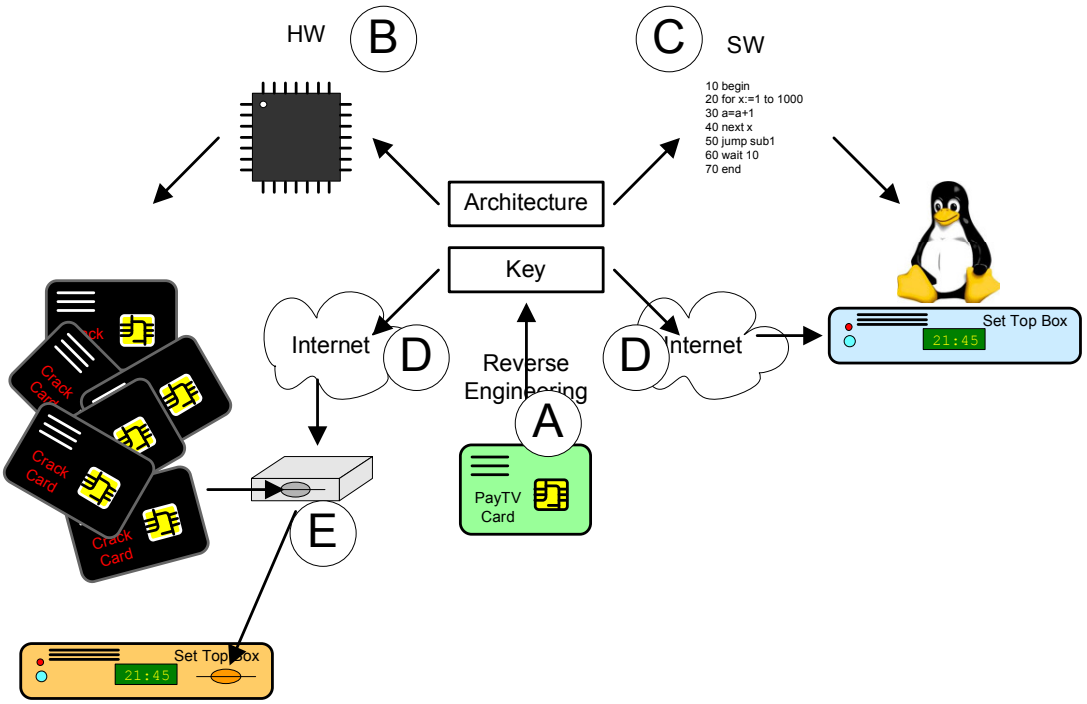


Figure 2.4.: Pirate decryption by reverse engineering of the pay-TV smartcard.

As a requirement one or more legal smartcards have to be available. As the first and most difficult step the card has to be reversed engineered (Ⓐ). Alternatively, the design of the smartcard may get into public due to security problems in the companies which are assigned for the development or the production of the smartcard. A valid key may be gained during its transmission process. Manufacturers have to introduce high security levels to prevent illegal access to these keys. This goes with high efforts and costs. If the architecture of the smartcard is available the rebuild can be done either by hardware or by software. In Ⓑ a new hardware is produced and

included into the smartcard. A valid key is stored with a smart card reader on the card. Now the card can be used more or less as the original pay-TV card. A second way to get illegal access to pay-TV program is via software. An example is shown shown in ©. The whole hardware (CAM and smartcard) is emulated in software. The new software can be installed on the set top box. Valid keys can be provided over the Internet.

How is it possible to prevent the hacking of smartcards? As stated above and shown in Figure 2.4, there are two requirements to hack a system: the architecture of the smartcard and the key have to be known. The architecture has to be kept secret by *security through obscurity*. Since it has to be assumed that the statement of Claude Shannon *The enemy knows the system* [129] is true, the algorithm in combination with the key has to be *secure by design*. Therefore, the key has to be stored on a device in a secure way within a hostile environment. Flash memories, e-fuses and other memories are the normal approach to store the key on the chip. There are two security issues related to that approach: Firstly, the data in the memory may be read-out by an adversary even if the chip is not powered up. Secondly, the key has to be transferred from the outside to the chip. Here PUFs can be a step towards more security. In this case PUFs are used, the PUF provides the private key for the card directly from the intrinsic properties of the cards itself. If an asymmetric cryptographic approach is used, the public key is also generated on the chip and then transfered to the key server of the conditional access system. Furthermore, the PUF output is only available during the read-out of the PUF-block which also complicates a possible attack.

2.3. Night-Shift Problem

The night-shift problem is shown in Figure 2.5. These days the production of chips is often

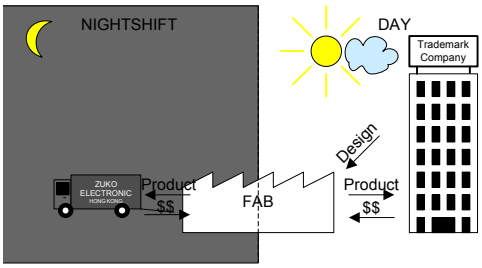


Figure 2.5.: Night-shift problem.

outsourced to foreign countries. That is why the design of the product is often done by different companies than the production of the chips itself. Companies that cooperate with contract manufacturers are often afraid that the producing companies fabricate more than the ordered number of items. Such overproduction is then illegally sold to third parties. This illegal business is called the *night-shift problem*.

A very simple approach to solve this problem is to include a PUF on the chip. This PUF can be read out by everyone. During the chip test/verification/enrollment phase the ID of the PUF is stored in a database. Of course, this step has to be controlled by the design company. If a chip is assumed to be sold illegally later, its ID can be read out and checked against the items in the database. If the ID does not appear in the database the assumption is approved. This information makes it possible to check, if a chip was produced on a legal or an illegal way.

Another approach is to activate the chip before selling. An authorized party is able to generate a code depending on the PUF ID to activate the chip. Due to the fact that the ID is a unique property of the chip, the activation code differs from chip to chip and thus cannot be generated easily by criminals.

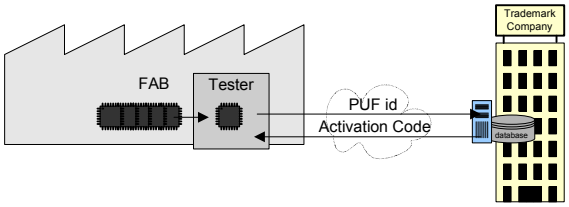


Figure 2.6.: Activation of chips with a PUF.

A schematic illustration is shown in the Figure 2.6. During testing the chips the PUF ID is read out and sent to the ordering party. The party has the algorithm and the secret key to calculate the activation code. The activations can be stored in a database. This would be a convenient way to control the activation of the produced chips.

2.4. Anti-Counterfeiting

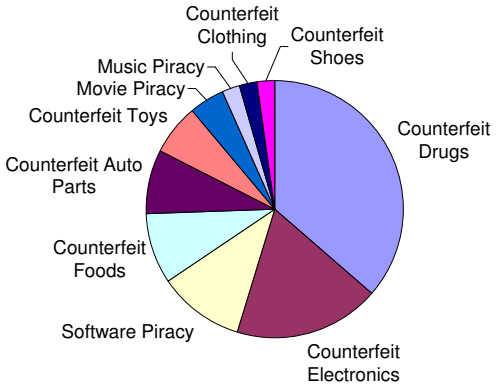
The following is a well known procedure: on a Turkish beach you can buy a branded wrist watch for $ 10 instead of $ 3990 at the jeweler. No one assumes that the watch bought on the beach is an original product. But the costumer, who buys a watch for $ 4000 at the jeweler wants to be sure to get an original product featuring the expected quality and exclusiveness.

This is only one example. Counterfeit consumer goods become an increasing problem in various business areas [109,110]. A ranking of the different areas is depicted in Table 2.7a , [58]. The table shows an estimation of economical damages to the US market caused by product counterfeiting.

Rank	Field	Value[a]
1	Counterfeit drugs	$ 200 Billion
2	Counterfeit electronics	$ 100 Billion
3	Software piracy	$ 58.8 Billion
4	Counterfeit foods	$ 49 Billion
5	Counterfeit auto parts	$ 45 Billion
6	Counterfeit toys	$ 34 Billion
7	Movie piracy	$ 25 Billion
8	Music piracy	$ 12.5 Billion
9	Counterfeit clothing	$ 12 Billion
10	Counterfeit shoes	$ 12 Billion

[a] In US Dollars.

(a)



(b)

Figure 2.7.: Ranking of counterfeiting-caused damages in the US market.

Different approaches are depicted to avoid counterfeiting [126]. PUFs can be used to fight counterfeiting as well [49,50]. PUFs can provide authenticity of goods. Thus, products can be authenticated at any point in the supply chain [34,82]. In the following paragraph drug counterfeiting is used as an example to illustrate the approach in detail.

Example: Counterfeit Drugs

Table 2.7a shows that in terms of pure volume drug counterfeiting is the most prominent type of product counterfeiting. Since the Internet becomes a major channel of drug distribution, control-

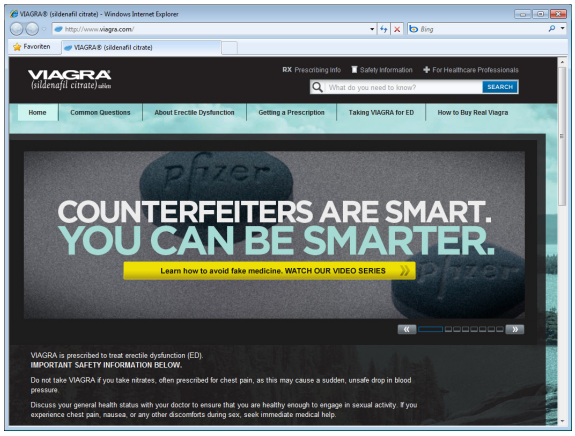


Figure 2.8.: Startsite of the web page www.viagra.com (14.2.2012).

ling becomes much more difficult. One famous example is the potency product *Viagra* also known as *Sildenafil* [21]. The blue pills help to treat erectile dysfunction and are produced by the pharmaceutical company Pfizer. Since for many costumers the most convenient way is to buy it over the Internet, Viagra can be seen as a paradigm. The dimension of the problem can be seen when opening Viagra's website (http://www.viagra.com). The website is shown in the Figure 2.8.

Furthermore, drug counterfeiting is not solely an economical problem. It also comprises health risks for the consumer. This makes drug counterfeiting especially dangerous. Therefore, solutions have to be found that allow all involved parties to check the product authenticity at any time.

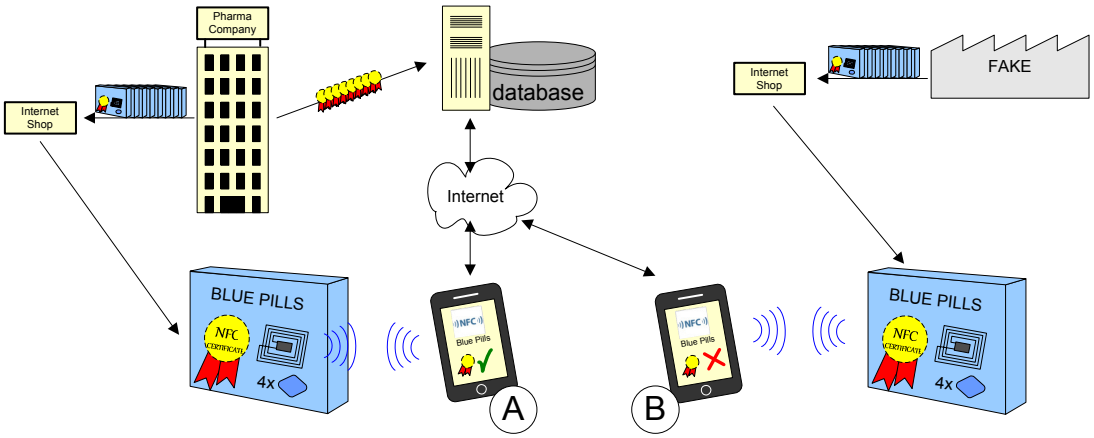


Figure 2.9.: Drug anti-counterfeiting.

In Figure 2.9 a possible authentication procedure is shown. Every protected good is equipped with a near field communication (NFC) tag (NFC is a special form of RFID [157]) including a PUF and an authentication algorithm. The tag can be read out using an NFC-compatible device. NFC is an upcoming feature in mobile phones. The NFC-device is communicating with a database over the Internet. In the database all certificates are being stored. The certificates are provided by the pharmaceutical company. Such a certificate can be made in different ways. It may comprise numbers composed of an ID and a public key from a asymmetric encryption algorithm. Only authorized parties are able to insert a new dataset into the database. Once the goods are registered any party is allowed to check for authenticity. In case Ⓐ of Figure 2.9 the consumer bought an original product with valid certificate. Therefore, the drug is authenticated. In case Ⓑ a counterfeit

product is checked. Therefore, the consumer gets the information that the product has no certificate which means that the consumer is in possession of an illegal product.

Such a system is not only suitable for drugs but also for other goods. For example, clothing can be protected by integrating a label into the product. A further application can be the protection of spare parts. Brake pads in a car are parts where counterfeited products could be a security risk for instance. Such pads could also be labeled and checked at the garage before mounting. The system could be extended to a system that does not start the engine unless all spare parts are certified by the car producer. Different possible technical solutions for authentication procedures are described in Section 3.3.

The main equipment and infrastructure used for the above authentication schemes are listed bellow:

- **Producer of the tag:** the RFID tag has to pass highest security standards.
- **Enrollment station:** the authentication data of the tags have to be inserted into a database; this could be done by the producer of this tag. If this is done by the producer of the good, the security level must be high to prevent the nigh-shift problem (see Section 2.3).
- **Authentication server:** the authentication server has to be controlled by a trusted party. One server can be used for example for all drugs of one company. It makes sense to operate different server for different business areas. Thereby the level of security can be adapted depending on the requirements.
- **Consumer:** the consumers need a device NFC-ready device. NCF-ready devices are available as mobile phones or as NFC-USB-reader.
- **App:** on the NCF-device an application software has to be installed that manages the communication between tags and server. This application has also to fulfill certain security standards.

Altogether, the infrastructure is easy to implement. The costs are small compared to the achieved security level. Mass produced tags are cheap compared to the gained benefit. Public-key cryptography is usually costly in terms of energy. These could cause a problem in passive RFID tags where only a limited amount of energy is available [11].

As in the case of Section 2.1 further services can be implemented once such an authentication system is available: it is possible to include dosage information on the drug. Another application could be to check the compatibleness to other drugs or to give information about the expiration date. Since the information is available electronically it could be made accessible to visual handicapped people for instance.

2.5. FPGA-Code Protection

In this section software code protection in FPGA is described as a final example. Here, FPGAs are one example of reconfigurable and programmable digital circuits. Others would be PLDs, DSPs or microcontrollers. In hardware development such electronic devices have a wide field of application. Even standard devices can be programmed in a way to include complex functionality. There are different aspects that make the use of such devices very attractive compared to ASICS or standard electronic components. These include time-to-market, well tested hardware, costs limited production and reconfigurability.

Typically, during power on the device, the software is loaded into the FPGA. This makes it easy for adversaries to clone the system. An FPGA is a standard product and the software can be read out from the NVM or can be recorded during the startup phase. To make software cloning more difficult two procedures have been introduced [130]: the NVM can be directly included

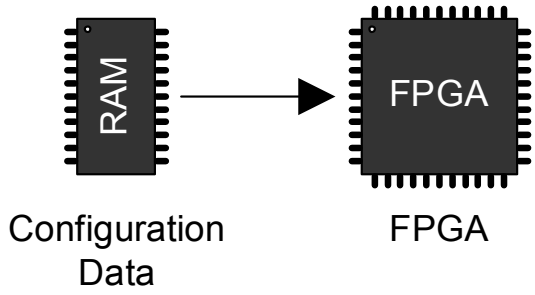


Figure 2.10.: Configuration of an FPGA.

into the package. Nevertheless, the problem still exists as long as updates have to be loaded by the costumer since an attack can also be carried out during transferring the data to the FPGA. Therefore, this still is not a satisfying solution. In [104] and [105], the concept of encrypting the configuration software is proposed. The encrypted software is stored outside the FPGA. During the startup phase the encrypted configuration files are loaded into the volatile memory of the FPGA. Using a key that is stored inside an internal NVM the data can be decrypted inside the FPGA. The decrypted data can now be used to configure the structure in the FPGA. The whole approach is shown in Figure 2.11 [5].

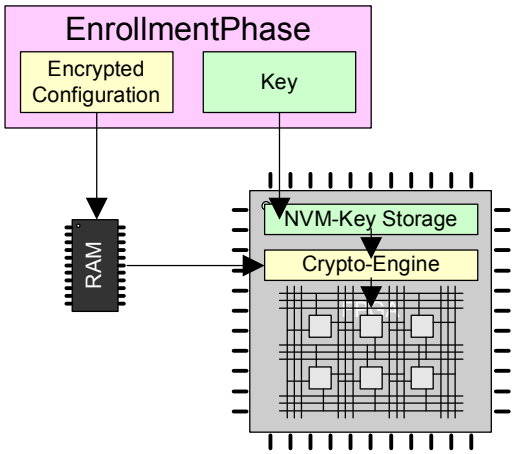


Figure 2.11.: FPGA with encrypted configuration system.

As shown in Figure 2.11 still at least a small NVM is needed inside the FPGA to store the key. That means that the plain key has to be saved somewhere inside the NVM. Here too, the usage of a PUF can improve the security level. A dedicated PUF module can be included or standard SRAM cells can be utilized for this purpose. The error correction data can be stored outside the FPGA as long as they do not include any information on the keys. To do so, no extra NVM is needed but the same memory chip can be used as for the encrypted configuration data. In Figure 2.12, the principle of the PUF FPGA code protection is depicted. During the enrollment phase the system is prepared to decrypt the configuration file. This has to be done in a secure environment. After that the key is sent to the FPGA. Inside the FPGA, the PUF cells are read out. The output is used to mask the key. The error correction code generates the helper data (HD). From now on the key can be reestablished each time it is used. Some examples of error correction procedures are discussed in Chapter 5.

Figure 2.12b shows the upload of the configuration data into the external NVM. The vendor

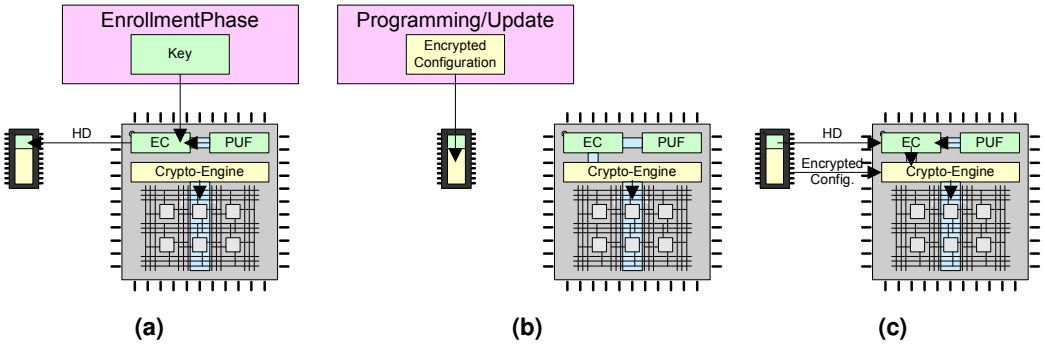


Figure 2.12.: FPGA with PUF Code protection: (a) Enrollment Phase: Generating the Helper Data; (b) Storing the encrypted configuration data; (c) FPGA configuration.

encrypts the configuration data before providing it to the customer.

In Figure 2.12c, the FPGA configuration process is depicted. Firstly, the key is reconstructed using the helper data and the PUF output. The key is then provided to the cryptographic engine. Now, the configuration data is read in and decrypted. Afterwards the FPGA can be configured. If new configuration data is available, the update can be easily distributed to the systems over any public channel without security risks. As already mentioned, a cryptographic engine and PUF cells have to be available on the FPGA for this approach. Some FPGAs already include a cryptographic engine [6]. Since an SRAM PUF can be implemented using the internal SRAM cells, such an FPGA provides all blocks needed.

Different implementation suggestions exist in the literature [52, 79, 107]. A further approach for IP protection using public-key cryptography is described in [48].

3. The Basic Applications

by Maximilian Hofer

As mentioned in the introduction, there exist three basic applications of physical unclonable functions: identification, authentication, and key generation [29]. Since electronic authentication is done with the help of cryptographic tools, the basic applications can be grouped into application with

- cryptographic purposes.
- non cryptographic purposes.

In Section 3.1 an introduction to the identification with PUFs is given. Section 3.2 gives an overview over key generation. Finally, in Section 3.3 the authentication process using challenge-response pairs (CRPs) generated by PUFs is explained.

3.1. Identification

In identification a known ID is assigned to a unknown entity. In the context of microelectronics and PUFs, such an ID is assigned to a chip [47]. The basic requirement for identification is that for each entity a *unique* ID is assigned. This is important to prevent false identifications [122]. Thus, the identification process has to be an injective function. In Figure 3.1 a typical identification process

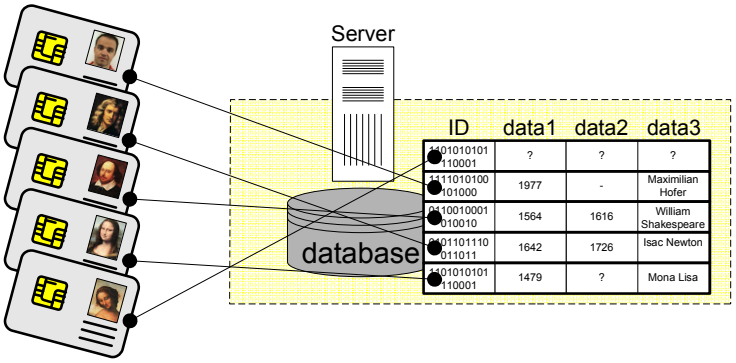


Figure 3.1.: Identification of identity cards.

is depicted. Here, identity cards are assigned to datasets in a database. In the database, additional properties of the objects are attached. That is why the ID can be used to get further informations about the claimed identity. But still, identification does not mean that the claimed ID gets verified. In general, identification does not include identity verification. It is not assured that the claimed ID is the correct one. If the identity is verified, it is called authentication. Authentication is treated in Section 3.3.

When PUFs are used in identification systems they should replace the process of generating the ID externally and replace the internal NVM on which the ID would be stored. If the implementation of the NVM can be saved by PUF usage, the costs per device can be reduced remarkably.

Therefore, PUFs can reduce costs by reducing fabrication complexity and by providing the ID from the intrinsic properties of the chip. This can be a huge advantage in cases where cheap chips are required. One example for that is to use RFID tags as an alternative to bar codes. The idea of identification with the help of PUFs is shown in Figure 3.2 [64].

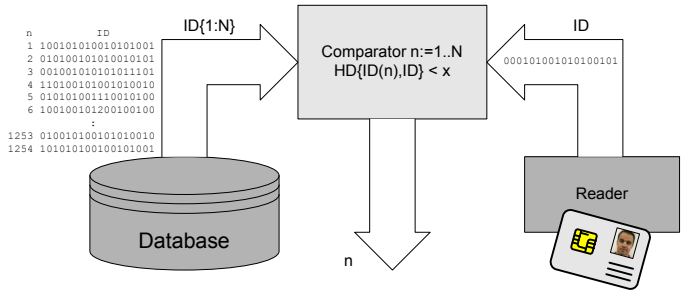


Figure 3.2.: Identification process with PUFs.

Since the output of PUFs is noisy, typically the identification system has to be error tolerant. Depending on the tolerance of the system, error correction codes can even be omitted. The measure of the tolerance for binary output data is called *Hamming distance*. If the error correction code (ECC) is omitted, the assignment of the data to an entry in the database is done as follows: during the enrollment phase, the output of the PUF is stored in a database. During an regular identification process the PUF is read out again. The received number is compared to the entries in the database. Due to errors in the PUF output, exact matching of the identification number and an database entry is unlikely. There are two options how to handle tolerance in the system:

- This dataset entry is chosen which shows the smallest distance to the received ID. The disadvantage of this method is that defect chips or chips which are not registered in the database are also mapped to a dataset entry.
- Only a pre-defined number of bit errors is accepted and declared as valid ID.

To increase the probability of correct ID assignment the number of output bits can be raised. In Figure 3.3 this relation is shown. In both figures, the mean bit error rate HD_{intra} (definition of

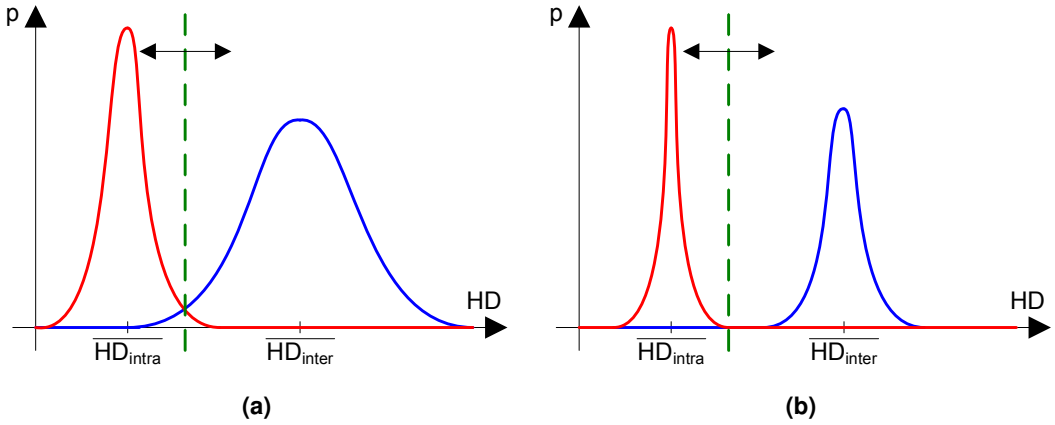


Figure 3.3.: (a) Small number of PUF cells; (b) Large number of PUF cells.

the Hamming Distance HD see Section 4.2.1) and the mean inter-chip Hamming distance HD_{inter} are identical. Due to the larger number of PUF cells, the number of output bits in Figure 3.3b is

higher than in Figure 3.3a. Thus the variance σ^2 is smaller than in Figure 3.3a. The green-doted line shows the maximal allowed Hamming distance between the entry in the database and the actual measured PUF value. If HD_{intra} exceeds this values, an erroneous assignment occurs. It can be easily seen that in the case of Figure 3.3a the probability p of an error is much higher than in the case of Figure 3.3b.

A more mathematical explanation of PUF-based identification is given in Section 4.4.7 where the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) are used to specify the quality of a PUF with respect to identification purposes. After specification the number n of output bits can be determined to fulfill pre-defined requirements. For example, the maximal allowed error $HD_{MAX}$is defined to be at the point where FAR=FRR. Furthermore, it is assumed that $HD_{intra}= 5\%$ and $HD_{inter}= 50\%$. The variances σ_{inter} and σ_{intra} are calculated by using $\sigma = \sqrt{\frac{p(1-p}{n}})$. The relation between n, FAR, and FRR is shown in Figure 3.4.

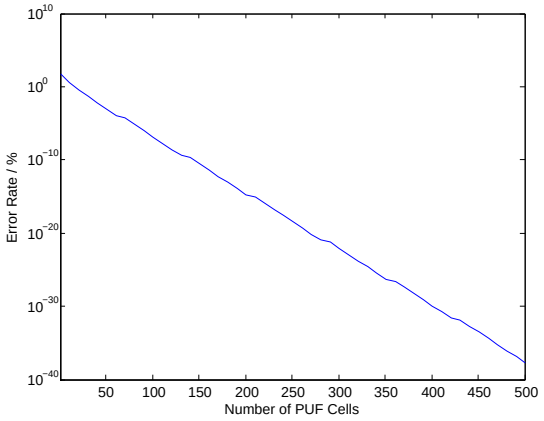


Figure 3.4.: Relation between n, FAR, and FRR.

3.2. Key Generation

A PUF can be used to generate a key for cryptographic purposes [64, 134]. For cryptographic purposes it is essential that the generated key is always exactly the same for the same PUF. Since PUFs always produce bit errors at their output, error correction has to be established. Thus, error correction is one of the major topics concerning PUFs. ECCs and approaches to reduce the error rate during design are discussed in more detail in the Chapters 5 - 11. One example of an application is shown in Figure 3.5.

Helper data are necessary for error correction coding. Helper data may be stored on the PUF chip itself. To do so, an NVM is required. This may be no option since NVMs are costly. The data can be stored externally as an alternative, for example on a server. During an enrollment phase these data are generated within the chip and sent to the database. If the PUF is read out the next time, the helper data are sent to the chip. To be able to find the right entry inside the database, identification/authentication procedures are necessary. Identification and authentication are discussed in greater detail in 3.1 and 3.3. The difference between external and internal data storing is depicted in Figure 3.6. Figure 3.6a shows the case when ID generation is done by encrypting a 0-vector. In the approach of Figure 3.6b, an additional block is required to identify the chip. This is realized by a second PUF block.

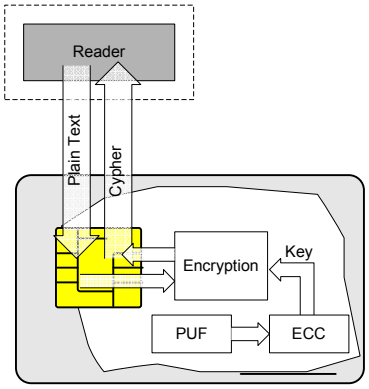


Figure 3.5.: Basic concept of encryption with PUFs.

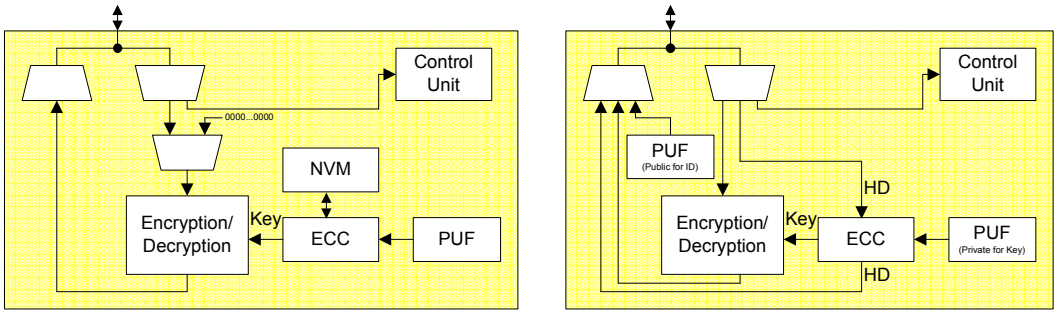


Figure 3.6.: (a) Helper data stored on the chip; (b) Helper data stored on a external database.

3.3. Authentication

Authentication is the procedure to verify a claimed ID of an entity. One prominent authentication concept is based on challenge response pairs. In information theory an authentication protocol, which uses this kind of technology, is the challenge-handshake authentication protocol (CHAP). In RFC 1994 "PPP Challenge Handshake Authentication Protocol (CHAP)" [131] this protocol is described in detail.

3.3.1. Authentication With Challenge-Response Pairs and PUFs

The idea behind authentication using challenge-response pairs is to send some kind of question to the entity that is to be authenticated [56]. The entity returns the answer to this question, i.e. the response. The sender verifies this answer by comparing it to the expected one in a database. If the answers match, the entity is authenticated. The procedure is shown in Figure 3.7.

If PUFs are used to generate the response, errors may occur even if the claimed identity is correct. Two response-generation scenarios exist: either the response is generated by hardware or by software. In the case of hardware-based generation the PUF cells must be able to generate different outputs depending on the actual input. Beside error correction, error tolerant systems as shown in Section 3.1 can be used. If the responses are produced by using software, an error-free ID is necessary. In this case error correction has to be done. An overview over the classification is depicted in Figure 3.8.

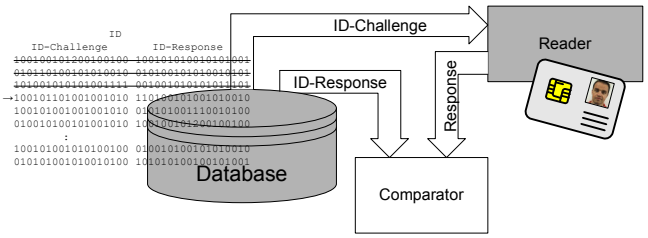


Figure 3.7.: Principle of authentication using challenge-response pairs.

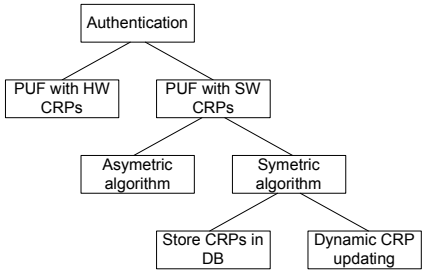


Figure 3.8.: Classification of authentication with PUFs.

General Security Considerations

One important pre-condition to secure CRP-based authentication is the availability of a sufficient number of CRPs. This means that the number of bits per challenge must be high enough so that an adversary cannot read out all possible combination of CRPs. In such a case, an adversary could clone the chip. Typically, today 128 challenge bits or even more are believed to provide protection against brute-force attacks. Furthermore, the challenges have to be random numbers. If an adversary can guess a future challenges, he or she can read out only the expected challenges and clone a chip partly.

Life Cycle of the CRP Approach

In the Figure 3.9 a life cycle of the authentication process with CRPs is shown. During the first phase, which is called enrollment phase, the chip has to be registered in a database. In cases where no NVM is available on the chip and error correction is needed, helper data has to be generated on the chip and transmitted to the database. Furthermore, an ID of the chip has to be stored in the database. Later on, this ID is going to be proved during the authentication process. Afterwards, the CRP generation phase starts. During that phase, a certain number of CRPs will be generated and stored in the database. The number of stored pairs depends on the estimated number of authentication processes of the chip. The last phase is the phase of the authentication itself. During this phase a claimed ID is verified by the system. A new CRP is necessary for each authentication procedure. This is important in order to make the reuse of recorded challenge-response pairs impossible. Even though, this approach makes the system very hard to attack, the limited number of available CRPs are the limiting factor in the whole authentication process. If all CRPs are used and the database is 'empty', either new CRPs have to be generated or the entity cannot be authenticated in the future again. In such a case the CRP generation phase has to be repeated to allow further authentications. The generation of additional CRPs has to be done in a trusted and secure environment.

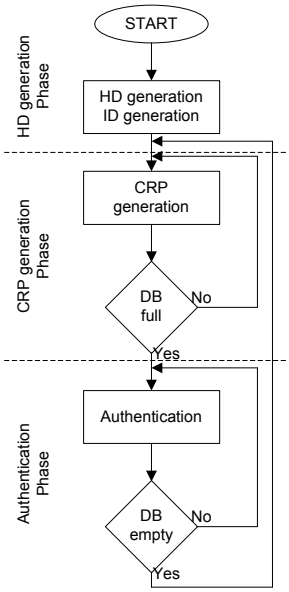


Figure 3.9.: Lifecycle of the authentication process with CRPs.

3.3.2. Authentication With Hardware-Based CRPs

A hardware-based CRP bases on PUF cells which output depends on an input to the PUF: $out = f(in)$. Such PUFs somehow work like hash functions. So, the same properties are important:

- Determinism: apart from possible noise at the output, a hash must always deliver the same output for the same input.
- Uniformity: each possible output value should be generated with the same probability.
- Predictability: the output of the PUF should not be predictable if the input or other input-output combinations are known.

If these conditions are fulfilled, a PUF can be assumed to be qualified for authentication purposes. The concept behind the hardware-based CRP generation is depicted in Figure 3.10. First of all,

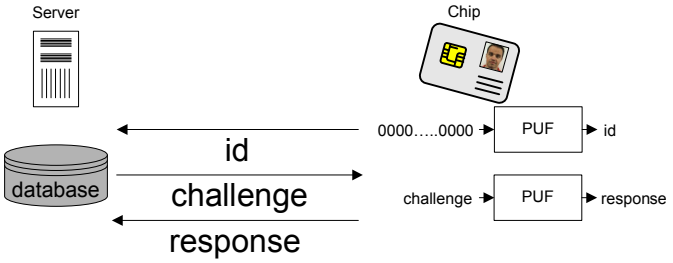


Figure 3.10.: Authentication with hardware-based CRP generation.

the server has to know the ID of the chip. For example, this can be done by reading out a pre-defined challenge-response pair: the zero-vector can be used as the input to the PUF. The generated response is defined as the ID of the chip. Usually, the returned ID is noisy. Therefore, the server has to compare the received ID with the IDs in the databases. If the Hamming distance to one of the IDs in the database is smaller than a certain pre-defined value, the chip is identified. After that, the server sends one challenge of the stored CRPs to the chip. The response is generated by the

PUF cell and returned to the server. If the response matches, the response of the stored CRP at least within a certain tolerance, the chip is authenticated.

Advantages (+) and Disadvantages (-):

+ No NVM is required.
+ Error correction is not required.
+ Simple approach.
- Hardware PUF with CRPs is hard to realize.
- PUFs are comparable big in size.
- A number of CRPs has to be stored on the server.
- CRPs have to be regenerated in trusted environment.

Different concepts of fitting PUF approaches are shown in Section 1.4.

3.3.3. Authentication With Software-Based CRPs

The basic idea behind this approach is that the challenge is mapped to a response by means of an encryption algorithm. In Figure 3.11, the basic concept is depicted. Different cryptographic

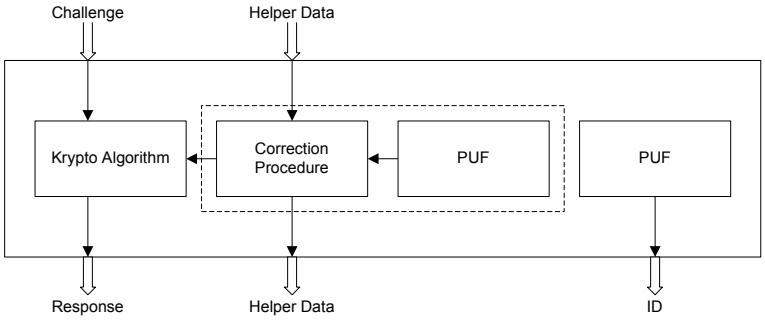


Figure 3.11.: Authentication with the software-based CRPs.

algorithms are suitable in general. An error free key is required to produce an identical output for all of this algorithms. Concerning PUFs this requirement implicates that error correction has to be done in any case. In the following text it is assumed that a stable key is returned from the PUF. The different procedures for stabilizing the output of a PUF are described in the later chapters.

First of all, the ID of the entity which has to be authenticated must be produced. This can be done in different ways:

- the ID is stored outside the chip and has to be entered in an extra step.
- the ID is stored on the chip.
- the ID is generated using a second PUF on the chip.

Public-key and symmetric-key cryptography can be used to generate the response. Both procedures are explained in the following sections.

Public-Key Cryptography

When using public-key cryptography to generate CRPs the incoming challenge is encrypted with the help of a private key on the chip. The result is returned as the response. Afterwards, the response can be decrypted at the server by means of a public key. If the decrypted response matches the challenge, the chip is authenticated.

To allow the authentication of an entity, during an initial enrollment phase the chip is registered in the system. Later on, during the authentication phase, the ID of the chip can be verified. The whole procedure is shown in Figure 3.12 and described in the following paragraphs.

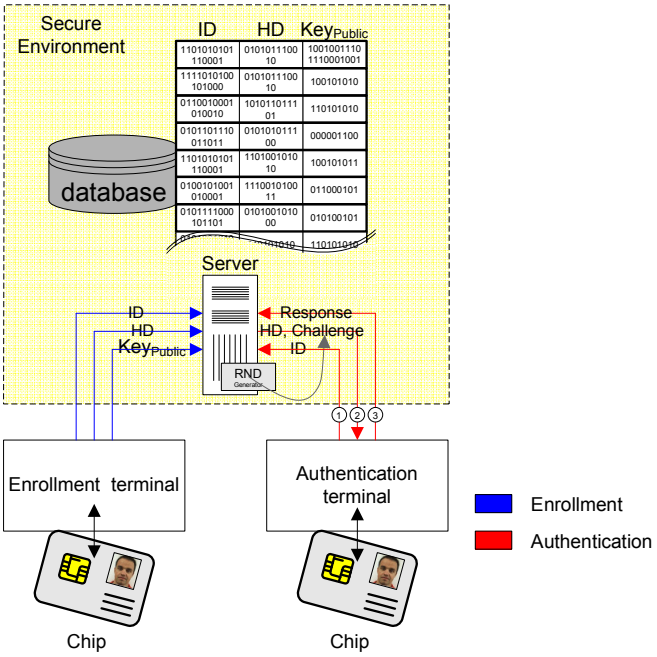


Figure 3.12.: Software-based CRPs with public-key cryptography.

Enrollment Phase:

- An ID of the chip has to be generated and stored on the server. The ID is used to find the right dataset in the database and also to identify the chip later.
- The PUF generates output data and helper data for later error correction. The helper data (HD) can be stored either externally on a server or on an internal NVM, if available. In Figure 3.12 the HD is transferred to the server and entered into the database.
- The public and the private keys are generated from the PUF's output. The generated public key is sent to the server where it is stored in the database. The generation of the public key has to be done on the chip for security reasons. The output of the PUF should never leave the chip.

Authentication Phase:

- As a first step, the ID is sent to the server.
- Afterwards, the server sends the HD to the chip, if necessary. Additionally, a random number is generated on the server. This number is provided as a challenge to the chip.

- The PUF generates an error free output using the helper data.
- The chip generates the private key from the PUF output.
- The response is generated by the chip. This is done by encrypting the received challenge by means of the private key.
- The response is sent back to the server.
- The response is decrypted by using the public key.
- If the data match, the chip is authenticated.

A block diagram of the chip is depicted in Figure 3.13.

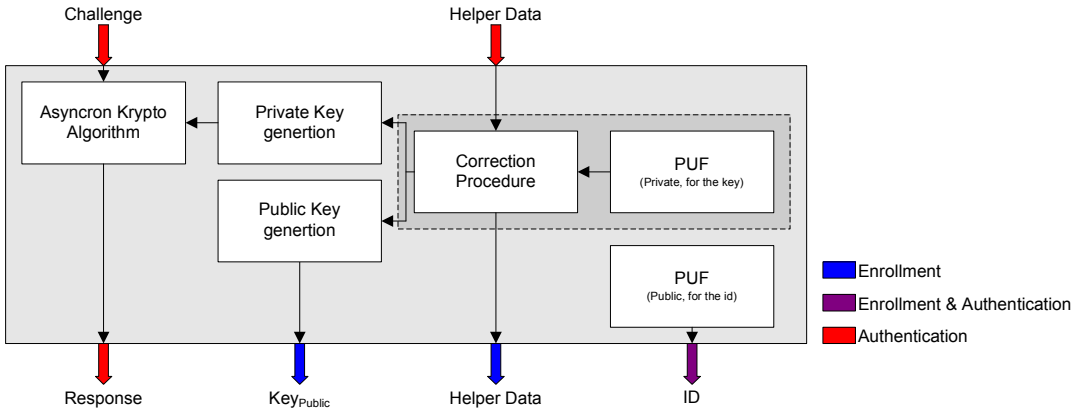


Figure 3.13.: Block diagram of a chip with public key based CRP generation.

Advantages (+) and Disadvantages (-)

+ PUF can be realized simply and small.
+ CRPs do not have to be stored on a server. CRPs do not have to be generated during enrollment phase.
+ Security requirements are not very restrictive.
+ Easy distribution of the database (to different authentication stations).
- Public key cryptography is complex in terms of the computational effort. Especially on chips, where resources (time, hardware, energy) are limited, this can be a serious problem.
- Lots of PUF bits are necessary to reach an acceptable security level.

Symmetric-Key Algorithm

There are different ways to use symmetric-key algorithms for chip authentication. Approaches exist where the secret of the PUF leaves the chip. This is not recommended since the intrinsic generation of the secret is one of the key features of PUFs and should be used to increase the security level. Due to that reason such approaches are not taken into account here.

There are different advantages of symmetric-key cryptography compared to asymmetric approaches. One of these is their smaller computational effort. Furthermore, the used key length is smaller. The typical key length of the asymmetric RSA algorithm is 1024 bit, for example. The key length of the AES (symmetric) is between 128 bit and 256 bit. Thus the chip design is less complex compared to the public-key approaches.

In this chapter two different concepts are presented. In the first concept a list of CRPs is saved on a server. The idea behind the second approach is to update the CRPs dynamically.

List of CRPs in a Database

The whole procedure can be seen in the Figure 3.14. It can be divided into three phases: during the enrollment phase the chip is registered in the database. During the CRP generation phase the CRP database is (re)filled. During the authentication phase, the chips are authenticated using the data from the database. The three phases are described more detailed below.

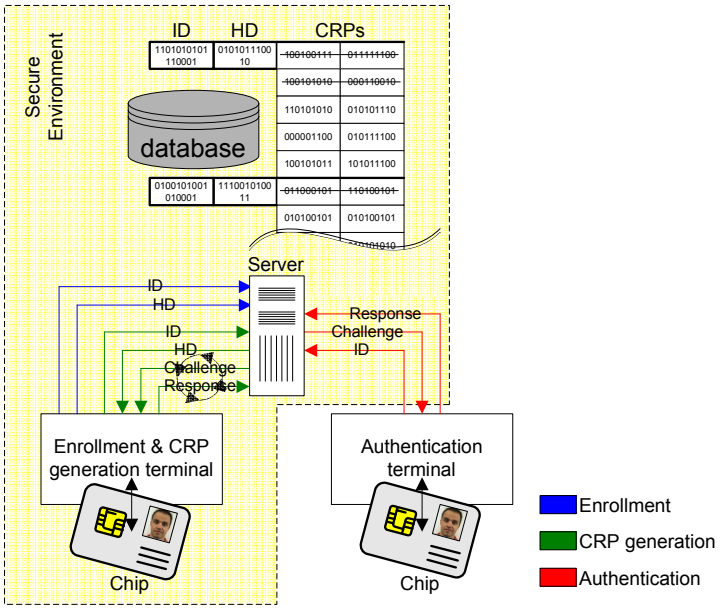


Figure 3.14.: Devices in a CRP approach.

Enrollment Phase: This phase is shown by the blue lines in the Figure 3.14.

- The PUF is generating an output and the necessary helper data. This helper data can be stored externally on a server or on an internal NVM.
- An ID of the chip has also to be stored on the server. The generation of the ID can be done using a second PUF or a pre-defined challenge (zero-vector).

CRP Generation Phase: This phase is shown by the green lines in Figure 3.14.

- Challenge-response pairs are generated in a secure environment. Usually the server generates a challenge which is then sent to the chip. The chip returns the corresponding response.
- These CRPs are all stored in a database. For each authentication one CRP is used. Accordingly, the number of stored CRPs depends on the application and the frequency of the authentication.
- If all CRPs are used, the database can be refilled with new CRPs in the same way, again in a secure environment.

Authentication Phase: This phase is shown by the red lines in Figure 3.14.

- In the first step the server receives an ID from the chip.
- The server sends one of the unused challenges of the claimed ID to the chip.
- The chip generates a response by means of the internal key which was generated by the PUF. The response is sent to the server.
- If the stored response and the received response match, the chip is authenticated.

Popular algorithms for this approach are the AES (symmetric block cipher), MD5, and SHA-1 (both cryptographic hash functions) algorithm.

Advantages (+) and Disadvantages (-):

+ Algorithm is computationally less complex than the public-key approaches.
+ Simple implementation.
- An error correction algorithm is required.
- A number of CRPs has to be stored on the server.
- CRPs have to be regenerated when used up.

Dynamic CRPs Updating

The advantage of dynamic CRP updating is that only one CRP is stored in the database at once. A new CRP is generated during every read out. Thus, there is no limitation with respect to the number of authentications. The approach is depicted in Figure 3.15.

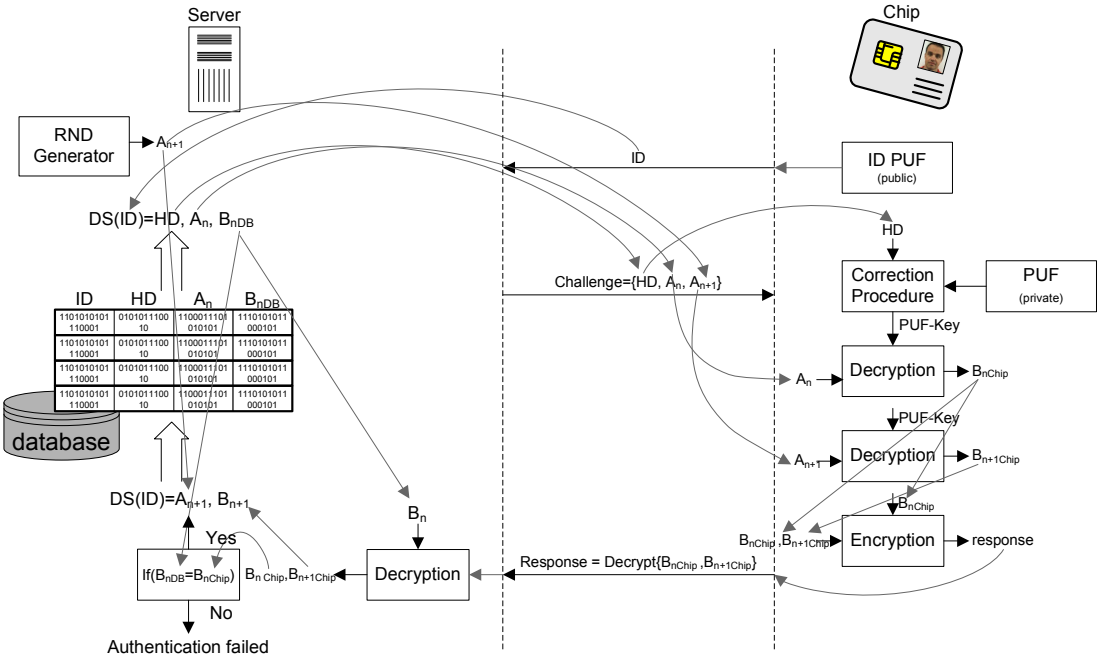


Figure 3.15.: Dynamic CRP updating: Authentication phase.

No special CRP generation phase is required. The enrollment phase is almost the same as in the preceding approach. Additionally, one CRP is stored in the database.

Enrollment Phase:

- The PUF is generating an output and the necessary helper data. This helper data can be stored on a server externally or on an internal NVM.
- An ID of the chip has also to be stored on the server. The generation of the ID can be done using a second PUF or a pre-defined challenge (zero-vector).
- In a secure environment one challenge response pair is generated. Since B_n must be available on both sides to encrypt/decrypt the responded data. The zero-vector can be used during the enrollment phase. The CRP will be stored in the database.

During the authentication phase some additional steps are required. A whole overview is given in Figure 3.15.

Authentication Phase:

- In the first step the server receives the ID from the chip.
- The server sends the helper data, actual challenge A_n and a new challenge A_{n+1} coming from a RNG to the chip.
- The chip generates a response by means of the internal key which was generated by the PUF. Both challenges are decrypted separately. After that, the two responses $B_{n,chip}$ and $B_{n+1,chip}$ are merged and encrypted with the first response $B_{n,chip}$. The encrypted data is sent to the server.
- At the server, the encryption key is known, since it is the response B_n to the first challenge A_n. So, the received data is decrypted using this response. If the data from the chip is correct, the decrypted data consists of the response to the first challenge $B_{n,chip}$ and the response two the second challenge $B_{n+1,chip}$. Hence, the chip is authenticated by the response to the first challenge. The response to the second challenge is stored on the server as the first challenge for the next authentication process.

A fitting algorithm for this approach is the symmetric block cipher AES.

Advantages (+) and Disadvantages (-):

+ Algorithm is computationally less complex than the public-key approaches.
+ No long list of CRPs has to be stored on the server.
- A correction algorithm is required.
- The whole approach is quite complex.

4. Testing and Specification of PUFs

by Christoph Boehm

When comparing different approaches of PUFs it is important to establish a set of parameters that measure the performance [14, 23, 60]. In this chapter an overview of important parameters and the ways to determine these parameters will be given.

4.1. Fundamentals

To analyze and compare PUF concepts, the type of PUF must be determined in the beginning. This is important in order not to compare apples with oranges. Some examples are listed below:

- common PUF
- PUF including pre-processing
- PUF including some kind of error correction

Furthermore, it is important to pre-define the environmental conditions in which the PUFs are tested. Especially, the temperature range influences the overall error rate a lot. Therefore, the temperature range has to be defined already during the specification of a PUF. In many cases, the enrollment phase of the PUF will be done at room temperature (25°C). If the PUFs are used in an industrial environment later on, the temperature may range from -40°C to +85°C. With respect to the enrollment condition, the temperature differences may reach +/-60°C. If the enrollment is done in industrial environment, the difference may be more than 120°C. The consequences of such differences are shown in Figure 4.1. In Figure 4.1a the enrollment is done at 25°C. In Figure 4.1b the enrollment is done at -45°C. It is obvious that due to parameter fluctuations the expected error rate in Figure 4.1b is higher over the whole temperature range than in Figure 4.1a. This behavior makes a good specification of the chips and also of the test conditions very important.

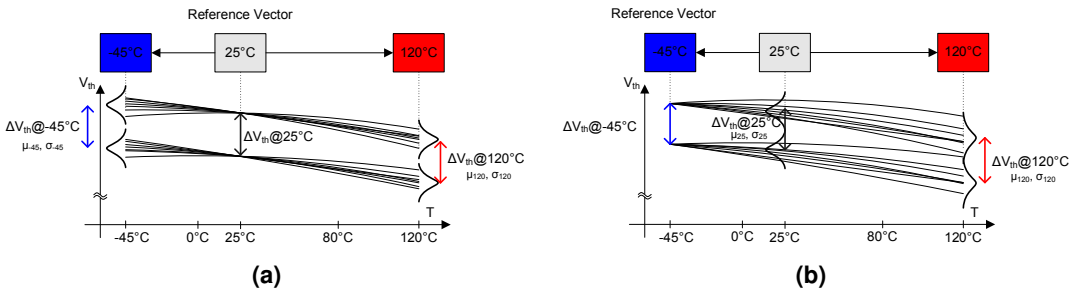


Figure 4.1.: Enrollment at different temperatures; (a) Enrollment at 25°C; (b) Enrollment at -45°C.

There are two alternatives how to test a PUF: the PUF can either be tested on the wafer or after packaging. The advantage of wafer testing is the huge amount of data that can be collected. Thus, precise statistics can be evaluated. Then again, if measured inside the package, the PUF shows exactly the behavior than later in the field. Furthermore, analysis can be performed in any lab

which makes things easier. At least fundamental functionality and corner parameters can be tested (e.g. power consumption).

4.2. Theory Basics

4.2.1. Hamming Distance

The Hamming distance (HD) is named after the mathematician Richard W. Hamming (1915 - 1998). The HD is a parameter of information theory which stands for the number of different elements of two strings of the same length. An example in matters of binary data is shown in Table 4.1.

Table 4.1.: Example of the Hamming distance.

String 1:	**0**	**0**	0	1	**0**	**1**	1	0	
String 2:	**1**	**1**	0	1	**1**	**0**	1	0	
XOR:	**1**	**1**	0	0	**1**	**1**	0	0	$\rightarrow \sum$=4

The differing bits are printed in bold type. By XORing the two stings summing up the result the Hamming distance of binary strings can be determined. Hence, in the example below, the HD is 4. This can be expressed as follows:

$$HD(String1, String2) = 4$$

The HD can also be expressed either as a ratio or in percent. In that case the result is divided by the string length (in this example the length is 8) and if necessary multiplied by 100 %:

$$HD(String1, String2) = 0.5 \; or \; 50\,\%$$

In the case of uniformly distributed bits the mean Hamming distance of two binary strings is 50 %. The Hamming distance of such strings is distributed binomially. The distribution of HD is often shown in a graph where the x-axis shows the probability of occurence and where the y-axis shows the frequnecy. An example is given in Figure 4.2b which shows a distribution of the HD of different strings. The mean HD is set to 8 % in this example.

4.2.2. Binomial Distribution

If the elements of bit strings are random variables, the HDs between different strings result in a binomial distribution. The binomial distribution is defined as

$$\binom{n}{k} p^k (1-p)^{n-k}, \tag{4.1}$$

where n is the number of bits in the bit string, k the number of occurring '1', and p the probability of occurrence of '1'.

If the number of bits is large, the normal distribution is a good approximation of the binomial distribution. The equations in 4.3 show the conversion[1] of a binomial distribution to a Gaussian distribution:

[1] Binom($n,\ p,\ q$) $\rightarrow$ Norm($\mu,\ \sigma$)

$$\begin{aligned} \mu &= np \\ \sigma^2 &= npq \end{aligned} \tag{4.2}$$

In some cases this conversion is quite useful. In Figure 4.2 two examples of binomial distributions for n=100 ($\mu = 50$, $\sigma = 5$; $\mu = 8$, $\sigma = 2.713$) are shown. The red curves show the appropriate Gaussian distributions.

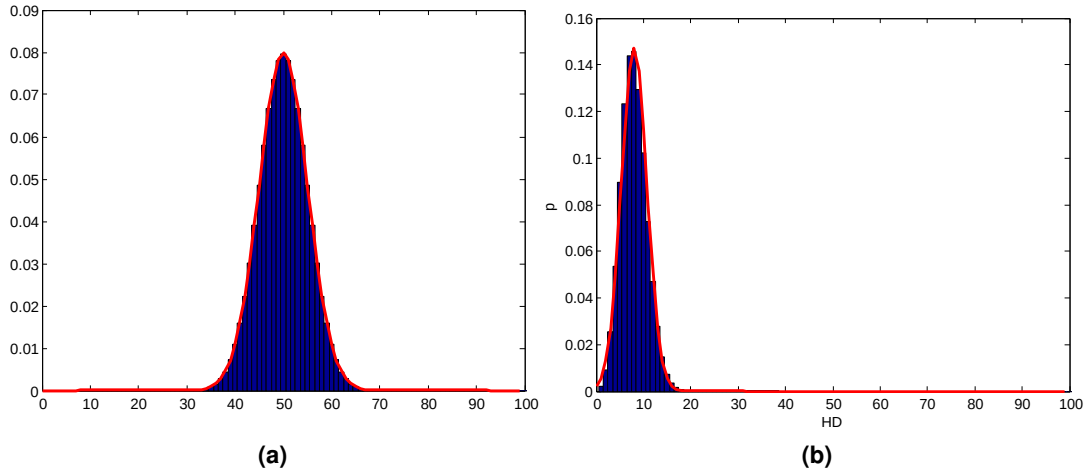


Figure 4.2.: (a) Binomial distribution: n=100, p=0.5, q=0.5 (blue bars); Gaussian distribution: μ=50, σ =5 (red curve); (b) Binomial distribution: n=100, p=0.08, q=0.92 (blue bars); Gaussian distribution: μ=8, σ =2.713 (red curve).

It is easy to determine the values p and q for a certain number N of binomial distributed datasets DS. n is the number of bits in each dataset. The mean of all bits μ_{all} in all datasets is determined in a first step:

$$\mu_{all} \approx \frac{\#\ of\ 1s\ in\ all\ DS}{n \cdot N} \tag{4.3}$$

p can be calculated from equation 4.3:

$$\begin{aligned} p &= \frac{\mu_{all}}{n} \\ q &= 1 - p \end{aligned} \tag{4.4}$$

4.3. Specification Parameters

To characterize the hardware properties of a PUF the following parameters are suggested: Mean value, error rate, correlation between bits, correlation between chips, and energy consumption. These parameters are explained detailed in the following sections.

Example

In order to help to explain the different parameters in detail a exemplary PUF test chip is used. 10 of the test chips were available, each of them containing 4096 PUF cells. Therefore, the number

Table 4.2.: Measurement settings.

Parameter	Min	Nom	Max
Temperature	-40°C	25°C	120°C
V_{DD}	1.25 V	1.35 V	1.45 V
I_{BIAS}	2 µA	16 µA	30 µA

of output bits is 4096. Each chip was read out 100 times to receive accurate results. To be able to determine all parameters, the chip was measured under the conditions shown in Table 4.2.

The nominal settings were used to determine the reference output. This is normally done during the enrollment phase. The other measurement results are compared later to the reference vector to estimate the error rate.

The Ideal PUF

To get a feeling for the properties a PUF should have an ideal PUF is described in the following section. The properties are shown in Table 4.3. It shows the ideal values for the mean value, the

Table 4.3.: Ideal PUF.

Property	Identifier	Value
Mean value	μ	0.5, $\sigma = \frac{\sqrt{N}}{2}$
Error rate	$HD_{intra(120C)}$	0 %
Corr. between bits	R_{xx}	0, $\sigma = \frac{1}{2\sqrt{N}}$
Corr. between chips	HD_{inter}	50, % $\sigma = 100\% \frac{\sqrt{N}}{2}$
Power consumption	E/bit	0pJ/bit

correlation between bits and the correlation between chips. Nevertheless, it is impossible to reach a bit error rate of 0 %[2] and an energy consumption of 0 pJ/bit.

4.3.1. Mean Value

As mentioned above, the outcome of a PUF must not be predictable. Thus, the output values '1' and '0' should be distributed equally. Looking at the output of the chip, the arithmetical mean value $\bar{x}$ of the 4096 cells should be around 0.5, and $\bar{x}$ is defined as

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{4.5}$$

where n is the number of cells and x_i the actual output. Since the results of the 10 test chips are distributed binomially, the different $\bar{x}$ are placed on top of the probability density function (pdf) of a binomial distribution (blue curve). The result can be seen in Figure 4.3. All $\bar{x}$ are situated around 0.5. This is the expected result. However, a small bias towards '1' can be observed. This bias lies within a statistical accurate range. Thus, the bias does not effect the predictability much. The amount of bias that can be accepted depends strongly on the application what the data are used for.

[2] without any further processing like pre-processing and post processing approaches explained in later chapters.

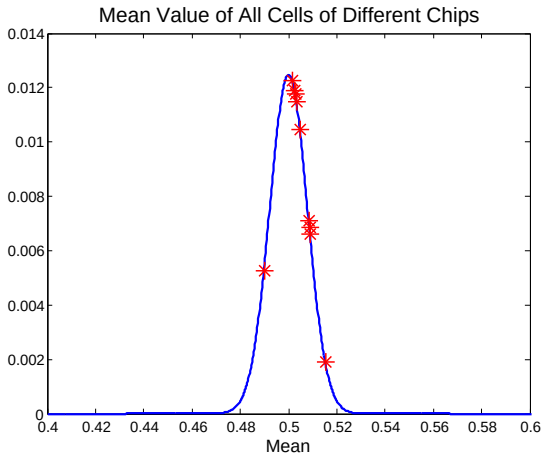


Figure 4.3.: Measured mean values on top of a binomial distribution.

The confidential interval (CI) can be used to determine whether the available data lie within an expected range. For example, if a confidence interval of 95 % is considered and the normal approximation interval is used, CI can be determined as follows:

$$CI = p \pm z\sqrt{\frac{p(1-p)}{n}} \tag{4.6}$$

Where p is the probability of occurrence, n is the number of output bits and z is the $1 - \alpha/2$ percentile. α is the error percentile. In this case $\alpha = 5\,\%$, therefore $1 - \alpha/2 = 0.975$, and $z(0.975) = 1.96$. According to $p = 0.5$, the Equation 4.6 becomes

$$CI = 0.5 \pm 1.96\sqrt{\frac{0.25}{4096}} = 0.4846 \text{ and } 0.5154. \tag{4.7}$$

Therefore, all $\bar{x}$ of the test chips lie within that confidential interval.

4.3.2. Error Rate

The output of a PUF should stay constant within a pre-defined region of operation. Errors occur, if bits change their output values between different runs. The error rate or bit error rate (BER) can be measured using the Hamming distance (HD) which is explained in Section 4.2.1. To determine the error rate of a PUF the HD of a reference vector and the actual output is determined. In this case the Hamming distance is called intra-chip HD (HD_{intra}). At an ideal PUF this value is 0. Due to the influence of noise, temperature effects, power supply variation, and variation of the bias current this value exceeds 0.

Concerning the test chip, the mean value of HD_{intra} of 100 runs at room temperature is 1.13 %. Figure 4.4a shows the HD_{intra} of the 100 runs with respect to the reference vector. In this case the reference vector is defined as the first output vactor of the 100 runs. As soon as the temperature changes, HD_{intra} increases. In the range from -40°C to 120°C the error rate reaches up to 5 % if the reference vector is defined at 25°C. This is a rather high value compared to HD_{intra} of around 1 % at room temperature. The error rate at different temperatures is shown in Figure 4.4b. There is nearly a linear dependence between error rate and temperature change. At -40°C there is a maximal error rate of $HD_{intra(-40°C)}$=4.1748. At 120°C, the maximal error rate $HD_{intra(120°C)}$=4.8828. This value increases further if the reference vector is not defined at room temperature. For example, if the reference vector is defined at -40°C, $HD_{intra(120°C)}$=8,2 %. Due to this fact, the reference

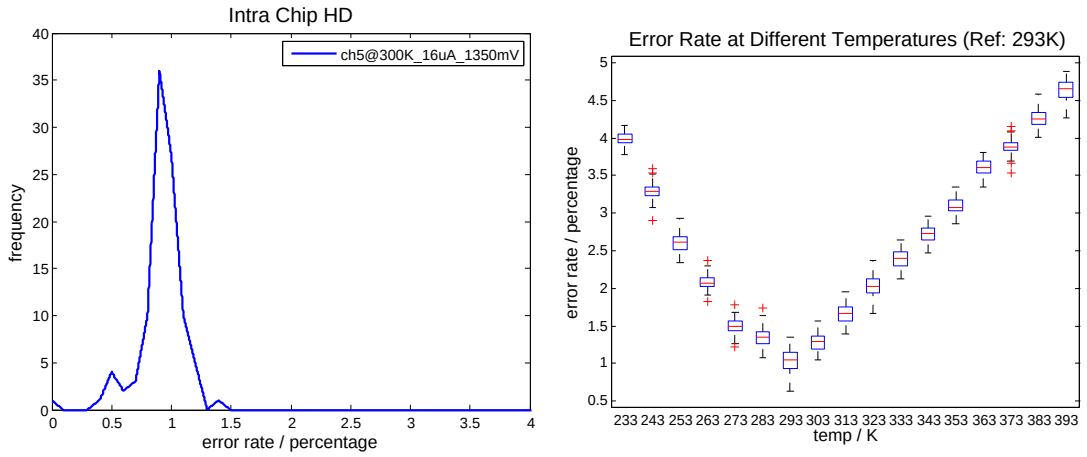


Figure 4.4.: Error rate of PUFs: (a) Intra-chip Hamming distance HD_{intra} at room temperature; (b) HD_{intra} at different temperatures.

vector has to be measured in a well defined environment at nominal conditions. This should already be done during the functional tests of the chip after production. The reason for the high temperature dependence is the mismatch between the temperature coefficients of the involved transistors. Due to this mismatch it can happen that transistors behave different with respect to each other at different temperatures. The effect can be seen in Figure 4.5. It shows the temperature behavior of the two defining elements. At crossings the behavior of the PUF changes and error occur. It turned out that temperature changes are the dominant source of errors in PUF cells [118, 137, 138].

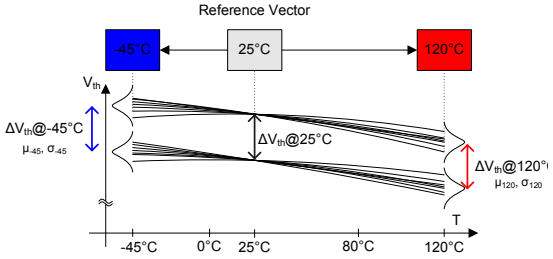


Figure 4.5.: Influence of temperature on the threshold voltage of transistors.

The test chip was also measured at supply voltages of 1.25 V and 1.45 V as well as at bias currents (I_{BIAS}) between 2 µA and 30 µA. No significant increase of errors could be observed.

The error rate is a significant measure for PUFs. It is a crucial parameter in the selection of suitable error correction codes (see Chapter 5).

4.3.3. Correlation Between Bits

Neighboring cells must not influence each other to avoid security issues. Correlation between cells would reduce the number of possible combinations and thus would increase the risk of successful brute-force attacks. In order to check whether correlation exists in the test chip, the auto-correlation function R_{xx} is used:

$$R_{xx}(j) = \sum_{n} x_n x_{n-j}, \tag{4.8}$$

where R_{xx} is evaluated at lag j. For $R_{xx}(j) = 1$ and $R_{xx}(j) = -1$ the data at lag j is completely correlated. For $R_{xx}(j) = 0$, the data is completely uncorrelated. The measurement result of the test chip is shown in Figure 4.6. The data is correlated only at lag $j = 0$. The correlation is

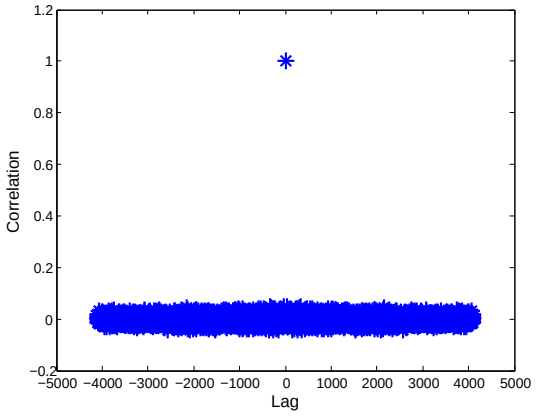


Figure 4.6.: Auto-correlation of the PUF.

negligible for lags other than 0 and thus, the test chip fulfills the requirements.

4.3.4. Correlation Between Chips

If the layout is not done carefully, it may happen that the output of the cells depends on the position in the layout. If this is the case, specific cells tend to a certain value. This effect also leads to security issues since knowing the output of one chip makes it easier to guess the output of other chips. As in the case of the error rate, the correlation between different chips can be determined using the Hamming distance. Under these circumstances the HD is called *inter-chip HD* (HD_{inter}) and should be around 50 %. If the mean HD_{inter} of all combinations of chips is 50 %, no correlation between the chips exist. The undelying distribution is the binomial distribution.

The measured HD_{inter} of the 10 test chips is shown in Figure 4.7. Again, the results are placed

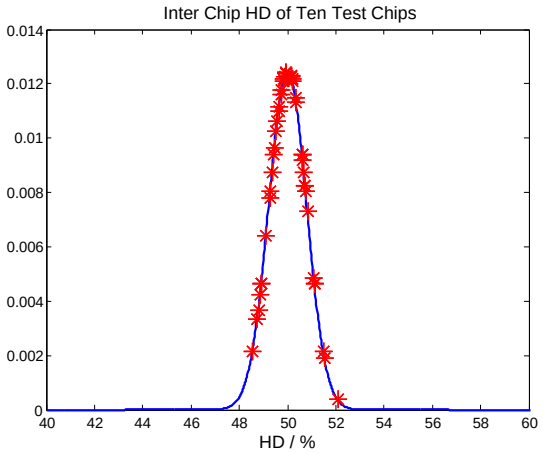


Figure 4.7.: Inter-chip Hamming distance of the 10 test chips on binomial distribution.

on a binomial distribution. The mean value of all HD_{inter} is 50.0271 % which is near to the optimal value of 50 %.

HD_{inter} can also be used to determine whether correlation between bits within one chip occurs. If some of the cells tend to a certain value inside the same chip, the distribution of HD_{inter} looks as in Figure 4.8a. If some of cells correlate within one chip, the distribution of HD_{inter} looks as in Figure 4.8b.

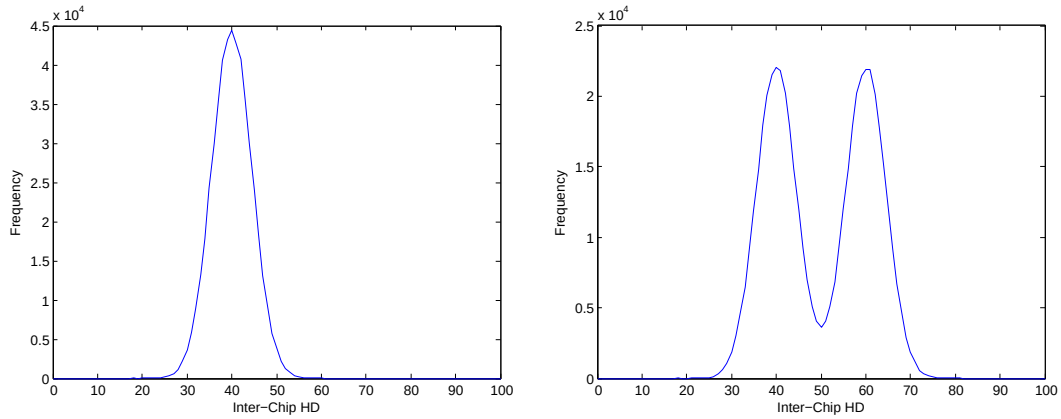


Figure 4.8.: HD_{inter}: (a) 20 % of the cells of a chip tend to '1'; (b) 20 % of the cells of a chip correlate.

4.3.5. Power and Energy Consumption

If the PUF is used in devices like RFID tags, power consumption plays an important role. But also energy consumption is often of interest. In the case of the exemplary test chip the average current consumption is around 290 µA. For V_{DD}= 1.35 V the power gets

$$P = 290\,\mu A * 1.35V = 391.5\,\mu W \tag{4.9}$$

Using a clock frequency of 1 Mhz and one bit output per cycle, the energy consumption per cell E_{bit} gets

$$E_{bit} = 391.5\mu W * 1\frac{\mu s}{bit} = 391.5\frac{pJ}{bit}. \tag{4.10}$$

Compared to other approaches this is a rather high value. A minimum value of 1.6 $\frac{\text{pJ}}{\text{bit}}$ was published in [138]. Since energy consumption was no design criterion, there is still room for improvement. One simple example would be to increase the clock frequency.

If the power consumption is a concern, a way to distribute the energy over time is shown in Figure 4.9. The difference of the power consumption between parallel and serial read-out is depicted. The integrated energy[3] is almost the same. In Figure 4.9a the cells are read-out in parallel at the same time. In this case, the cells can be read out very fast. The drawback of this approach is the peak in the current consumption. The whole energy for the evaluation has to be available during a short time. In Figure 4.9b the read-out is done serially (the read-out of three cells is depicted). In this case the peak energy consumption is much smaller but occurs several times. The total energy consumption is stretched to a longer time span. The drawback of this approach is the increase in evaluation time. The serial read-out is useful for applications of limited maximal power.

[3]Energy is the area under the I/t curve multiplied by the voltage $E = \int uidt$

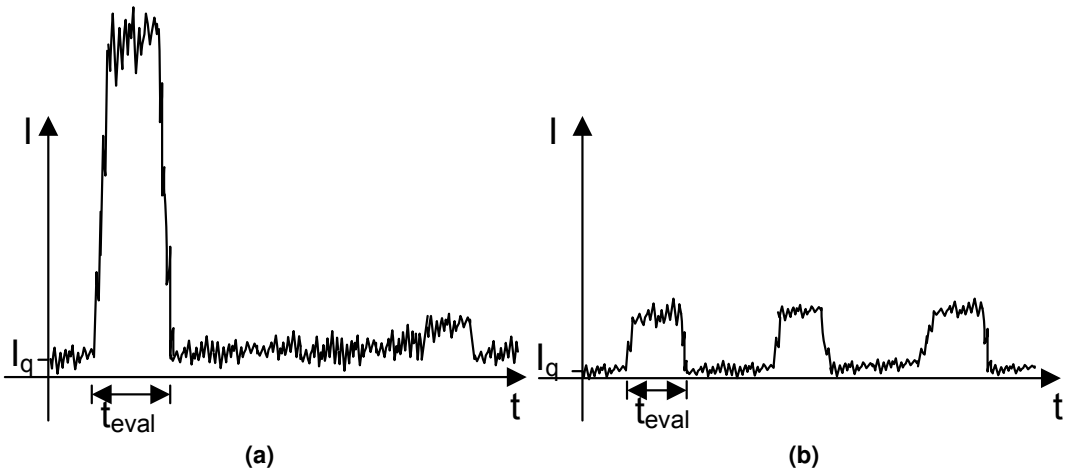


Figure 4.9.: Power consumption: (a) Parallel read-out; (b) Serial read-out.

4.4. Alternatively Exploitable Properties

The following section deals with alternative methods and additional properties to characterize a PUF chip.

4.4.1. Size

The size the PUF can be characterized by area per bit $\left(\frac{\text{m}^2}{\text{bit}}\right)$. Since the size depends on the technology strongly, a universal area specification is beneficial. In this case the area is expressed in $\frac{F^2}{\text{bit}}$, where F^2 is the minimum features size. It is the square of the minimal channel length. F^2=8100 nm^2 in a 90 nm process. As an example, a common 6T-SRAM cell has a F^2 of 140 [25].

4.4.2. Correlation Between Chips: Alternative Method

The method introduced in Section 4.3.4 (HD_{inter}) is the preferred approach to detect the correlation between chips. However, by using HD_{inter}single bits that provide the same output on each of the chips cannot be detected. If there is the possibility that certain cells always produce the same output (e.g. due to layout issues) another approach has to be chosen. One feasible approach is to calculate each cell's mean over all available chips:

$$\mu(j) = \frac{1}{N}\sum_{i=1}^{N} x_{ji}, \tag{4.11}$$

where N is the total number of chips, and j is the cell identifier. To get adequate results, a sufficient number of chips has to be measured.

4.4.3. Speed

Another criterion for evaluating PUFs is speed. Speed may be a concern in some circuits. If power consumption plays a role in it, some circuits may exceed some timing constraints before providing the output. The measurement unit for speed is $\frac{\text{bits}}{\text{s}}$. How to determine the read-out speed depends on the cells that are read out in parallel or serially. In the case of the exemplary test chip the read-out frequency is 1 MHz. Since the cells are read out serially, the the read-out of the 4096 cells takes $t = \frac{4096}{1\text{MHz}}$=4.096 µs. Thus, the speed is 10E6 $\frac{\text{bits}}{\text{s}}$.

4.4.4. Error Rate from Temperature Shifts

Looking at Figure 4.4b an almost linear relationship between temperature and the error rate can be observed. If a linear relationship is assumed, the error rate (e_{temp}), caused by temperature shifts, can be expressed as follows:

$$e_{temp} = \frac{E_T - E_N}{\Delta T}, \tag{4.12}$$

where E_T is the error rate at temperature T, E_N is the error rate at the nominal temperature, and $\Delta T = T - N$. The measurement unit is $[\frac{\%}{°C}]$.

In the case of the exemplary test chip, $E_N = 1.13\,\%$, $E_T = 4.64\,\%$, the nominal temperature N is 25°C and the actual temperature T is 120°C. Thus $\Delta T = 95°C$:

$$e_{temp} = \frac{4.64\,\% - 1.13\,\%}{95°C} = 0.04\ \frac{\%}{°C} \tag{4.13}$$

e_{temp} helps to estimate the error rates at certain temperatures from error measurements at two different temperatures.

4.4.5. Error Rate: Variations in Current and Voltage

Additionally to the temperature dependence of the PUFs' output, it may also depend on variations of the supply voltage or of bias currents. The influence of these parameters, if they do exist at all, depends strongly on the PUF circuit. In the case of the exemplary PUF, changes in both parameters effect the output of the PUF. The result can be seen in Figure 4.10: the current is varied between 12 µA and 20 µA, the voltage was set to 1.1 V and 1.5 V. The reference was defined in I_{BIAS}=16 µA and 1.35 V.

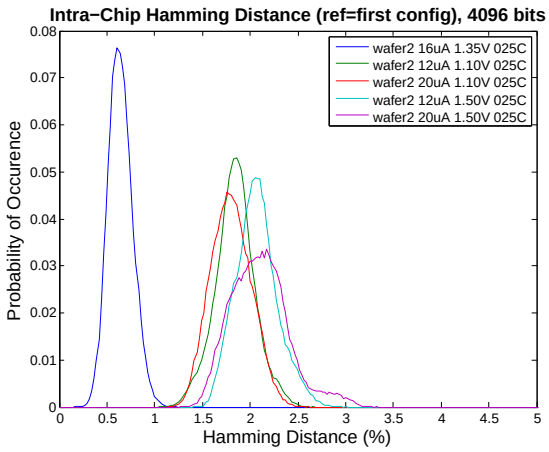


Figure 4.10.: Influence of different currents and voltages.

Figure 4.10 shows that changes in the supply voltage and the bias currents have considerable effects on the output. The error rate increases to around 2 %. Concerning PUF circuits it is especially important that changes in the operation point are kept as small as possible. To stabilize the output, PUFs should work under the same conditions as during the reference vector read-out.

4.4.6. Correlation Between Bits: Block Analysis

In some PUF concepts special structures in the layout of cells or shared parts make further correlation analysis necessary. As an example, a PUF including a shared sense amplifier is depicted in

Figure 4.11. It shows N PUF cells that share one sensing circuit. In such a case a mismatch in

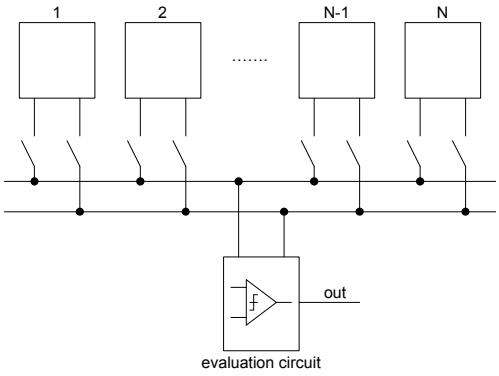


Figure 4.11.: Cells with shared sense amplifier circuitry.

the shared unit influences the output of all the cells. This effect can be seen in Figure 4.12. The

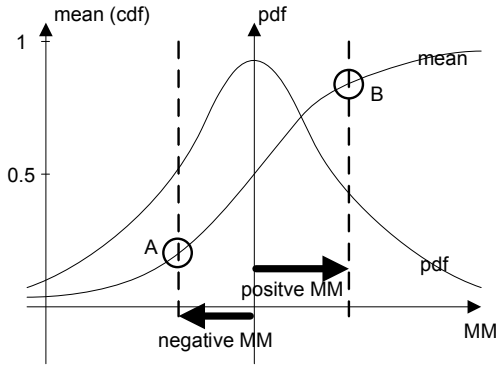


Figure 4.12.: Impact of a bias on the mean output value. The CDF is placed on top of the PDF.

influence of a negative mismatch (MM) and the influence of a positive mismatch is shown in it. In the nominal case the mean output equals 0.5. If a negative MM is introduced (A), the mean output is reduced. If a positive mismatch is introduced (B), the mean output is increased. Therefore, there are more zeros at the output. In practice, a bias on the output is a security issue since the number of likely combinations reduces.

The following measurement concept can be used to find biases on the output due to biases in parts of the circuit:

If N cells are connected to one sense amplifier, the whole output is also grouped into blocks of N. Afterwards, the mean of each block is determined. Since there is a binomial distribution for the mean values of such blocks, the results can be placed on top of that distribution. An example is shown in Figure 4.13. The upper and the lower curves depict the CI of 3σ (99.7 %). The effect of biases can be seen clearly. Near the mean the values tend to lie bellow the expected value. In return, for the values at the outer regions, the measured values tend to exceed the expectation values.

To get numerical results, it is reasonable to approximate the binomial distribution by a normal distribution. Hence, the expected standard deviation σ_{ideal} and the real standard deviation σ_{real} can be estimated. In the example ($N = 16$) the two values are $\sigma_{ideal} = 2$ and $\sigma_{real} = 2.488$ (maximum likelihood estimation).

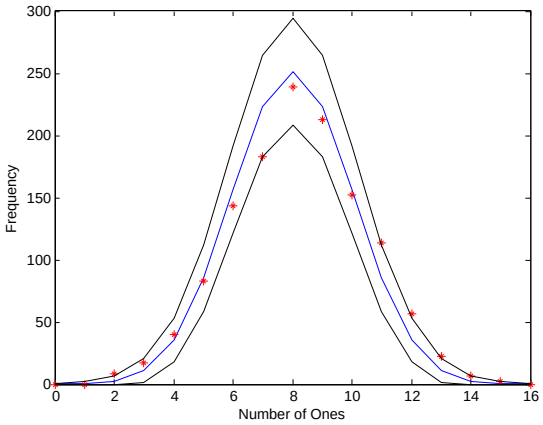


Figure 4.13.: Distribution of mean of blocks of 16 cells.

4.4.7. FAR and FRR

The false acceptance rate (FAR) and the false rejection rate (FRR) help to describe the identification performance of a system. When PUFs are used, the error rate of the chips and the number of the output bits define that performance. This means that FAR and FRR also define the overall performance of the PUF.

Example: the IDs of chips are stored in a database on a server. During the identification phase, a chip sends its ID to the server. If the Hamming distance to a entry in the database is below an pre-defined value, the received ID is assigned to that entry by the server. Errors can happen either, if the chip is not assigned to the ID (false rejection) or, if an unknown chip is assigned to an ID (false acceptance). Different measures are defined in literature: in [83] the false alarm rate (FAR) and the false detection rate (FDR) are being used. In [94] the false acceptance rate (FAR) and the false rejection rate (FRR) are utilized which are also the common measures in biometrics. Therefore FAR and FRR are preferred in this work, too. In Figure 4.14 the two parameters are depicted.

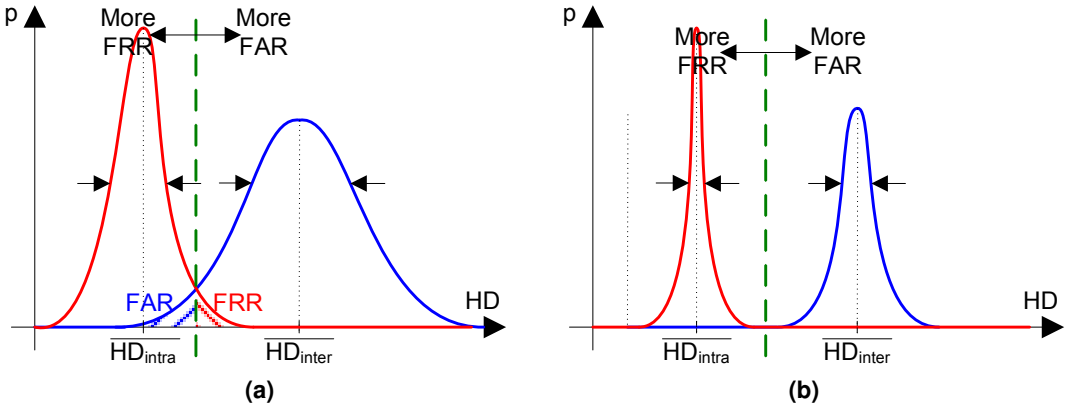


Figure 4.14.: False acceptance rate (FAR) and false rejection rate (FRR): (a) Small number of PUF cells; (b) Large number of PUF cells.

The green doted line depicts the maximal number of erroneous bits (HD_{MAX}) which is still allowed to accept the received ID. If HD_{MAX} is large, the FRR is low but number of false acceptances gets high. If the FAR is high, the change to accept any ID (e.g. sent by an attacker) gets

higher. And vice versa: if HD_{MAX} is chosen small, fake IDs may not be accepted but the number of rejections increases, too.

The two measures are defined mathematically as follows:

- **False Acceptance Rate:**
$$FAR = \frac{NFA}{NAA} * 100\,\%, \tag{4.14}$$
where NFA is the number of false accepted attempts and NAA is the number of attacker attempts.

- **False Rejection Rate:**
$$FRR = \frac{NFR}{NIA} * 100\,\%, \tag{4.15}$$
where NFR is the number of false rejections and NIA is the number of identification attempts.

The relationship between the FRR and FAR is shown in Figure 4.15. A trade-off between FRR

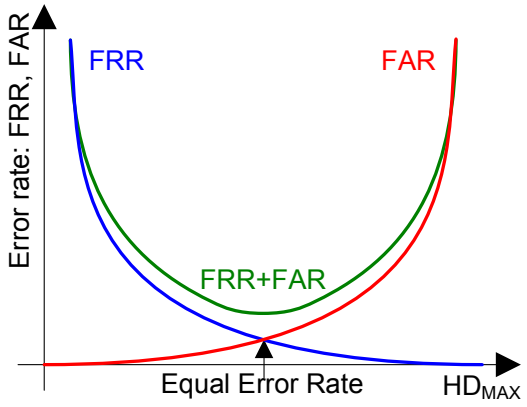


Figure 4.15.: Relationship between FRR and FAR.

and FAR has to be found depending on the application.

It is possible to determine the FRR and the FAR statistically. If the distributions in Figure 4.14 are assumed to be Gaussian, the following parameter can be derived: $HD_{intra} = \mu_{intra}$, σ_{intra}, $HD_{inter} = \mu_{inter}$, σ_{inter} and HD_{MAX}.

- FRR can be expressed as follows:
$$FRR = \frac{1}{\sigma\sqrt{2\pi}} \int_{HD_{MAX}}^{\infty} e^{-\frac{1}{2}\left(\frac{x-\mu_{intra}}{\sigma_{intra}}\right)^2} dx * 100\% \tag{4.16}$$

- For one ID in the database the FRR can be expressed as follows:
$$FAR = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{HD_{MAX}} e^{-\frac{1}{2}\left(\frac{x-\mu_{inter}}{\sigma_{inter}}\right)^2} dx * 100\% \tag{4.17}$$

In the case of the exemplary test chip $\mu_{intra} = 4.6392\,\%$, $\sigma_{intra} = 13.46$, $\mu_{inter} = 50\,\%$ and $\sigma_{inter} = 32$. The results for FRR and FAR in dependence of HD_{MAX} are depicted in Figure 4.16. The linear scale of Figure 4.16a is not suitable. A logarithmic scale should be preferred (Figure 4.16b). For $HD_{MAX} = 10\%$ FRR becomes approximatly $\approx$ 10E-40. The FAR is smaller than 10E-500.

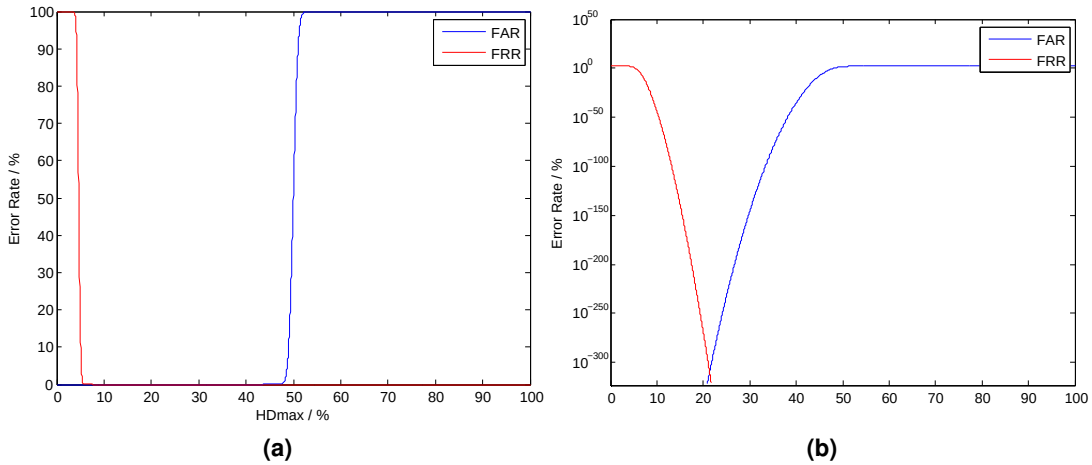


Figure 4.16.: False Acceptance Rate (FAR), False Rejection Rate (FRR) in dependence of HD_{MAX} (a) Linear scale; (b) Locarithmic scale.

4.5. Summary

To summarize this chapter, the measurement results from the exemplary test chip are shown in Table 4.4. The mean value $\bar{x}$, the correlation between bits (R_{xx}), and correlation between chips

Table 4.4.: Summary of the specification results.

Property	Identifier	Ideal PUF
Mean value	$\bar{x}$	0.5046
Error rate	$HD_{intra\,120°C}$	4.6392 %
Corr. between bits	R_{xx}	≈ 0
Corr. between chips	HD_{inter}	50.0271 %
Power consumption	$\frac{E}{bit}$	391.5 $\frac{pJ}{bit}$

(HD_{inter}) turn out to be very good. All of these measures are within statistic feasible ranges. The intra-chip Hamming distance HD_{intra} also stays small as long as the temperature does not change. Hence, the noise performance is satisfying, too. The temperature dependence turns out to be the biggest issue. An error rate of 5 % is too high for reasonable error correction codes. Thus, in future implementations a focus should be on this problem. As already mentioned above, also the power consumption is also too high but they are not of major improtance.

5. Error Correction Codes

by Maximilian Hofer

5.1. Fundamentals

In general, error correction is based on information theory. The theory was developed by Claude E. Shannon and it was published in the year 1948 in *A Mathematical Theory of Communication* [128]. Furthermore, Hamming's work on coding theory was published in *Error detecting and error correcting codes* [55]. In the following section the fundamentals of PUF-related error correction are covered.

5.1.1. PUFs and Errors

The ideal PUF always returns the same output. If the output of the PUF cells is stable, no error correction is necessary [63]. Therefore, no NVM is needed to store the helper data which is desirable due to the extra costs of such memory. As a matter of fact, the output of PUFs is noisy and so helper data of the ECC needs to be stored to guarantee a stable output. There may be some application like identification (see 3.1)where a noisy PUF output can be accepted as long as the error rate stays below a certain limit. In most other applications a constant output is required (see Chapter 3) and thus error correction is unavoidable.

Normally, error correction is used to correct errors that occur during transmission and storage of data over a transmission channel. Additional data are added as redundant information which allows to correct a certain amount of errors [85]. In the case of PUFs the errors do not occur during transmission but during data generation. Nevertheless, the problem is equivalent. Since a binary output is provided by the PUF, the bit error rate (BER) is used as a measure. The BER is defined as the ratio between the number of false bits and the whole number of bits in a bit string. The BER is often given in percent. The BER is defined by

$$BER = \frac{e}{N}, \tag{5.1}$$

where e is the number of errors and N is the total number of bits.

5.1.2. Binary Symmetric Channel

To analyze the behavior of the errors mathematically the binary symmetric channel (BSC) model is being used. A binary symmetric channel is a very simple model in the information theory that is applied to characterize the errors in a channel. In a BSC the probability of a '1' to turn into a '0' and the probability of a '0' to turn into a '1' are identical. Since PUFs show this behavior, it makes sense to utilize the BSC approach. The model is shown in Fig. 5.1 and in Equations 5.2 to 5.5.

$$Pr(Y = 0|X = 0) = 1 - p \tag{5.2}$$
$$Pr(Y = 0|X = 1) = p \tag{5.3}$$
$$Pr(Y = 1|X = 0) = p \tag{5.4}$$
$$Pr(Y = 1|X = 1) = 1 - p \tag{5.5}$$

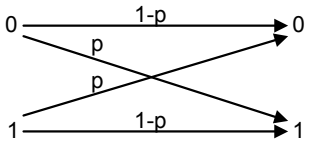


Figure 5.1.: Binary Symmetric Channel.

Y contains the value of a PUF cell which is stored in the reference vector and X is the value of the same cell during usage. p is the the probability that an error occurs (i.e. BER).

The capacity of a channel is defined as [92]:

$$C = 1 - H(p), \tag{5.6}$$

where $H(p)$ is the binary entropy function:

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p) \tag{5.7}$$

If $p = 10\,\%$, $H(0.1) = -0.1 \log_2 0.1 - 0.9 \log_2 0.9 = 0.469$. Thus, the channel capacity in this example gets 53.1 %. In Figure 5.2 the relation is depicted.

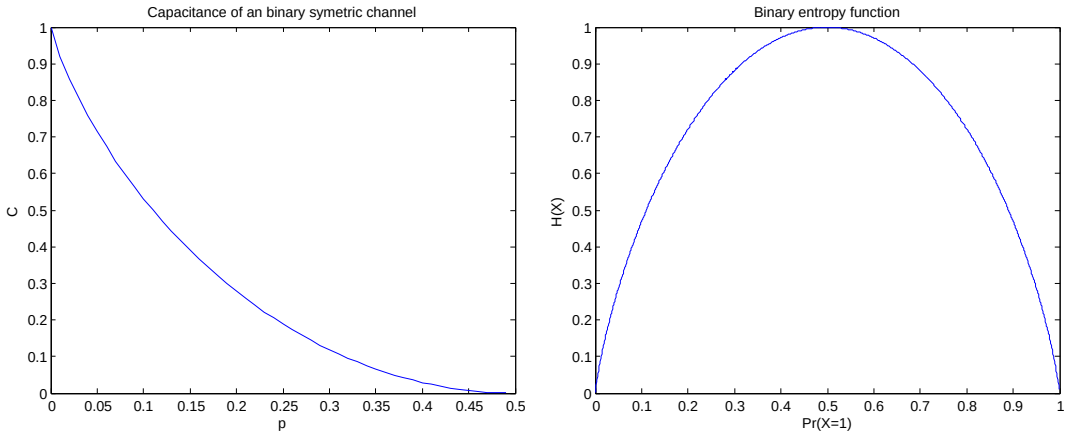


Figure 5.2.: (a) Channel capacity; (b) Binary entropy.

The Shannon's channel capacity theorem [128] defines an upper bound for the required redundancy of error correction codes. This can be seen in the Figure 5.2a. In practice, this value is higher. This occurs due to the limited block sizes, the complexity of error correction codes and because of not ideal ECC algorithms. The upper limit also depends on the desired BER after error correction.

5.1.3. Error Correction Codes

Without any correction, the output of a PUF is usually noisy. Depending on the environment PUF cells show error rates of 10 percent and more [138]. To get a noiseless output an additional procedure is necessary. Pre-processing and post-processing are two possible approaches for such a procedure. In post-processing approaches error correction codes help to reduce the number of errors [36]. The principle behind error correction for PUFs is shown in Figure 5.3. Helper data

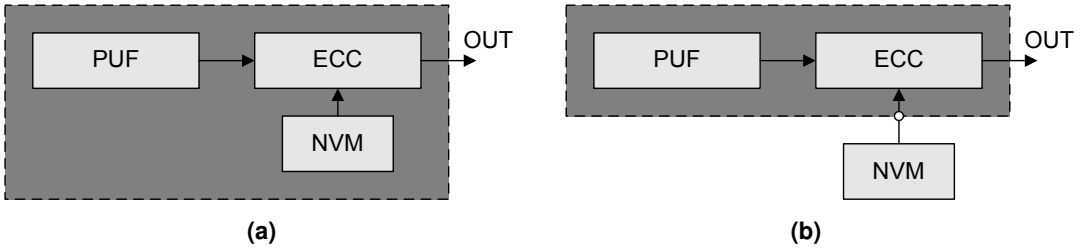


Figure 5.3.: PUF with error correction
(a) On chip NVM; (b) Exernal NVM.

are necessary for the error correction codes. The helper data are generated during an initial phase before the PUF is used for the first time. These helper data have to be stored on an NVM and have to be assumed to be public. Therefore, it is important to make sure that an possible adversary, who has access to these data, is not able to derive information about the PUF's output. If this is guaranteed, the helper data can be stored on the chip (Figure 5.3a) or even on an external storage device (Figure 5.3b).

To keep the secrecy of the PUF, the number of PUF output bits has to be larger than the number of required output bits. In this case, the redundant information is used to mask the helper data. The redundancy is defined as [55]:

$$R = \frac{n}{m}, \tag{5.8}$$

where n is the number of required output bits, and m is the number of transmitted bits.

When choosing an ECC, the following points should be considered:

- redundancy R: low redundancy is followed by a low number of required bits. A lower number of bits is followed by a smaller area consumption.
- complexity of calculation: complex calculations require time, power and appropriate hardware.
- error reduction: for cryptographic purposes no errors are tolerable. So, an appropriate ECC depends on the error probability of the raw output of the PUF cells and the required error probability after error correction.

The selection of an ECC depends strongly on the application dependent requirements in terms of post-ECC error rates, area demand, power consumption, etc.

To generate the redundancy, an ECC encoder is needed at the transmitter-side. At the receiver-side an ECC decoder is used to generate the error-reduced output. There are two possible locations that are suitable for the encoding process. These locations are depicted in Figure 5.4. In 5.4a the encoding is done on the chip. In 5.4b the encoding is done externally.

The advantage of external ECC encoding is that the hardware has not to be implemented on the chip. The big disadvantage is that confidential information has to leave the chip during the initialization. Thus, initialization has to be done within a secure environment. Usually, the decoding process is much more complex than the encoding process [85]. Therefore, it would not be very costly to place the encoder on the chip as well. Even the same hardware may be used for both, encryption and decryption.

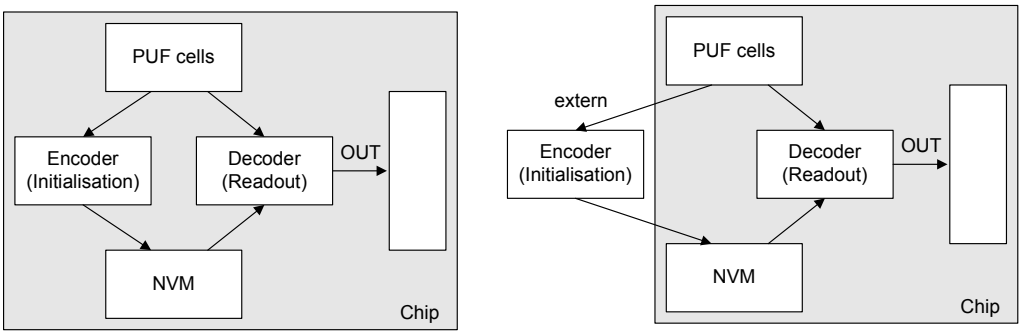


Figure 5.4.: ECC encoder: (a) On chip encoder; (b) Exernal encoder.

5.2. Error Correction Techniques

5.2.1. Approach: Reducing Errors Using More Than one PUF Cell

Example: three cells are used to produce one bit of output. The output is determined by a majority decision: If there are more ones than zeros (011, 101, 110, 111) in the bit string, a '1' appears at the output. Otherwise, if there are more zeros than ones (000, 001, 010, 100), a '0' is generated. There are two bit strings (000, 111) where two bits have to change in order to change the output value. Furthermore, there are six bit strings in which one bit has to change in order to change the majority decision. We are now entitled to ask: does such an approach reduce the error probability?

Example: three zeros (000): an error occurs, if two or all three bits change. If BER is the error probability per bit, the over all error rate can be determined as follows:

$$3\left(BER^2(1-BER)\right) + BER^3 = 3BER^2 - 2BER^3 \tag{5.9}$$

Example: 001, 010, 100: an error occurs, if one or two '0' change or if all bits change. In this case the overall error rate gets

$$\begin{aligned} & 2(BER(1-BER)^2) + (BER^2(1-BER)) + BER^3 \\ = \;& 2BER - 4BER^2 + 2BER^3 + BER^2 - BER^3 + BER^3 \\ = \;& 2BER - 3BER^2 + 2BER^3. \end{aligned} \tag{5.10}$$

Combining the above results the overall error rate BER_0, which shows the probability that an error occurs at a nominal '0' output, can be determined as follows:

$$\begin{aligned} BER_0 &= \frac{1}{4}(3BER^2 - 2BER^3) + \frac{3}{4}(2BER - 3BER^2 + 2BER^3) \\ &= \frac{3}{4}BER^2 - \frac{2}{4}BER^3 + \frac{6}{4}BER - \frac{9}{4}BER^2 + \frac{6}{4}BER^3 \\ &= \frac{6}{4}BER - \frac{6}{4}BER^2 + \frac{4}{4}BER^3 \\ &= \frac{3}{2}BER - \frac{3}{2}BER^2 + BER^3 \end{aligned} \tag{5.11}$$

To evaluate the overall error rate BER, BER_0 and the overall error rate BER_1, which shows the probability that an error occurs at a nominal '1' output, have to be summed up: $BER =$

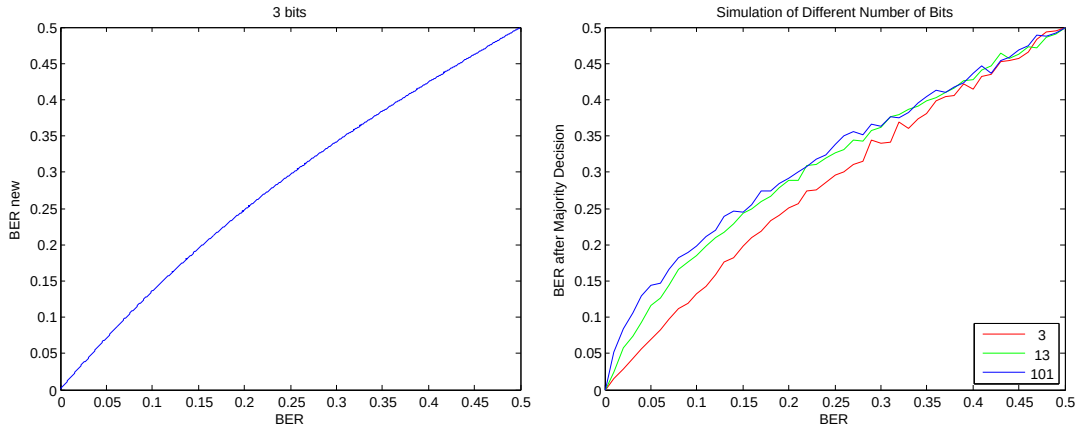


Figure 5.5.: BER after combining: (a) Calculation; (b) Simulation results.

$\frac{1}{2}BER_0 + \frac{1}{2}BER_1$, where $BER_0 = BER_1$. Thus, $BER = BER_0$. In Figure 5.5a BER is depicted. Figure 5.5b shows simulation results for different block sizes. It is obvious that the BER after the majority decision is worse in nearly any case. In the case of BER=0 and BER=0.5 per cell, the BER after the majority decision stays constant. In the case BER of 10 % per cell, the BER the majority decision increases to 13.6 %. In Figure 5.5b it is shown that the situation gets worse concerning combinations of more than three bits.

$$
\begin{aligned}
BER &<= \frac{3}{2}BER - \frac{3}{2}BER^2 + BER^3 \\
-\frac{1}{2}BER &<= -\frac{3}{2}BER^2 + BER^3 \\
-\frac{1}{2} &<= BER^2 - \frac{3}{2}BER \; (BER >= 0) \\
-\frac{8}{16} + \frac{9}{16} &<= BER^2 - \frac{3}{2}BER + \frac{9}{16} \\
\frac{1}{16} &<= (BER - \frac{3}{4})^2 \\
case1 \; BER <= \frac{3}{4} : \frac{1}{4} &<= \frac{3}{4} - BER \\
BER &<= \frac{1}{2} \\
case2 \; BER >= \frac{3}{4} : \frac{1}{4} &<= BER - \frac{3}{4} \\
1 &<= BER
\end{aligned}
\tag{5.12}
$$

The equations show that the assumption is true for 0>=BER<=0.5. Therefore, the following hypothesis is made (without any proof): it is not possible to reduce the BER by combining the output of cells logically without further information.

5.2.2. Error Correction Codes

There exists a number of different error correction schemes which are used to reconstruct an erroneous digital signal and thus to compensate a noisy channel. In the case of PUFs this "channel"

consists mainly of the environment in which the PUF-cell produces its output. The environment is defined by noise (thermal and Flicker noise), temperature, the supply voltage and other effects which may force the PUF-cell into an unexpected direction and thus produces a wrong output. The sources of errors are discussed in detail in the Chapter 6.

To detect and correct false outputs of the single PUF cells, error correction schemes are being used. This becomes especially important as soon as the output of the PUF is used for error-sensitive purposes like key generation in cryptographic applications. In the case of PUF error correction it is crucial to know that the channel does not just produce random (noise) but also deterministic errors (temperature, supply voltage). These deterministic errors turn out to be even worse than noise which can be reduced by averaging techniques.

Error control can be divided into automatic repeat requests (ARQ) and forward error correction (FEC) [85]. Since PUF errors may be both, deterministic and random, a simple repetition of request will lead to the same wrong output and thus ARQ schemes are not feasible. FEC algorithms have to be selected for PUF error correction. Thus, redundancy is added to the data - also called parity or helper data - to be able to correct errors. This is carried out by choosing the most likely signal from a set of allowed signals in dependence of the incoming data. FEC knows two classes of error correction codes: the convolutional codes and the block codes [85].

Block codes use fixed-size blocks and can be decoded in polynomial time to their block length in practice. There are different block codes that are used [85,106]. One example is the Reed-Salomon code (CD, DVD, DVB). This code works especially well with regard to burst error corrections. For that reason Reed-Salomon codes are not a good choice for PUF purposes since the errors are distributed uniformly. Generally, BCH codes for high error rates are the preferred block code for PUF error correction. Those codes are able to correct single bits. The power of such a code depends on the block length and the number of redundant bits. In the upcoming sections the Hamming Code(Section 5.2.3), the repetition Code(Section 5.2.4) and the BCH Code (Section 5.2.5) are explained more detailed.

An mathematical model for an ECC is depicted in the following equation:

$$\begin{aligned} Encoding: &\quad \{0,1\}^k \rightarrow \{0,1\}^l \\ Decoding: &\quad \{0,1\}^l \rightarrow \{0,1\}^k, \end{aligned}$$

where k is the number of bits in the original signal and l is the number of bits of the encoded block. By means of error correction redundancy is added to the data (HD_{MIN} bits and to correct $\left\lfloor \frac{HD_{min}-1}{2} \right\rfloor$ errors.

To reach the Shannon's channel capacity theorem (equation 5.6 and 5.7) the following inequality must hold true:

$$1 + p \log_2 p + (1-p) \log_2 (1-p) \leq \frac{k}{l} \tag{5.13}$$

To receive an adequate BER after ECC much more overhead is needed than the theoretical minimum value.

The second group of FECs are the convolutional codes [85, 106]. Unlike the block codes which usually use hard decisions for their input and output (one or zeros), the convolutional codes are commonly based on soft decisions (analog inputs) which increase their performance. Most of the codes use the Viterbi algorithm for decoding which allows an asymptotical optimal decoding with an exponential increasing complexity. Convolutional codes and their iteratively working derivatives like the turbo-convolutional-codes could be good approaches for PUF error correction since the implementation effort is comparable small. The biggest problem is to find an appropriate code. Unfortunately, this cannot be conducted deterministicly [85, 106].

5.2.3. Hamming Codes

In the following section the Hamming code is used as a block code example. This type of error correction code was presented in the year 1950 by Richard Hamming in [55] wherein the Hamming distance between the single code word has a minimal value of *three* per definition. Thus, only one error can be corrected.

To categorize the Hamming code the following syntax is used: Hamming(N, n), where N is the total number of bits, and n is the number of data bits. In Table 5.1, different Hamming codes are shown. In this case k is the number of parity bits. The encoding using a Hamming(7,4) code is

Table 5.1.: Different Hamming codes Hamming(3,1) - Hamming(255,247).

n	k	N=n+k
1	2	3
4	3	7
11	4	15
26	5	31
57	6	63
120	7	127
247	8	255

presented below: here, 4 data bits and 3 parity bits build the encoded data. The redundancy in the code is $R = \frac{N}{n} = \frac{7}{4} = 1.75$. Table 5.2 [55] shows the encoding table for four data bits (decimal numbers from 0 to 15).

Table 5.2.: Encoding with Hamming(7,4) Code.

1	2	3	4	5	6	7	Decimal	Binary
0	0	0	0	0	0	0	0	0000
1	1	0	1	0	0	1	1	0001
0	1	0	1	0	1	0	2	0010
1	0	0	0	0	1	1	3	0011
1	0	0	1	1	0	0	4	0100
0	1	0	0	1	0	1	5	0101
1	1	0	0	1	1	0	6	0110
0	0	0	1	1	1	1	7	0111
1	1	1	0	0	0	0	8	1000
0	0	1	1	0	0	1	9	1001
1	0	1	1	0	1	0	10	1010
0	1	1	0	0	1	1	11	1011
0	1	1	1	1	0	0	12	1100
1	0	1	0	1	0	1	13	1101
0	0	1	0	1	1	0	14	1110
1	1	1	1	1	1	1	15	1111

The table shows a minimal HD of 3 between the different bit strings. Thus, it is possible to correct one error. Example: the decimal number 10 is encoded to 1011010. If one bit changes (1011**1**10) it can be corrected because only one original bit string of HD=1 exist. If two bits are

changing (10**0**1**1**10), the string will be corrected to 1001100 (Decimal 4) because the HD to the original string (HD=2) is larger than the HD to 4 (HD=1). An overview is shown in Table 5.3.

Table 5.3.: Example of the algorithm, with one error and with two errors.

Original	Encoded	With errors	Corrected	Decoded
1010 (10)	1011010	1011**1**10	1011010	1010 (10)
1010 (10)	1011010	10**0**1**1**10	1001100	0100 (4)

The mathematical background and the algorithm for encryption and decryption can be found in [55].

5.2.4. Repetition Code

The *repetition code* is a very simple error correction code. The idea behind the repetition code R(n) is to repeat the transmitted bit n times [85,106] and then make a majority decision. Although it is very simple the repetition code is powerful. It can be used for signals of very high error rates. The drawback of such codes is the high redundancy that is required. Thus many PUF cells are needed to produce one bit of output. In Table 5.4 an example of an R(9) is shown. Here, the redundancy is $R = \frac{N}{n} = \frac{9}{1} = 9$. The R(9) is capable of correcting four bits of errors. The performance of different repetition code lengths is shown in Figure 5.8.

Table 5.4.: Example of the repetition code R(9).

Original	Encoded	Errors (red)	Decoded
1	111111111	001101011	1
1	111111111	001001011	0

In Chapter 15.2 the repetition code is used to correct errors in the *Microcontroller PUF*.

5.2.5. BCH Code

The BCH Code is an error correction code that was invented by Alexis **H**ocquenghem [Hocquenghem1959] and by Raj Chandra **B**ose and Dwijendra Kumar Ray-**C**haudhuri [Bose1960]. The name BCH derives from the initials of the three inventors. In BCH(n, l, d_{min}) n is the code word length, l the length of the original data and d_{min} the minimal Hamming distance between the code words. $(d_{min} - 1)/2$ errors can be corrected.

As an example, a BCH(15,7,5) code is analyzed. A BCH(15,7,5) has 7 data bits and 5 parity bits. Hence, 2 errors in a block of 15 bits can be corrected. The redundancy R is $R = \frac{n}{l} = \frac{15}{7} = 2.14$.

5.2.6. Comparison

In Table 5.5 an overview of the different error correction codes is given. R(3) is the simplest code. The R(3) shows the best performance but also the highest redundancy. The BCH(15,7,5) reduces the error rate only up to BERs of 6.21 %. The Hamming(7,4) code reduces the error rate up to BERs of 5,79 %. In Figure 5.6, the BER before and after the error correction is shown in a logaritmic and a linear scale (two different ranges).

Table 5.5.: Overview of the different ECCs.

ECC	R	Complexity	Performance	Range of improvement
R(3)	3	Simple	Best	<50%
BCH(15,7,5)	2.14	Complex	Good	<6.21%
Hamming(7,4)	1.75	Middle	Middle	<5.79%

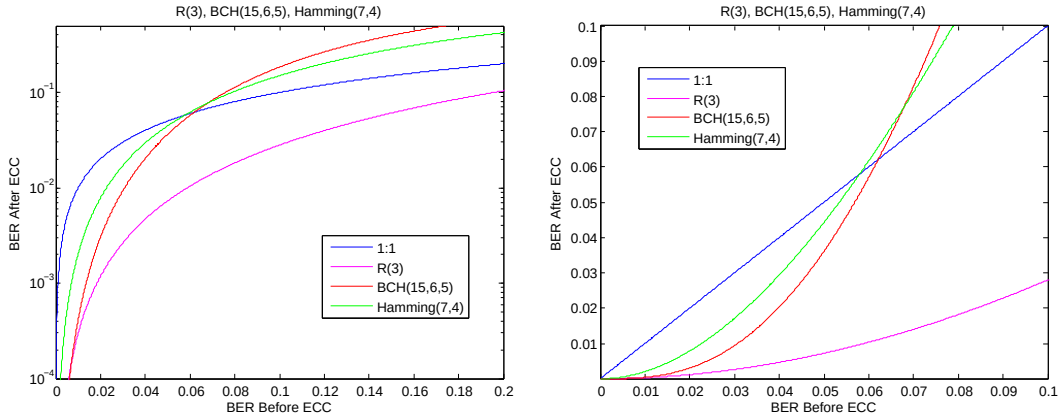


Figure 5.6.: Comparison of R, BCH and Hamming code.
(a) Logarithmic scale; (b) Linear scale.

The blue line shows the error rate without any error correction. At some point the BCH and the Hamming code lines cross the blue line. In the case of higher BER the error rate increases, if error correction is used.

5.3. Error Correcting Codes and PUFs

For most purposes erroneous PUF outputs are not suitable. Due to this fact error reduction is required. In Figure 5.7 a whole error correction scheme for PUFs is shown. The procedure of error correction is divided into two parts: The initialization phase and the read-out phase.

During the initialization phase the helper data are created. This is done by the encoder of the ECC. The helper data is stored in a non-volatile memory (NVM). Since the data in the NVM has to be regarded as public data, it cannot be stored directly because an attacker could possibly use that data for recovering the PUF-output. Thus the helper data is x-ored with additional PUF output bits before storing it in the NVM. To provide those extra bits the number of total PUF cells has to be increased:

$$\begin{aligned} \#\,PUF\_cells &= \#\,required\_output\_bits + \\ &+ \#\,helper\_data\_bits \end{aligned}$$

Since errors are likely to occur at any read-out, the PUF value at READOUT1 which is used during initialization is also likely to be erroneous. To keep this error as small as possible it makes sense to reduce it for example by using the average of a number of read-outs. Nevertheless, noise induced decisions during the initialization phase cannot be avoided.

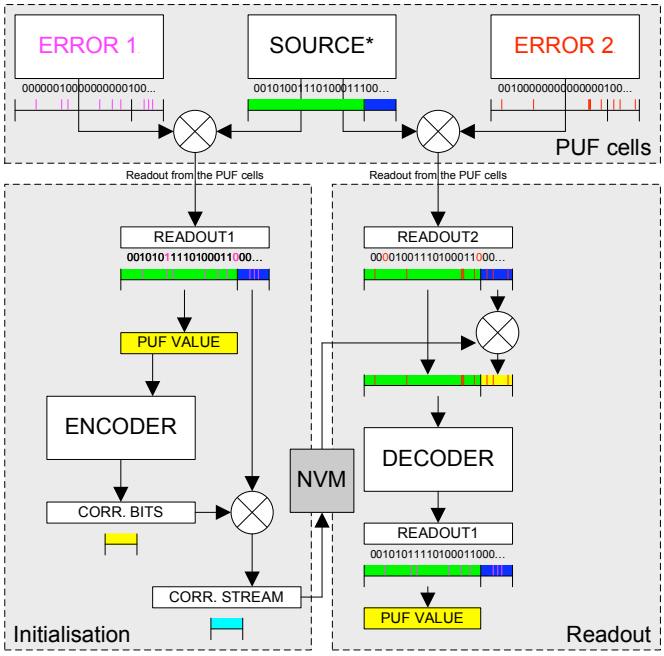


Figure 5.7.: The ECC scheme. The ⊗ denotes an X-OR operation.

The output during the read-out phase should match READOUT1. To reach this goal, during the first step READOUT2 is determined by reading-out the PUF-cells as depicted in Figure 5.7. The additional PUF-cells are xored with the correction stream which is stored in the NVM and noisy correcting bits are received. If the error rate is not too high, the original PUF value READOUT1 can be recovered using the correcting bits with the decoder algorithm. Even if it seems to be simple in practice, it is difficult to implement error correction for the high PUF error rates. For example, the computational complexity of the Berlekamp-Massey algorithm, which is used to decode BCH encoded data, increases quadratically with the block-size [103]. In [32] the authors analyze some concepts for error correction in image and video watermarking with very high bit error rates (BER). In [17] the authors introduce concepts of using the combination of more than one code.

All these complex EECs become difficult to handle as soon as the amount of energy is highly limited as it is for example on RFID tags. In such cases other solutions are required. For this reason the repetition code, a combination of repetition and Hamming code, and a combination of repetition and BCH code is analyzed. For their implementation simplicity and correction capabilities, such codes are an alternative to highly complex codes.

5.4. Analysis of Selected Error Correction Schemes

In Figure 5.8 the error correction capabilities of the repetition code are shown. It can easily be seen that the effort for significant error reduction is high. For example, if a BER of 8 % (p=0.08) has to be reduced to 1E-06 for the whole PUF output of 1024 bit, a BER of about 1E-09 for each bit is needed. Figure 5.8a shows that a repetition code Rep(31) can deliver the desired specification. In the example, 1024*31 = 31744 physical PUF cells and an NVM of 31744-1024=30720 bits are needed to generate the output PUF value. In other words, the area of the cells would increase by

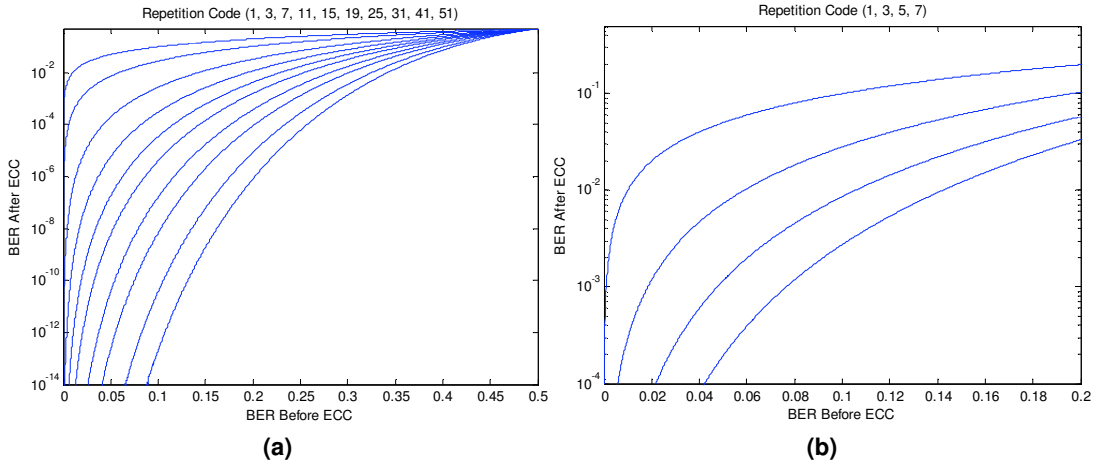


Figure 5.8.: BER of the repetition code;(a) Rep(1) - Rep(51); (b) Rep(1) - Rep(7).

the factor of 31 plus the NVM area.

In Figure 5.8b the benefit of the smallest repetition codes is presented. In this range of BER common ECCs can be used without any concern. This leads to the idea of combining the repetition code and the Hamming code as depicted in 5.9. Concatenated codes were introduced by David Forney in [43]. In table 5.9a the BER after the correction with the repetition code can be seen. Here, an initial BER of p=0.08 is assumed. This first reduction of the BER is very helpful for the

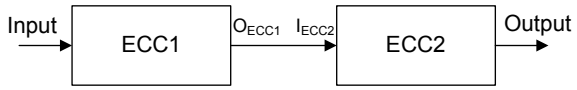


Figure 5.9.: ECC chain.

subsequent error correction.

To provide an example the Rep(7) and the Hamming Code (7,4) is combined. Thus, for 1024 bits of secure output bits 1024*7*7/4 = 12544 physical bits are needed. By means of this combination an error rate of 3E-05 can be reached. This error rate can be determined as follows: the Hamming code is able to correct 1 or 0 errors with the probabilities for the Hamming code(7,4) which are shown in Table 5.6. More errors cannot be corrected.

Table 5.6.: Probabilities for the Hamming code(7,4).

# Errors	Probability
0	$(1-p)^7$
1	$(p\,(1-p)^6)\cdot 7$

From the above results the probability of one or more errors after the Hamming code(7,4) can be determined:

$$p_{afterEC} = 1 - ((1-p)^7 + 7p(1-p)^6) \tag{5.14}$$

Another approach is to use a BCH code which is able to correct more than one error. For example, the BCH(15,6,5) is able to correct 2 bits of error in a block of 15 bits. The probabilities

Table 5.7.: Probabilities for the BCH(15,6,5).

# Errors	Probability
0	$(1-p)^{15}$
1	$(p\,(1-p)^{14}) \cdot 15$
2	$\binom{n}{r} p^2(1-p)^{13} \;=\; 105 \cdot p^2(1-p)^{13}$

of zero, one, and two errors are shown in Table 5.7. The probability of one or more errors after the BCH(16,6,5) can be determined as:

$$p_{afterEC} = 1 - ((1-p)^{15} + 15 \cdot p\,(1-p)^{14} + 105 \cdot p^2(1-p)^{13}) \tag{5.15}$$

In the Tables 5.9a-5.9c the results for different combinations of error correcting codes are given. The advantages and disadvantages of the different approaches can be compared. For example, a repetition code Rep(13) followed by a Hamming code(7,4) having an overhead of about 23 cells exhibits an error probability (one or more errors within 1024 bits) of 1.67E-06. Using a BCH(15,6,5) after a Rep(9) requires the same number of overhead bits but provides an error rate of as low as 1.40E-08. These results show that a appropriate choice of error correction allows to increase the performance of a circuit considerably.

Figure 5.10 shows an approach to a figure of merit for such error correction code. Despite the number of required bits and the error rate after ECC the implementation complexity is being taken into account. The complexity may include energy consumption, effective area on the chip, difficulty of digital implementation or a combination of them. Since the determination of the complexity is difficult without implementing the whole system, Figure 5.10 shows just a symbolic estimation of it. An interesting result of Figure 5.10 is the relation between the number of required cells and the error rate. It nearly shows a linear correlation between the number of cells and the log-scaled error rate.

5.5. Summary

In this chapter an overview of different error correcting codes was presented. Different approaches were analyzed towards their ability of PUF error correction. It turned out that in the case of a low power PUF environment, combinations of repetition codes and simple BCH codes provide promising results and should be taken into account when implementing error correction for PUF systems.

Table 5.8.: BER of different combinations of ECCs. The error probability of a single bit before correction is assumed to be 8 %. The error probability of a single bit after ECC is denoted by p. The propapility of one or more errors within a block of 1024 bits is denoted by e_{1024}. $\#PUF\_cells$ denotes the number of PUF-cells needed to provide one output bit: (a) Repetition Code; (b) Repetition code + Hamming code; (c) Repetition code + BCH code.

Rep(x)	p(Rep(x))	e_{1024}	$\#PUF\,cells$
Rep(3)	1.82E-02		3
Rep(5)	4.53E-03		5
Rep(7)	1.18E-03		7
Rep(9)	3.14E-04		9
Rep(11)	8.50E-05		11
Rep(13)	2.33E-05		13
Rep(15)	6.45E-06		15

(a)

x	p(Rep(x), Ham(7,4))	e_{1024}	$\#PUF\,cells$
3	6.53E-03	6.16E-01	5.25
5	4.24E-04	6.01E-02	8.75
7	2.89E-05	4.23E-03	12.3
9	2.06E-06	3.02E-04	15.8
11	1.52E-07	2.22E-05	19.3
13	1.14E-08	1.67E-06	22.8
15	8.74E-10	1.28E-07	26.3

(b)

x	p(Rep(x), BCH(15,6,5))	e_{1024}	$\#PUF\,cells$
3	2.32E-03	1.47E-01	7.5
5	4.05E-05	2.76E-03	12.5
7	7.33E-07	5.00E-05	17.5
9	1.40E-08	9.55E-07	22.5
11	2.79E-10	1.91E-08	27.5
13	5.77E-12	3.94E-10	32.5
15	1.23E-13	8.41E-12	37.5

(c)

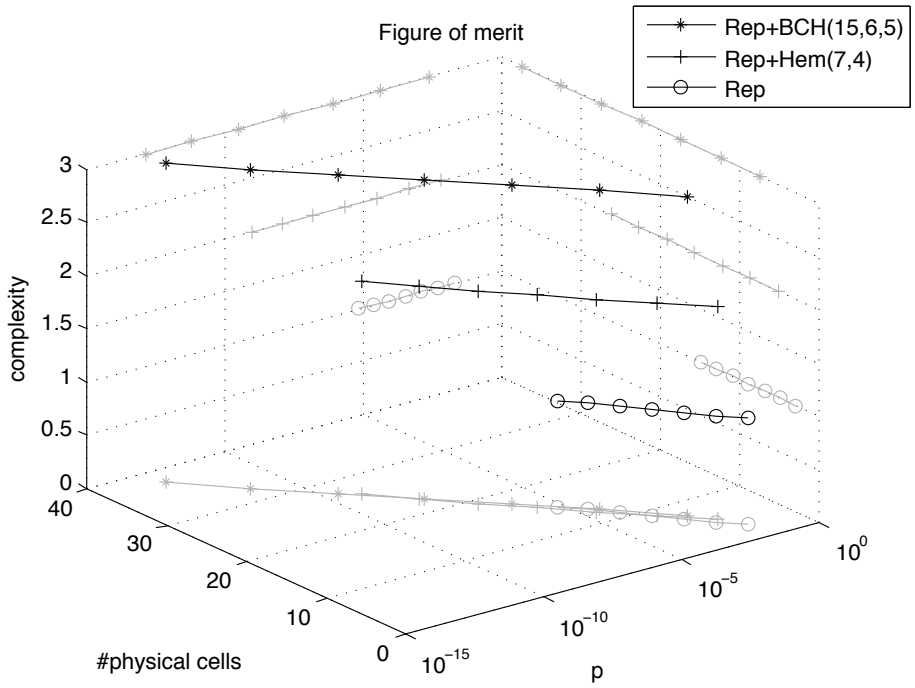


Figure 5.10.: The figure of merit shows the relation between complexity of the ECCs, the number of required PUF-cells and the error rate.

Part II.

PUFs in Silicon

6. Sources of Mismatch and Errors

by Christoph Boehm

The basic requirement for PUFs is that a mismatch between at least some of the involved parts exist and thus a PUF specific output can be generated. In silicon PUFs the mismatch can be provided by different mismatch sources. In general, all available microelectronic components can be used for this purpose: transistors, resistors, capacitances, and inductances. The following chapter introduces the different components and their suitability to be used as mismatch source.

6.1. MOS Transistor

Since MOS transistors are comparable small and since they show high local mismatches compared to the other available electrical components, in most silicon PUF circuits transistors are chosen as a source of mismatch.

There are two types of transistor mismatches: *global/systematic/extrinsic* and *local/stochastic/intrinsic* mismatches [12, 102]. The global mismatches come from *operating dynamics of a modern fabricator* [12]. Local mismatches come from stochastic atomic-level differences which cannot be controlled during production. Figure 6.1 shows the influence of local and global parameter variability on a transistor property (e.g. V_{TH}).

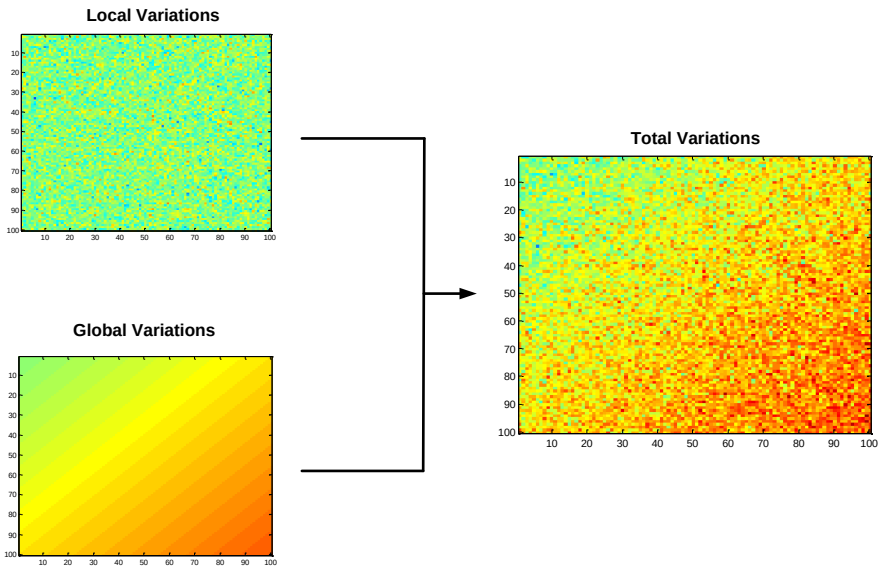


Figure 6.1.: The figure above shows the influence of local and global mismatches. The global mismatch introduces a gradient on the parameter which leads to an overall (see right hand side) non-Gaussian variability distribution. This can lead to a biased PUF output.

6.1.1. Types of Mismatches

Global Mismatches

As mentioned above, global mismatches come from inaccuracies of the production process. As stated in [12] the present inaccuracies result in different mismatch phenomena. These include lot-to-lot, wafer to wafer, and chip to chip variability depending on the source of variation. In [12] it is stated that about 20 % of the variabilities come from production inaccuracy. The global/extrinsic process variability is caused for example by temperature gradients accross the wafer during annealing, by photoresist development, etching process, or photolitographic process variations. These variations lead to global effects on the produced device characteristics. For example, gradients of threshold voltage values can occure over the whole wafer towards a certain direction or device properties may change depending on their placing/direction. Global mismatches can be reduced by chip design and layout: common centroid layout can minimize the influence of gradients on the transistor, or identical current flow directions can compensate asymmetric behavior of circuits.

Local Mismatches

The second kind of mismatch is the local mismatch. This kind of mismatch does not result from inaccuracies during the production process but are based on process inherent variability. PUFs use this kind of variation to build-up the *fingerprint*.

Local mismatches come from stochastic atomic level differences. Up to now there is no way to control the process on a atomic level. Moreover, the smaller the minimal feature size, the higher the influence of each atom. In nanometer-scale transistors the gate oxide thickness is defined by only a few atoms. Due to that the thickness can vary up to 50% within a single transistor [9]. This leads to variation of mobility and gate tunneling (gate current). Which mismatch sources really influence the behavior of the devices depend of course on their physical structure. For example, there exist various types of sub-100nm-transistors which have different mismatch sources: conventional bulk transistors show dependencies on the doping concentration in the poly-silicon gate region which do not appear in metal gate transistors with high-k dielectric material. Fully depleted SOI (silicon on isolator) transistors show a dependence of the threshold voltage on the body thickness [12]. A planar bulk MOSFET is influenced by three major local mismatch sources: Random discrete doping (RDD) in the channel region, line edge roughness (LER) which discribes the gate length variation along the width, and polygate granularity (PGG) which is the effect of poly-silicon grain boundery distribution on the threshold voltage [146]. It can be seen that for this kind of transistor RDD dominates the mismatch behavior.

Concerning halo implanted devices the RDD consists of two sources: the substrate doping which is defined by the concentration and distribution of the implant atoms over the whole substrate region, and the doping in the two halo regions. The halo regions are placed on top of the substrate region to reduce the width of the depletion region and thus minimize the short channel effects like DIBL in short channel transistors [145]. This leads to further mismatch effects which are described later more detailed.

Temporal Mismatches

Temporal mismatches are mismatches that are not resulting from production variabilities but show up during the usage of the devices [12]. Temporal mismatches can be classified into reversible and irreversible mismatches. Reversible mismatches include mainly local temperature differences that lead to a mismatching behavior of the involved devices. Also local variations of VDD or bias currents can cause temporal mismatches. The second category are the irreversible temporal

mismatches which include device degenerating effects like NBTI and hot electron effect. Some of the temporal mismatches are presented in the following subsections.

Local Temperature Shifts: local temperature shift occur when the current is distributed over the chip unevenly. So, the different parts will show different temperatures. In PUF design local temperature shifts between the mismatch transistors have to be avoided since the change of behavior due to the temperature difference could lead to unexpected output. This can be done by placing the mismatch devices as close as possible.

V_{DD} Inaccuracies: due to voltage drops across the power supply local V_{DD} variations can occur over the chips. Figure 6.2 shows the simulated V_{DD} at the different regions of a chip. The changes have to be taken into account during PUF design.

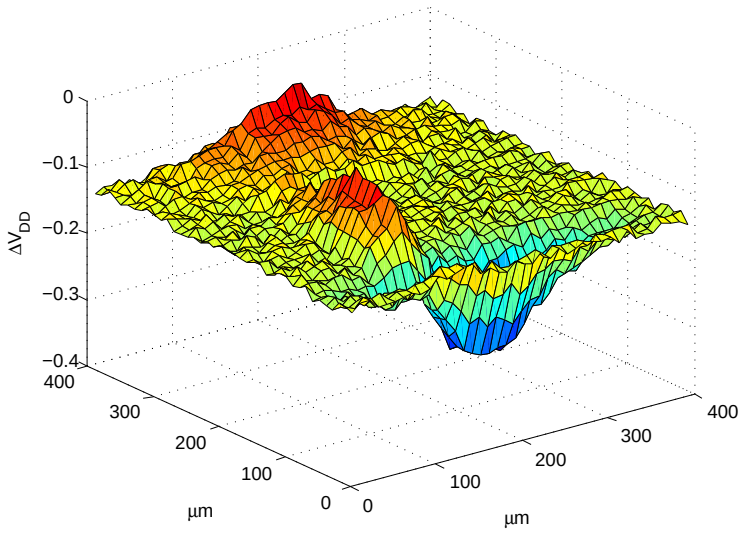


Figure 6.2.: The figure shows the V_{DD} values at different regions of a chip. These values come from a simulation.

NBTI and PBTI: negative bias temperature instability (NBTI) describes an physical degeneration effect which results in an increase of the threshold voltage and the subthreshold slope and a decrease of the g_m over time of negative biased transistors. The negative bias refers to the gate-source voltage. In practice, NBTI is mainly an issue of PMOS transistors since NMOSTs are commonly not biased in accumulation regime. NBTI arises for two reasons: *Interface traps* and *charge traps*. Interface traps occur when holes in the channel weaken the Si-H bond between the silicon in the channel region and the hydrogen atoms in the substrate-SiO2 interface. In the interface region, hydrogen atoms are placed to satisfy all the Si atoms in the substrate since this is not provided by the SiO2 alone (Figure 6.3).

Due to the influence of temperature, hydrogen atoms are displaced. Some of the H atoms will again bond on Si atoms, others will diffuse as H2 molecules and leave a gap in the lattice. On account of the gaps in the lattice, some of the holes in the transistors will close these gaps before the current starts to flow. So, the threshold value is increased. If too many hydrogen atoms leave a gap it even may cause a short circuit between substrate and gate and finally destroy the device. NBTI increases with growing temperature and with increasing negative gate-source voltage.

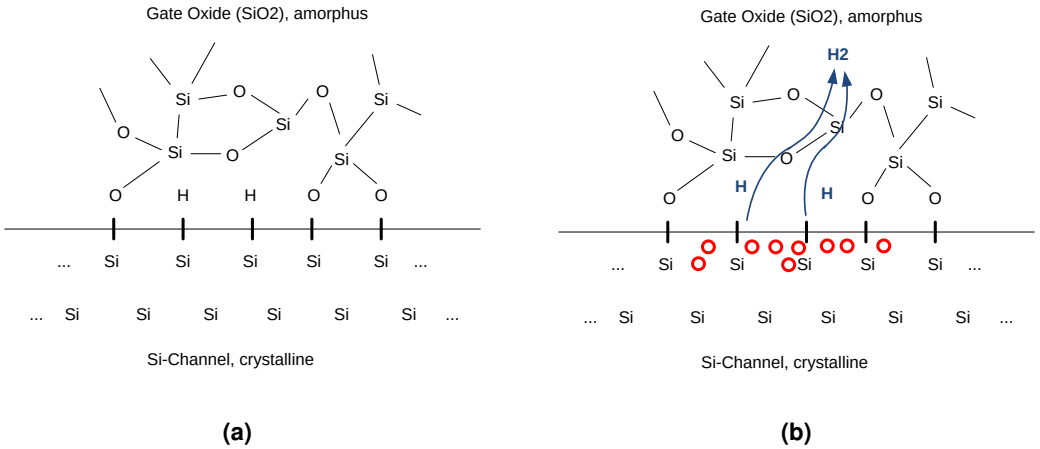


Figure 6.3.: Example of channel-oxide interface with H atoms at the boundary 6.3a. If negative bias voltage is applied, charge carriers (holes (o)) weaken bonds and H2 molecules diffuse 6.3b. (Inspired by [4])

Since some of the free H atoms will bond again as soon as the bias is removed, NBTI is partially reversible. The effect of NBTI can be seen in the following formula:

$$N_{IT}(t) \propto K \cdot t^{1/6}, \tag{6.1}$$

where the interface trapping N_{IT} is the number of produced gaps, K the physical properties of the material and t the time.

In addition to Si-H bond breaking, which results in the interface traps N_{IT}, *charge trapping* N_{HT} is another reason for NBTI. Due to defects in the oxide, charges are trapped again depending on the gate-source voltage and the temperature. For some types of PMOS transistors, charge trapping can be the dominant effect for NBTI [71]. Unlike the N_{IT} caused NBTI the N_{HT} induced V_{TH} shift can be completely recovered once the negative biased is removed.

For high-k dielectrics which are for example based on Hafnium dioxide (HfO_2) and SiO_2, the positive bias temperature instability (PBTI) has also to be taken into account [71, 120]. Here, due to oxide defects - as in the case of the hole traps in NBTI - some of the electrons in the channel are trapped inside the dielectric and thus the threshold voltage is increased. This effect affects mostly NMOS transistors since those are mostly biased with a positive gate-source voltage. Like in the N_{HT} NBTI case the number of trapped electrons N_{ET} again reduces as soon as the bias voltage is removed. Since in contrast to NBTI, interface trapping has no effect on PBTI, PBTI is recoverable. It is important to mention that for transistors with high-k dielectrics both NBTI and PBTI exist.

Figure 6.4 shows a schematic view of a metal gate high-k transistor and the effects that lead to NBTI and PBTI. In Figure 6.5 the effect of NBTI and PBTI on the threshold voltage of a transistor is shown.

Hot Electron Injection: another important device degeneration effect is called *hot carrier degeneration* or *hot carrier injection* (HCI) [68]. HCI is based on the same degeneration phenomenon as BTI: Si-H bonds are broken and leave a charge trap which changes the behavior of the affected transistor. Unlike BTI in which the bonds are weakened by the charges in the channel even if no current is flowing, HCI occurs as soon as charge carriers reach a kinetic energy level where they can overcome the potential barriers between the different areas of the transistors, i.e.

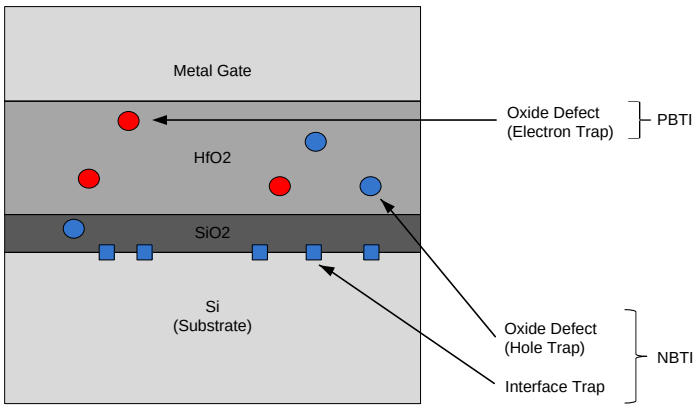


Figure 6.4.: Part of transistor with HfO_2 high-k dielectric and metal gate. The interface traps and the negative oxide defects lead to hole traps and thus NBTI. The positive oxide defects lead to electron traps and thus to the reversible PBTI.

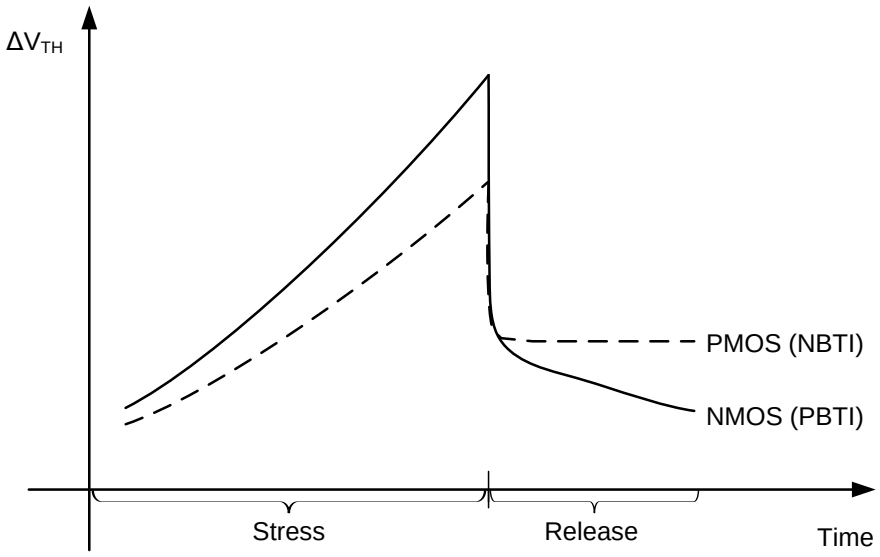


Figure 6.5.: The above figure shows the change in threshold voltage of an NMOST and a PMOST if the transistors are stressed for a certain time. After the stress the original V_{TH} is partly recovered [120].

the gate dielectric or the substrate under the channel. Then, a leakage current flows through the substrate or through the gate and traps can occur. Si-H bonds are broken and change the behavior of the transistor. Two requirements must be achieved to allow HCI: the electric field must be high and the mean free path must be long. The higher the voltage and the shorter the channel is, the higher is the electric field in the transistor. The lower the temperature is, the longer is the mean free path for the electrons to be accelerated.

In addition it has to be taken into account that the different approaches to increase the performance of MOS transistors also change the behavior of HCI. For example, in the case of drain-extended NMOS transistors, HCI appears during the off-state ($V_{GS}= 0$) with $V_{DS}> 0$ considerably [153].

HCI is not reversible and degrades the transistor (NMOS and PMOS) in perpetuity. In Figure 6.6 a schematic overview of hot carrier behavior is shown to illustrate the effects.

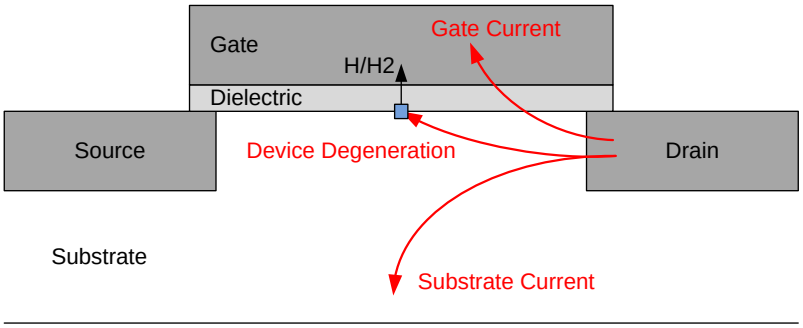


Figure 6.6.: The hot charge carriers have different effects on the transistor which result in leakage current and device degeneration.

Gate Dielectric Breakdown: time dependent dielectric breakdown (TDDB) is a degenerative effect which finally leads to the destruction of the device. Due to high electric fields and enforced by high temperatures traps in the gate dielectric are generated. This first phase is also called pre-breakdown stage [140] or build-up stage. At some point in time (t_{DB}) it may happen that several traps build a direct connection between gate and substrate and a current starts to flow. This current produces new electron-hole pairs due to impact ionization. This ionization is like a positive feedback which leads to a higher current which eventually destroys the transistor. This second phase is also called fast runaway process. The position were the breakdown takes place is often a weak region of the oxide which results from production variabilities. A schematic view of such a breakdown is given in Figure 6.7

Electromigration Another important degradation effect is called electromigration. This effect affects the copper or aluminum lines between the transistors and other components. Due to the shrinking dimensions of power lines electromigration becomes more important in modern technologies. Electromigration is the process of material transport due to the current through the interconnects. Through the loss of material the impedance of the interconnect changes and finally electromigration may lead to an open circuit. The mean time to failure created by electromigration can be estimated by Black's equation [16]:

$$\frac{1}{MTF} = AJ^2 exp\left(-\frac{\phi}{kT}\right), \tag{6.2}$$

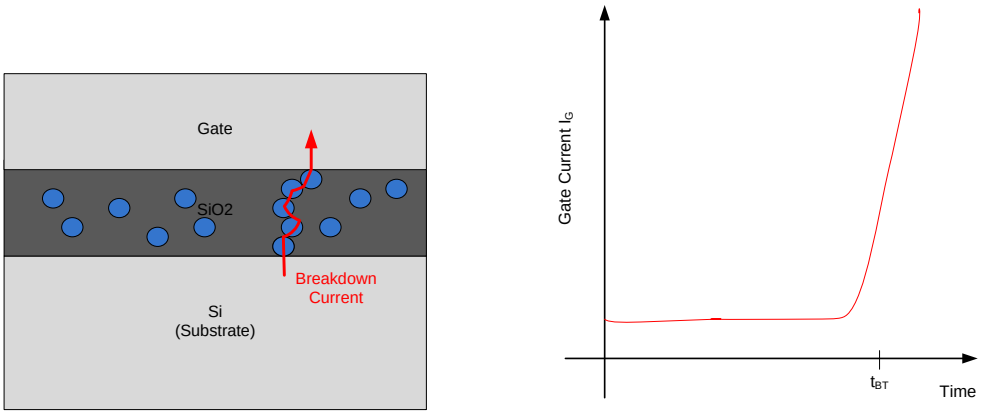


Figure 6.7.: In 6.7a an example of TDDB is shown. At some point in time (t_{DB}) a channel is built through the oxide which leads to a short current. 6.7b shows the development of gate current over time [153]. At t_{DB} the current increases rapidly.

where MTF is the median time to failure in hours, A a cross-sectional area dependent constant, J the current density, ϕ an activation energy, k the Boltzman's constant, and T the temperature in degrees Kelvin. A sketch of the migration process can be seen in Figure 6.8.

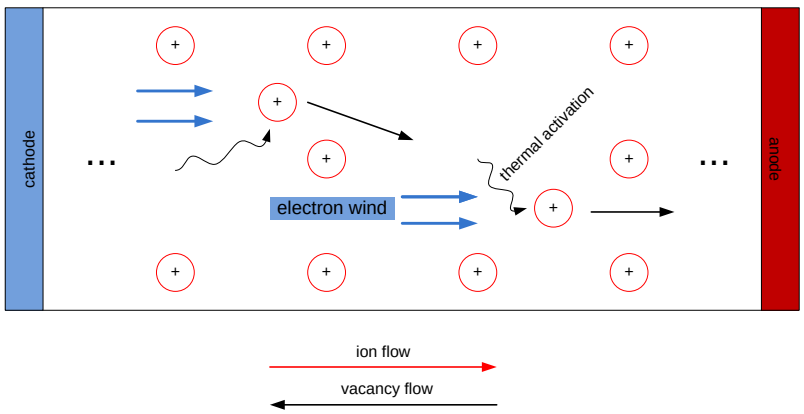


Figure 6.8.: Electromigration happens due to the interaction between electrons and metal atoms. According to the electron wind, the ions flow towards the anode.

Due to the electron wind the ions are forced towards the anode and form discontinuities like crystals or hillocks. The generated vacancies move towards the cathode and build up defects.

6.1.2. PUF-Specific Layout

Minimizing Global Mismatches

To maximize the randomness in the PUF output, the mismatches between the involved transistors have to be free of any gradient. Otherwise, a bias at the output can occur which makes it easier to predict. In such a case, inter-chip Hamming distance is not 50%. There are different approaches to minimize that gradient for MOS transistors. A list of rules is given in [57]. In order to minimize global misatches the following rules apply.

- The matching transistors have to be of the same size (length and width). Also the single fingers of one transistor have to match in size.
- If voltage should match, keep V_{GS} small. This can be realized by increase of $\frac{W}{L}$ ratio.
- If current should match, V_{GS} has to be high. Thus the influence of V_{TH} is decreased.
- The orientation of the matching transistors have to match to minimize mobility variations.
- Transistors have to be placed as close as possible to minimize effects of gradients.
- Use common centroid layout to minimize effect of gradients.
- Use dummy gates at the sides.
- No contacts on top of the active gate.
- No metal lines across active gates.

In Figure 6.9 some of the layout rules are depicted.

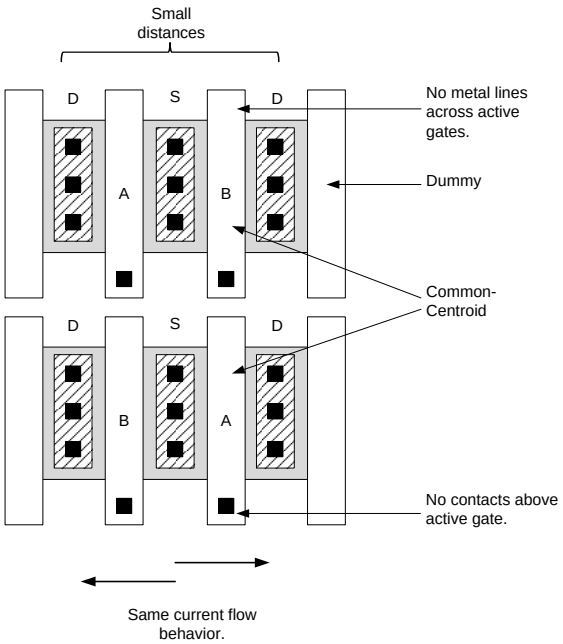


Figure 6.9.: Some of the basic layout rules are shown to minimize the effect of the global mismatches on a pair of transistors.

Maximizing Local Mismatches

In contrast to common analog circuits it is important in PUF design to maximize the local mismatches between the transistors. These mismatches are assumed to follow a Gaussian distribution. The larger the mismatch's standard deviation gets the smaller the error rate will be. This occurs because of a larger standard deviation which produces less PUF cells with near-zero mismatches. The only effective way to increase the local mismatch is to minimize the size of the transistors. Pelgrom [114] published this correlation in 1989: due to the smaller area the random fluctuations are less averaged-out compared to larger devices. This is still true for modern devices.

In general, PMOS and NMOS transistors show a different amount of local mismatches. The influence depends on the operating point of the transistor. In Figure 6.10 the influence on the drain current is compared between PMOS and NMOS transistors at different gate-source voltages. Figure 6.10a shows the standard deviation of the relative local mismatch as simulated for a standard CMOS process. It can be seen that PMOS transistors exceed their NMOS counterparts over the whole V_{GS} range. Especially in the sub-threshold region the relative mismatch grows up to 85 %. So this region of operation should be ideal for PUF circuits. Unfortunately, I_D in the subthreshold region becomes so small that the noise takes over and defines the output current. This can be seen in Figure 6.10c. If V_{GS} is smaller than 0.8 V, the noise is the dominant part (red lines) and it becomes worse if the high frequency components (up to 100 GHz) are included in the analysis (black lines). Figure 6.10d shows the same result from another perspective. Here, the standard deviation of $I_D$is compared to the RMS of the noise (red and black lines). As soon as these curves cross no meaningfull ouput can be expected from a PUF. Both figures show that the NMOS transistor performs better. This is because of the higher current at identical V_{GS} (see Figure 6.10b). Thus, it is not so clear which type of transistor fits better for PUF purposes. In the end this depends strongly on the technology that is used and has to be determined from case to case. Additionally, the aging properties have to be accounted for. Here, NMOS transistors often show a more stable behavior. Even if these results were received from simulation data, they give a strong hint to where to focus on during PUF design.

6.1.3. Parameter Variability and PUF Design

The different sources of mismatch influence the behavior of the involved devices. Transistor variability can be observed by comparing their electrical parameters which are drain current, input voltage (V_{GS}), trans-conductance (g_m), and output conductance (g_o). Drennan and McAndrew proposed a MOSFET mismatch model [37] where they included all important process parameters which lead to a valid transistor model. They extended the approach of Pelgrom et al. [114] with it. The model from [37] shows the influence of the physical parameters on the electrical parameters. The list of physical parameters includes flatband voltage (V_{fb}), mobility (μ), substrate doping concentration (N_{sub}), length offset(ΔL), width offset (ΔW), short channel effect (V_{tl}), narrow width effect (V_{tw}), gate oxide thickness (t_{ox}), and source/drain sheet resistance (ρ_{sh}). If halo doping is used, the influence of the doping profile has to be taken into account and the model has to be extended like it is done for the Pelgrom model in [125]. The variance of the physical parameters also influences the variances of the electrical parameter (e.g. I_D). This can be expressed by the following formula [38]:

$$\sigma_e^2 = \sum_i \left(\frac{\partial e}{\partial p_i} \right)^2 \sigma_{p_i}^2, \tag{6.3}$$

where σ_e^2 is the variance of a electrical parameter, $\sigma_{p_i}^2$ the variance of the physical parameter, e the electrical parameter, and p_i any physical parameters.

In addition to the influence of variation on $I_D$at constant temperature, parameter variability also

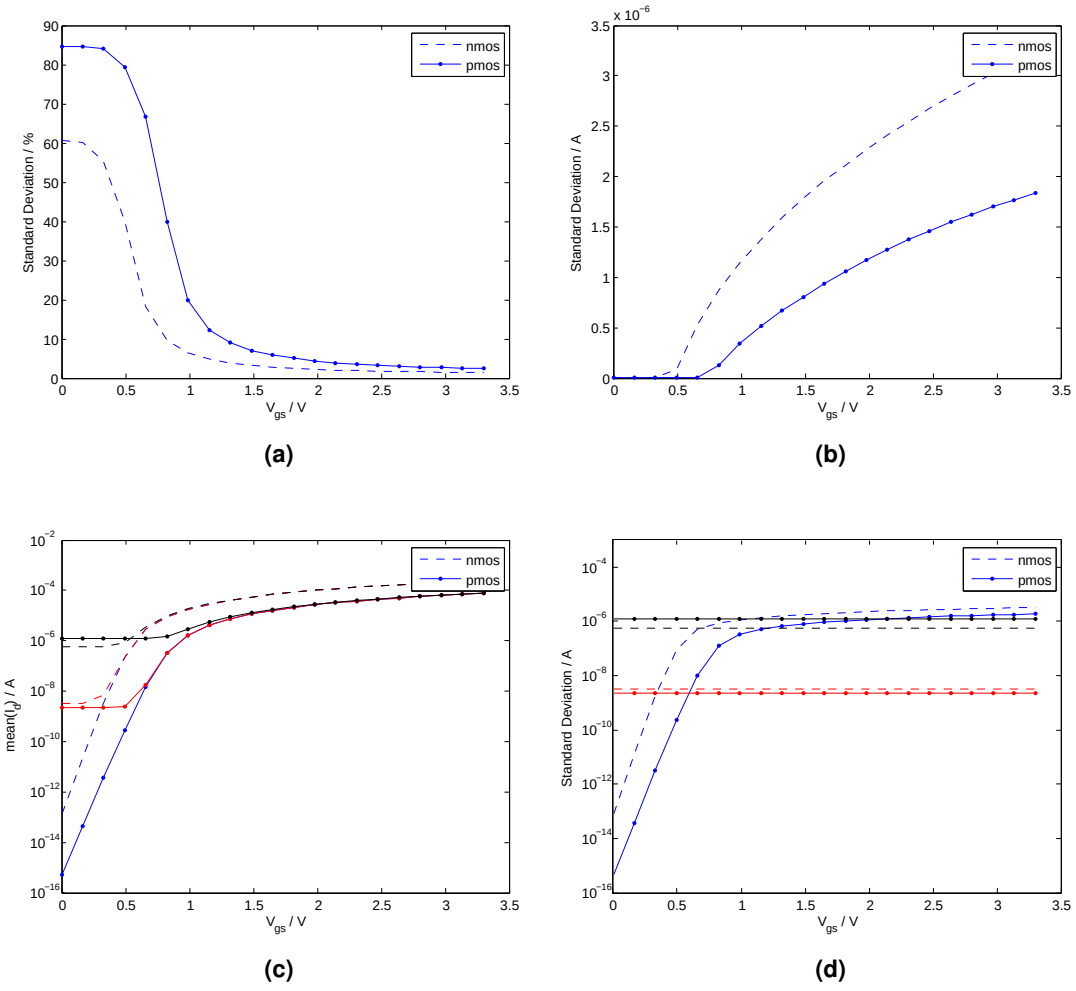


Figure 6.10.: Influence of mismatch at different V_{GS}: (a) Standard deviation of I_D in %. (b) Standard deviation of I_D in A. (c) Mean I_D noiseless (blue), noisy (1 Hz-1 GHz)(red), noisy (1 Hz-100 GHz)(black). (d) Std I_D (blue), RMS noise (1 Hz-1 GHz)(red), RMS noise (1 Hz-100 GHz) (black).

changes the temperature behavior of the transistor. Though these effects are rather small, the mismatch between devices influences the behavior of the circuits. Especially in high sensitive analog circuits, the temperature behavior mismatch can result in inappropriate results. If halo implanted MOSFETs are used - normally in sub-100 nm transistors - the variability of doping concentration in the halo and the substrate can lead to this kind of mismatch. This effect is described more detailed in Chapter 7.

There are two types of transistor parameter mismatch that can nicely be used for PUFs: the threshold voltage (V_{TH}) mismatch and the mobility (μ) mismatch. These are the transistor parameters that show local variations due to statistical variations. In device simulation (BSIM model) this is realized by the variation of V_{TH0} and U0. Both variations have different influence on the drain current $I_D$in the different regions of the transistor.

In *weak inversion* μ variation has nearly no effect on I_D compared to V_{TH} variation. PUFs that are working in the sub-threshold region will be defined by the threshold voltage shifts of the involved transistors. This can also be seen in the equation for I_D in weak inversion (adapted from [145]):

$$I_D = \mu K1 \exp \frac{V_{GS} - V_{TH}}{K2}, \tag{6.4}$$

where $K1$ and $K2$ are the parts of the equation that are independent of μ and V_{TH}, and $V_{GS}$is the gate-source voltage. Whilst I_D is linearly related to μ, the influence of V_{TH} is in the exponential part of the equation.

In *strong inversion* the effect of mobility variations becomes larger. In the *linear region*, the drain current is linearly dependent on V_{TH} and on μ. This can be seen in the simple equation for drain current in linear region:

$$I_D = \mu K3[(V_{GS} - V_{TH})V_{DS} - \frac{V_{DS}^2}{K4}], \tag{6.5}$$

where $K3$ and $K4$ are terms that show no local mismatches, and V_{DS} is the drain-source voltage. The influence is also shown in Figure 6.11.

In *saturation region* the effect of V_{TH} variation increases again. μ still is related linearly to I_D whereas V_{TH} is in a quadratic term (equation without channel length modulation):

$$I_D = \mu K3(V_{GS} - V_{TH})^2 \tag{6.6}$$

Figure 6.11 shows the effect of μ and V_{TH} variability on I_D in strong and weak inversion, as well as linear and saturation region.

To be able to develop stable PUFs it is very important that the PUF will behave identical within a pre-defined region of operation. Since transistors are usually compared within a PUF to produce a output, the involved transistors should show the same relationship across that region. In Figure 6.12 the normalized I_D versus V_{GS} is shown for the linear and the saturation region. The graphs were produced by Montecarlo simulations of the same NMOS minimal size transistor. It can be seen that some of the curves cross at some point of the graph. A crossing of the curves means that the output of a PUF will change at the point of crossing if I_D is involved in the decision. Since the probability of crossing has to be minimized, it is important to minimize the variability of V_{GS} within the defined region of operation. It is also worth to mention that in the linear regime more crossings occur due to the higher influence of mobility variation.

Figure 6.13 shows the same normalized I_D but this time versus temperature. In the graph no crossings occur. The reason for this behavior is the inaccurate model that was used for simulation. In the model no temperature coefficient mismatch is modeled and thus all curves stay in parallel. As shown in the subsequent section temperature coefficient mismatch shows up in reality and generates errors at the PUF output.

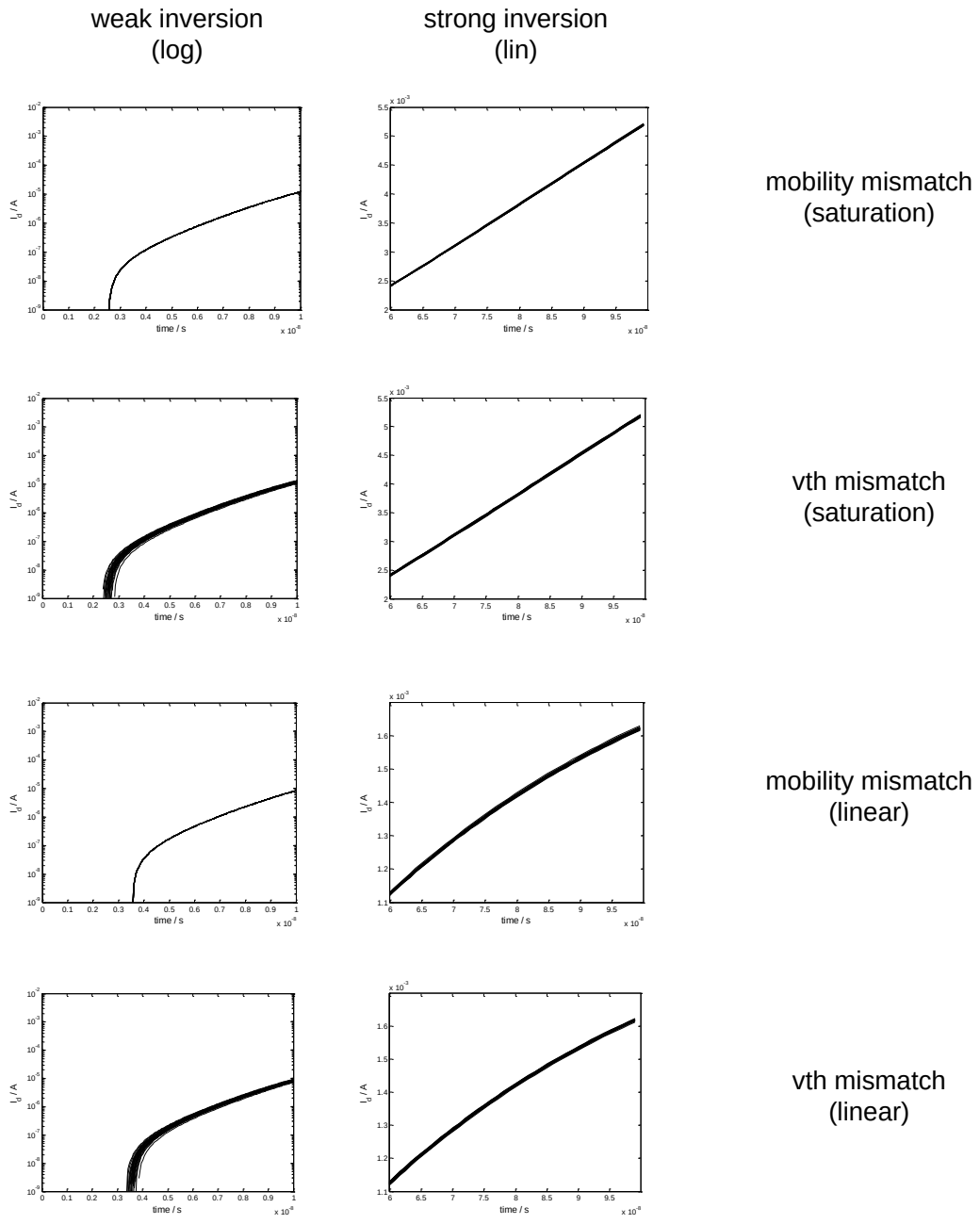


Figure 6.11.: The influence of μ and V_{TH} variability on I_D is shown for different regions of operation. It illustrates the dominant effect of V_{TH} variability on circuit mismatch. The data are from simulation using the BSIM3 model.

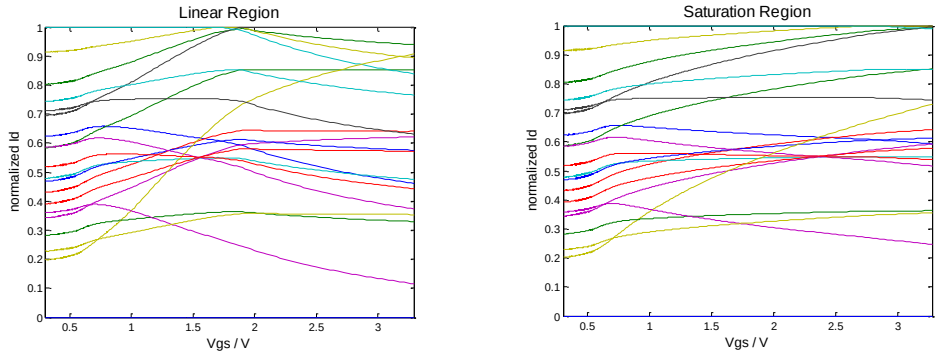


Figure 6.12.: The normalized I_D vs V_{GS} in linear (left) and saturation (right) region is shown. This graph is important to show the occurence of crossings at different V_{GS}. These crossings may produce errors at the PUF's output if V_{GS} changes during operation.

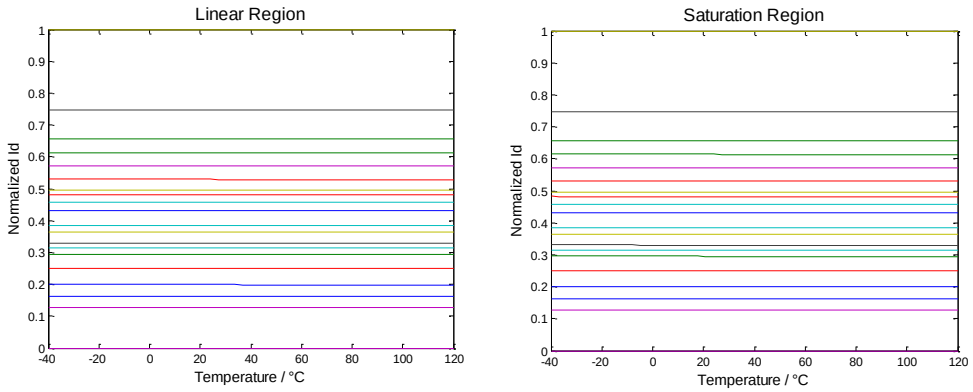


Figure 6.13.: The normalized I_D vs temperature in linear (left) and saturation (right) region is shown. In the used BSIM model no temperature coefficient mismatch is included and thus no crossings occur.

In addition to the errors according to temperature coefficient mismatch errors can occur if current mirrors are involved in the circuit that provide a bias voltage to the mismatch transistors. A sketch of such a circuit can be seen in Figure 6.14.

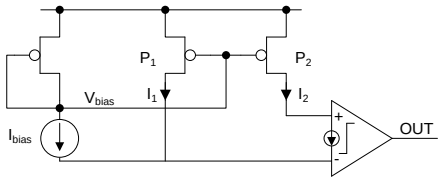


Figure 6.14.: A current source PUF with current comperator is shown. A current mirror is used for the biasing.

If the current mirror is fed by a temperature independent current source, the V_{GS} of the current mirror will change. Figure 6.15 shows the normalized I_D versus the temperature of a circuit including current mirror. In the case of the circuit V_{GS} varies from 0.96 V to 1.06 V. Thus, the effect on the error rate is present but limited.

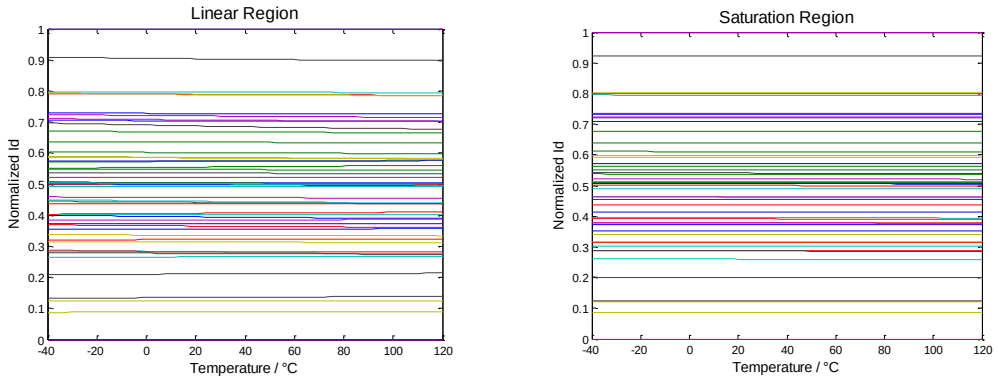


Figure 6.15.: The normalized I_D vs temperature in linear (left) and saturation (right) region of a PUF circuit with current mirror is shown. Due to the temperature dependence of the current mirror, V_{GS} of the mismatch transistors changes and thus errors may occur.

In a real circuit this kind of temperature dependence can be compensated by varying the bias current with temperature. Thus, V_{GS} at the mismatch transistors stays constant. Figure 6.16 shows the effect of constant V_{GS} and constant I_D during a temperature shift. Concerning PUFs constant V_{GS} must be preferred. Indeed, the current flowing through the transistors decreases due to temperature increase but a constant current leads to a shift in V_{GS} over temperature and this leads to errors as shown in Figure 6.12.

Figure 6.17 shows the effect of bias current variation at -40°C measured at the first PUF test chip. Although the reference vector was determined with 16 µA at 27°C the minimum error at -40°C shows up at 22µA. The higher bias current compensates for the V_{GS} negative shift caused by the temperature shift. It is important here to guarantee the functionality of the circuit (sense amplifier), even if the current changes. In the simulation the current changes from 23 µA to 19 µA

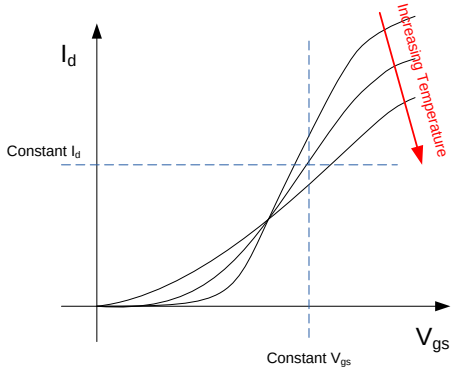


Figure 6.16.: The temperature behavior of MOS transistors is shown. If I_D is fixed over temperature, V_{GS} will change and the other way round. Since PUFs are sensitive to V_{GS} changes, it is helpful to fix V_{GS} in current source driven PUFs.

in the range from -40°C to -120°C which corresponds to a decrease in current of 17.39 %.

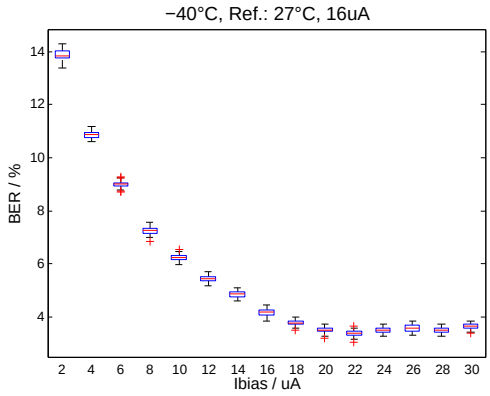


Figure 6.17.: If the bias current is adjusted, the error rate can be reduced. At -40°C the minimum error rate appears not at 16 µA but at 22 µA.

6.1.4. Noise in PUF Circuits

Noise is generated in all electrical circuits and influences their behavior. Since PUF circuits measure small mismatches between circuit components noise can easily influence the decision process. For this reason, it is important to understand the noise generating effects which are causing the troubles during circuit operation. In addition to noise that is generated by noise at the terminal signals (gate, source, drain, bulk) of the involved transistors, it is also generated inside the devices. There are several types of noise which are described in the following sections.

Thermal Noise

Thermal noise is generated by scattering between charge carriers in thermal motion. In circuits, thermal noise is produced in the resistive parts. Up to high frequencies - in the Terra hertz region

- thermal noise has a white spectrum , i.e. a flat power spectral density. The density S_{vt} is [145]

$$S_{vt} = 4kTR. \tag{6.7}$$

Here, k is Boltzmann's constant, T the temperature in degree Kelvin, and R the resistivity. In transistors thermal noise is produced mainly at the source and drain terminals.

In short channel devices, thermal noise gets clearly dependent on gate-source voltage and drain-source voltage. Figure 6.18 sketches that effect. Due to hot electrons the thermal noise increases for high V_{DS}.

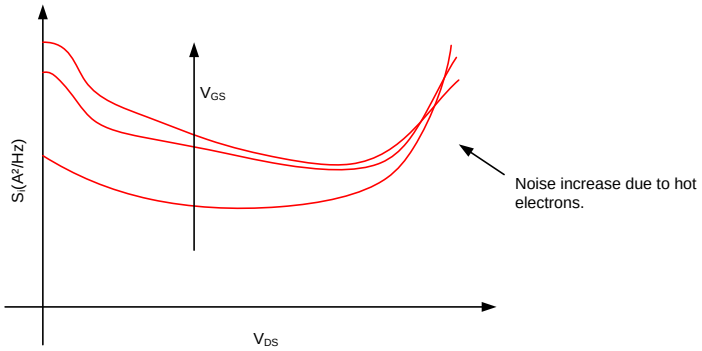


Figure 6.18.: In short channel devices thermal noise increases for high values of V_{DS} caused by occurence of hot electrons. The figure is inspired by [145].

Shot Noise

Shot noise is noise that is produced when charge carriers have to overcome a potential barrier like within transistors when moving from the source region into the channel. Since the crossing of the barrier is a statistic process, the current flow is not static but also 'noisy'. The power spectral density of shot noise with respect to the noise current is [145]

$$S_{is} = 2qI. \tag{6.8}$$

Here, q is the charge of the carrier and I is the average current flow. It can be seen that unlike thermal noise, shot noise is not directly correlated to the temperature.

Flicker Noise

The Flicker noise is due to trapping and releasing charges in traps near the Si-SiO_2 interface. The theories differ [145] but probably due to V_{TH} and mobility shifts noise occurs on I_D. This noise depends on the frequency linearly (1/f). The power spectral density can be estimated with [145]

$$S_{vf}(f) = \frac{K_1(V_{GS})}{C'^2_{ox}} \frac{1}{WL} \frac{1}{f^c}. \tag{6.9}$$

Here, $K_1(V_{GS})$ is a V_{GS} dependent component, C_{ox} the oxide capacitance per unit area, and the exponent c a constant between 0.7 and 1.2.

For short channel devices, the area of the gate gets so small that the influence of the single trap does not average out but will have a distinct effect on the noise behavior of the transistor. Therefore, in short channel transistors the 1/f characteristic does not hold. If only a few traps exist, even their position - e.g. with respect to the source and drain region - can influence the noise behavior. Flicker noise may also occur if no traps exist [145].

Random Telegraph Noise, Burst Noise, Popcorn Noise

As soon as the devices get small each single trap influences the behavior of the transistor. Random telegraph noise (RTN) occurs when the trapping of a single charge influences the drain current remarkably (up to some 10 percent [145]). This noise is a low frequency noise (under 100 Hz). In fact, Flicker noise is just the sum of a large amount of RTN. A figure of RTN is shown in Figure 6.19

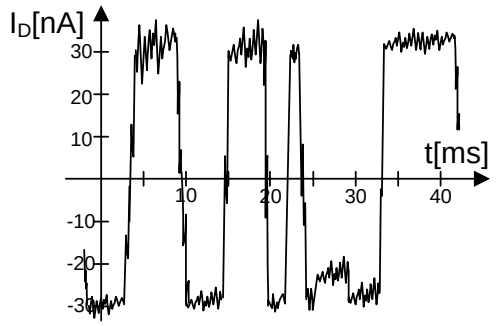


Figure 6.19.: Popcornnoise at different V_{GS}.

Influence of Noise on PUF circuits

Beside operating point shifts noise is an important source of error when in comes to PUF circuits. This is especially true for PUF circuits that do not amplify the mismatch between the transistors during a first amplification phase. In general, high frequency noise can be removed by filtering the signal. The cut-off frequency of the circuit depends on the individual demands, e.g. read-out frequency. The real problems are due to the low frequency components independent of the kind of noise. The noise components can influence the circuit for longer periods of time and therefore lead to errors. This problem becomes worse in the case of very small transistors. Since minimum size transistors should be used in PUF design to maximize the local mismatch, there is no way to get rid of the low frequency noise from the design perspective. The best way to reduce noise induced errors during read out is to average the results of multiple read-outs. Since popcorn noise components often last several milliseconds, the time between the read-outs should be chosen sufficiently long.

6.2. Other Sources of Mismatch

6.2.1. Resistor

The next microelectronic component that is analyzed is the resistor. Resistors are widely used in microelectronic circuits and are available in all analog processes. Therefore, resistors could be an interesting source of mismatch for PUF circuits. There are four types of resistors that are commonly used. Diffused resistors use doped silicon between two contacts. For low power and linear circuits ion implanted resistors are used. Thinfilm resistors are implemented by introducing a film of resistive material on the substrate. These transistors can be made very accurate by laser trimming. The last version is the polysilicon resistors. Polysilicon resistors are fabricated by implanting ions into a layor of polysilicon. These resistors show a high resistance but with low tolerance. For that reason, polysilicon resistors could be a good choice for PUF circuits. In a

typical CMOS process the polysilicon resistors vary locally about 10 %. Ten percent is a high value and could be ideal for PUF circuits. Unfortunately, there are two drawbacks: the minimum size of polysilicon resistors is about ten times higher than the minimum size of MOS transistors. That makes them less atractive since area plays such an improtant role. Furthermore, resistors produce thermal noise. This especially becomes a problem, if small currents are used which is important for low power circuits. For instance, a minimum size resistor has an impedance of 1 kΩ. If 1 µA is sent through it, the voltage drop is just 1 mV. Since the standard deviation is ±10 %, the mismatch between two resistors that has to be measured is about 0.1 mV which is farely small, especially, if the noise is taken into account. The thermal noise induced effective voltage drop ($U_{R,\text{eff}}$) of a 1 kΩ resistor at a room temperature gets

$$\begin{aligned} U_{R,\text{eff}} &= \sqrt{4kTR\Delta f} \\ &= \sqrt{4 \cdot 1.38 \cdot 10^{-23} \frac{\text{VAs}}{\text{K}} \cdot 300\,\text{K} \cdot 1000 \frac{\text{V}}{\text{A}} \cdot 10 \cdot 10^{9} \frac{1}{\text{s}}} \\ &= 0.4\,\text{mV}, \end{aligned} \tag{6.10}$$

where k is the Boltzmann constant, T the temperature, R the resistance, and Δf the noise bandwidth. In this example, the noise induced voltage drop exceeds the standard deviation of the mismatch. In such a case, not the mismatch but the noise would define the PUF's output. The problem could be solved by either increasing the current through the resistors or by filtering the noise. Increasing current would lead to higher power consumption which often is not acceptable. Filtering the noise would work well as long as the read-out frequency of the PUF does not play an important role. Thus, both approaches have to be well thought out.

In the following text two resistance-based PUF approaches are introduced. The first one is a very simple circuit which is shown in Figure 6.20). The current sources provide the same amount of current. Due to the mismatch between the transistors the voltage drop differs between the branches. The comparator detects the difference and produces the output.

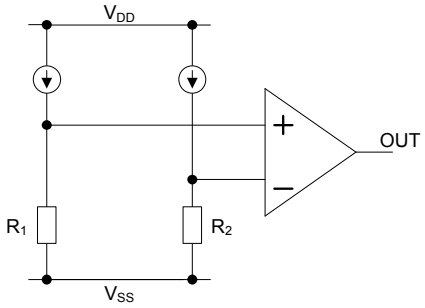


Figure 6.20.: Concept of a resistor-based PUF.

In Figure 6.21 another approach is shown. Figure 6.21a depicts the whole circuit. An output is produced within three phases. During the first phase (Figure 6.21b) the circuit is reset. It is important that the two NMOS transistors match well to make sure that their influence on the decision is as small as possible. In the second phase current starts to flow through the resistors. Due to the mismatch, the voltage drop differs between the two branches. The negative resistance of the cross coupled pair amplifies that mismatch. During the last phase a latch is activated which further stabilizes the output.

As a conclusion, resistor-based PUFs are not a good approach to PUF circuits. The big problems related to them are the area consumption and the occurring noise. Even hough the noise problem can be minimized resistors should not be the first choice compared to other microelectronic components.

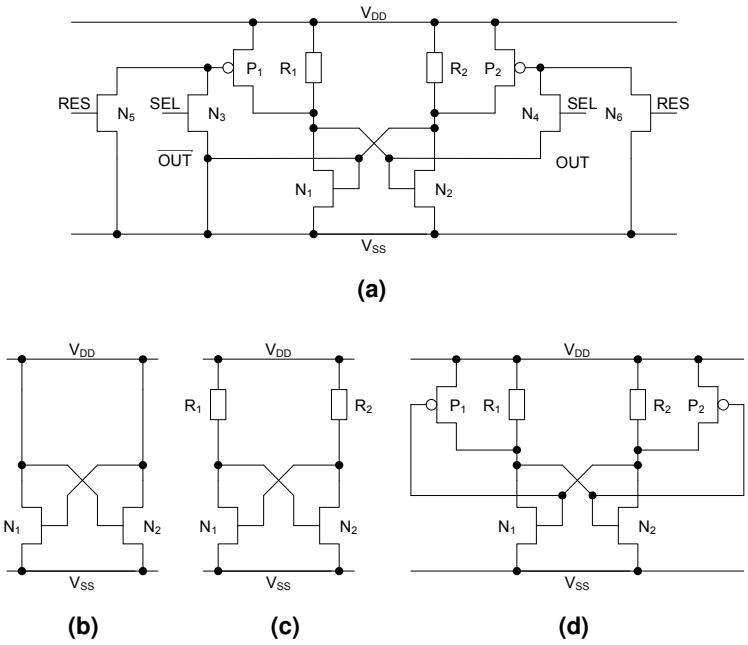


Figure 6.21.: Resistor-based PUF: (a) Complete circuit; (b) Reset; (c) Amplification; (d) Evaluation.

6.2.2. Capacitor

Another important component in microelectronic circuits are capacitors. Different types of capacitors are available in common technologies. Two important types are the poly capacitors and the metal-insulator-metal (MIM) capacitors. Both capacitors show local mismatches and therefore they are interesting for PUF purposes in general. Poly capacitors are built in two layers of polysilicon material having an insulating dielectric material in between. MIM capacitors are parasitic capacitors between adjacent metal layers. Of course, these capacitances can be exploited in circuit design especially if sufficient metal layers are available in the process. Since in modern processes, the number of metal layers commonly is high, MIM capacitors are widely used. Compared to poly capacitors, MIM caps show less variability. Due to weak dielectrics between the metal layers, MIM caps demand more area. This effect can be seen in Figure 6.24. MIM caps are not a useful choice for PUF purposes because of their area consumption. The properties of poly caps are not sufficient since the variability is still too small.

Nevertheless, two examples of capacitor-based PUF circuits are presented below. The first one is a SRAM-based version. The second one is similar to a ring-oscillator PUF.

In Figure 6.22a, the SRAM-based type is shown. During the power-up phase of the circuit, the two capacitors C_1 and C_2 are being loaded. If the other components of the circuit do match well, C_1 and C_2 define the output. If C_1 is larger than C_2, $\overline{\text{OUT}}$ increases slower than OUT. Due to the latch, this behavior is amplified and OUT moves towards V_{DD} .

Figure 6.22b shows the other capacitor-based PUF example. In this case the two mismatch capacitors are connected to a Schmitt-trigger that output is fed back to its input. Due to the differences of the capacitors the time of charging and discharging differs and so does the frequency of the two oscillators. The frequency is compared and the output is set either to '0' or to '1'. Here again, the size of the capacitances makes the approach unattractive. Furthermore, the frequency comparator is quite complex to implement and consumes area.

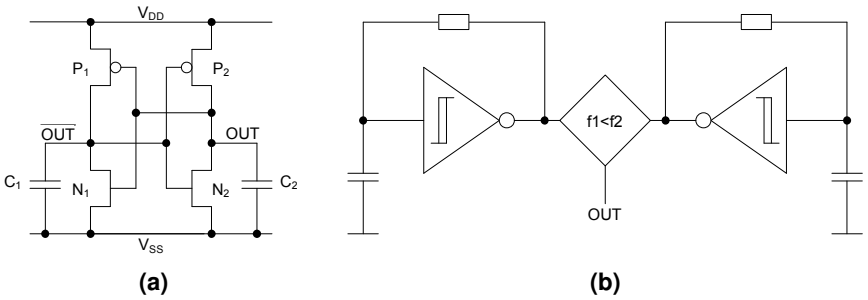


Figure 6.22.: Examples of capacitor-based PUFs: (a) SRAM with mismatch capacitors; (b) Capacitor-based PUF using Schmitt-Triggers.

6.2.3. Bipolar Transistors

Bipolar transistors are another important component in microelectronic design, if they are available. Many processes nowadays only offer MOS transistors. But generally, bipolar transistors show local mismatches and therefore they can be used for PUF purposes. Bipolar transistors mismatches are mainly caused by base and emitter junction area doping variations [57]. The effect of such variations on the collector current in a standard BiCMOS process can be seen in Figure 6.23. In Figure 6.23a the collector current vs. V_{BE} is shown. The variation gets evident for V_{BE} of about 0.9 V and higher. The standard deviation of the collector current vs. V_{BE} in Ampere (Figure 6.23b) and % (Figure 6.23c) reveal the optimum operation point of a bipolar transistor using it as mismatch transistors. At about 1V the standard deviation reaches its maximum value in both ways, absolutely and percentually. At this V_{BE} the error rate should reach its minimum value. If noise (range: 1Mhz to 1Ghz) is added to the circuit (Figure 6.23d), it can be seen from the comparisson of the noisy signal (red curve) and the clean signal (blue curve), that at about 0.8 V and below the noise defines the mean output value. That is the point where the circuit will not work correctely since mainly noise defines the output and not the mismatch between the transistors.

There is no reason why bipolar transistors should not be used for PUF purposes as long as the operation point is chosen in a way to maximize the local mismatch and at the same time minimize the influence of noise. This is always a trade-off especially when power-consumption is getting relevant.

6.2.4. Inductor

Also inductors are available as microelectronic components. There are different types like square planar, multi layer, or coupled inductors [77]. Even if inductors show local mismatches, inductors are large and not applicable for PUF circuits with large numbers of mismatching pairs. Thus, inductors are not taken into account as a source of mismatch.

6.3. Summary

To sum up the preceding chapter, the different microelectronic components are compared with respect to their suitability for mismatch sources in PUF circuits. There are two parameters that are important. Firstly, the size of the component is important since a PUF circuits should be as small as possible to make it cheap. Secondly, the average amount of component mismatch should be as high as possible to reduce the probability of errors. Figure 6.24 shows the different components that were discussed above except the inductance which is considered for the reasons

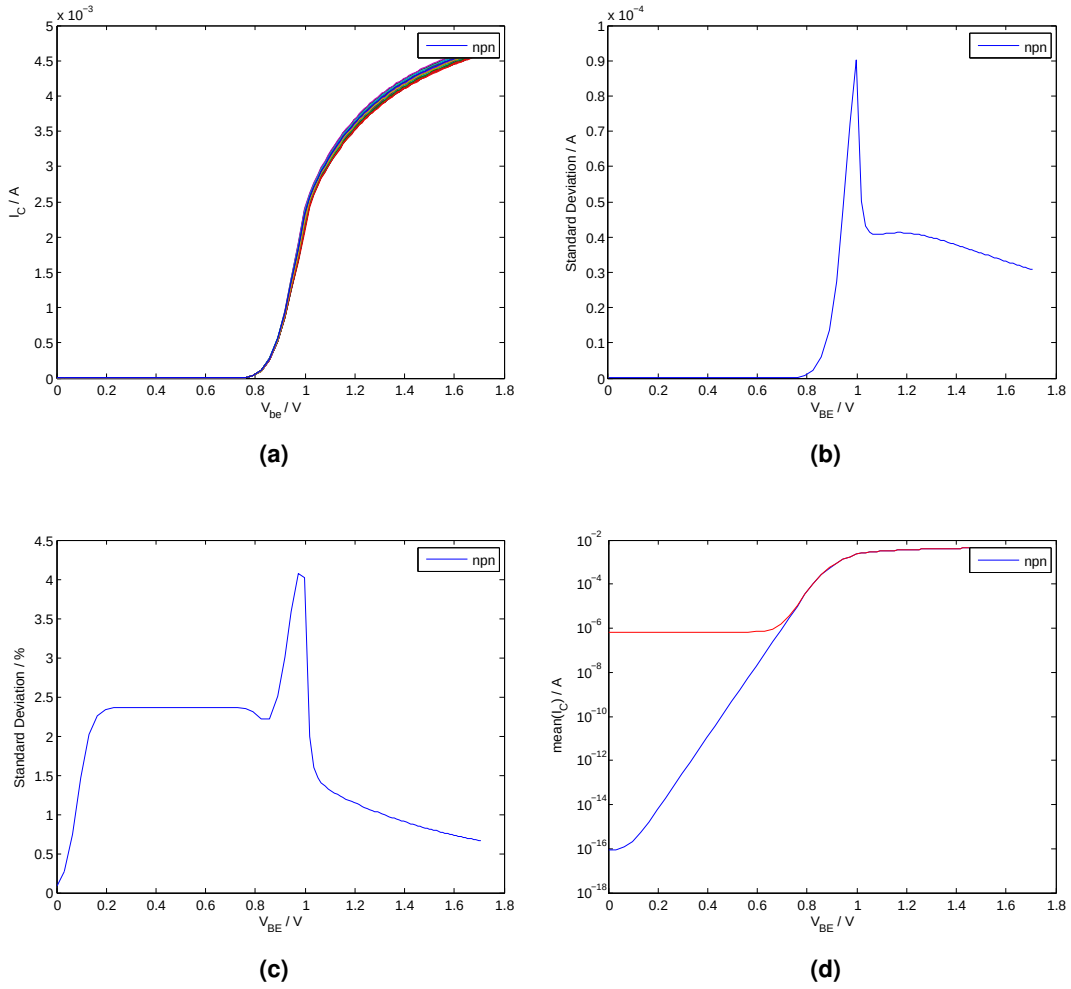


Figure 6.23.: Influence of mismatch at different V_{BE}: (a) I_C vs. V_{BE} of different Montecarlo runs. (b) Standard deviation of I_C vs. V_{BE} in A. (c) Standard deviation of I_C vs. V_{BE} in %. (d) I_C (blue) and I_C plus standard deviation caused by transient noise (red curve).

described above. On the abscissa, the area demand of the minimum size component is shown. The ordinate shows the relative standard deviation. As already seen in Figs. 6.10 and 6.23, the amount of mismatch of the transistors varies depending on the operation point. Nevertheless, a trend is visible. In the presented process the MOS transistors have to be preferred over the others since they show the highest mismatch and the smallest size. The bipolar npn-transistor also behaves quite good. The residual elements should not be taken into account, if possible.

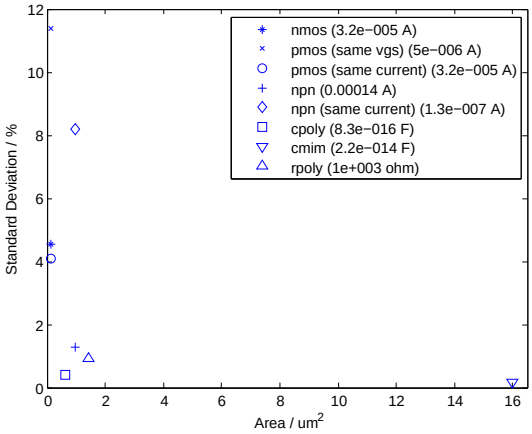


Figure 6.24.: Typical mismatch of different components.

7. Refine Models for PUF Simulation

by Christoph Boehm

7.1. Introduction

An ideal PUF does not produce any errors in a defined region of operation. Thus, an ideal PUF returns the same number every time as long as it works in this region. The region is mainly defined by a supply voltage range and a temperature range. Unfortunately, a real PUF does not have these properties. Due to the Gaussian distribution of the mismatch between the involved transistors, there always exist PUF cells which have very small mismatches between the transistors. The probability is high for these cells so that a shift in operation region will change the output. Moreover, a small mismatch increases the impact of noise on the decision process. Thus, a real PUF always exhibits a certain bit error rate. If no errors at the PUF's output are allowed, these errors have to be detected and corrected. This is especially important if the PUF output is used for key generation. In this case a single bit error completely changes the output of the cryptographic function which is not acceptable. Error correction codes are used to solve the problem. During the design of the PUF it is important to estimate the future error rate as accurate as possible. If good simulation results are available the error correction code can be defined in a way to minimize its complexity still meeting the correction requirements. The simpler the error correction gets the smaller the area, the power, and the time consumption of the blocks will be. To be able to estimate the error rate, Montecarlo mismatch simulations are used. These simulations consider the local mismatches between the involved devices like transistors, capacitances or resistors. This is done by simulating the same circuit using different values of device parameters. Often the BSIM parameters for threshold voltage (V_{TH0}) and mobility (U0) are varied for the MOS transistors. The degree of variation changes between different types of transistors or different processes and is defined by modeling experts. The variation is done by using Gaussian distributions.

To estimate the error rate of PUFs Montecarlo simulations are used in combination with shifts of supply voltage or temperature. Unfortunately, the BER estimations do not match the measurement results of a real PUF circuit. Even though errors produced by V_{DD} shifts match between simulation and reality the estimated temperature induced error rate is too small. Figure 7.1 depicts the difference between measurement and simulation. Figure 7.1a shows the measurement result: The error rate increases from around 0 % up to 10 %. Furthermore, the increase is not linear. The outcome of the Spectre simulation (Figure 7.1b) shows a far smaller error rate of a maximum of about 2 %. The increase shows no nonlinear characteristic as the square-root-like characteristic of the measurement.

A wrong BER estimation might have serious consequences, if the error correction block, which was developed based on the simulation results, does not fulfill the real requirements.

It turns out that additional to the local variations of the threshold voltage and the mobility also the temperature coefficients change from device to device. In the following chapter the temperature behavior of MOS transistors is analyzed and a modeling approach is presented.

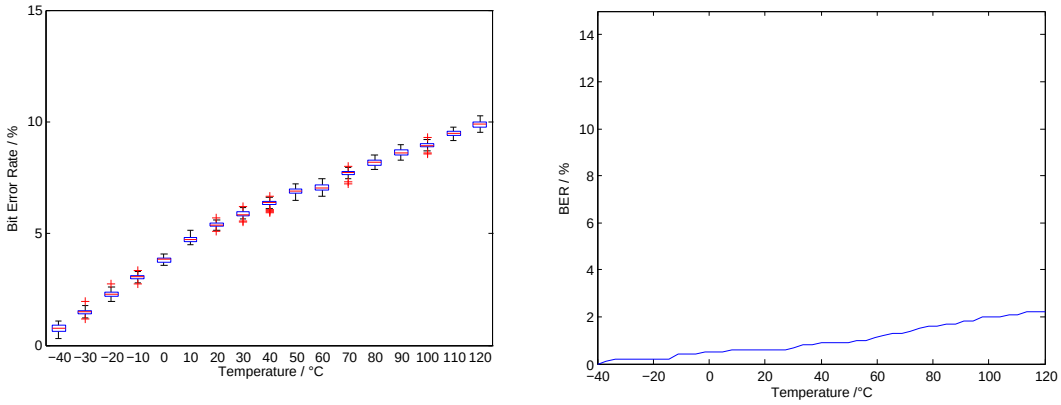


Figure 7.1.: Huge differences between the BER vs. temperature of a real chip (a) and a simulated circuit (b). The reference vector was chosen at -40°C.

7.2. Temperature Dependence of PUF Outputs

As all circuits PUFs change their behavior when temperature changes. In general, this should be no problem since a PUF returns a digital signal which is the result of some comparison. If the compared elements change their behavior in the same way, the result of that comparison should not change and thus no errors should occur. Unfortunately, there are two reasons why PUFs do not behave like that. Firstly, if temperature changes, the shift in the operation point may lead to a different relation between the compared elements. Secondly, the involved transistors do not just have the desired parameter mismatch, but also have a mismatch in their temperature behavior which as well may lead to an increased error rate. Both problems are described more detailed in the following section. Before that a simple exemplary PUF circuit, which builds the basis for further analysis, will be introduced.

7.2.1. Exemplary PUF Circuit

To analyze and model the temperature behavior in detail, a exemplary PUF circuit is used. Concerning its simplicity a current PUF is chosen. This is done without loosing any generality. More complex PUF circuits just involve more transistors which influence the overall behavior. The current PUF is depicted in Figure 7.2.

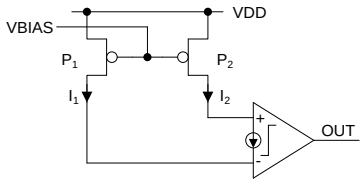


Figure 7.2.: Circuit which describes the functionality of a two transistor PUF using an ideal current comparator.

The two PMOS transistors P_1 and P_2 build the core of the PUF. They have to be designed in a way to maximize their local mismatch. Thus, these transistors are minimum in channel length and

width. The voltage between the supply voltage V_{DD} and the bias Voltage V_{BIAS} forms the gate-source voltage V_{GS} of the two transistors. Since V_{GS} is identical at both transistors, the drain currents are equal in the case of perfect matching. Due to the mismatch between the transistors the currents differ. The current comparator detects the higher current and sets its output depending on the decision. If $I_1 > I_2$, OUT is set to HIGH. If $I_2 > I_1$, OUT gets LOW. In the following analysis all components of the PUF except P_1 and P_2 are assumed to be ideal. Therefore, the analysis concentrates on the two crucial devices.

7.2.2. Shifting the Operation Point

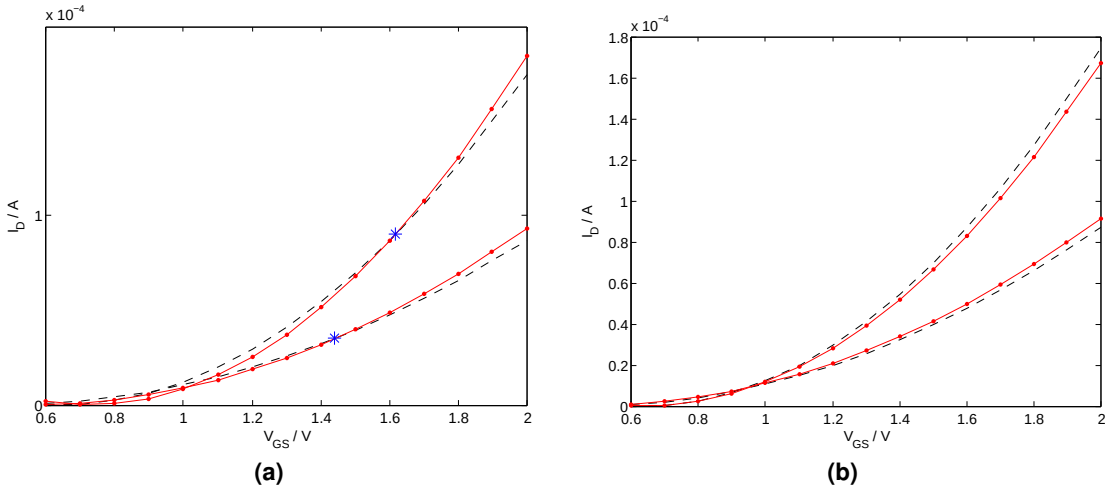


Figure 7.3.: Sources of temperature dependent errors. (a) The crossing (blue marker) moves depending on the temperature. (b) I_{D1} (solid) and I_{D2} (dashed) at two different temperatures.

Assuming ideal components, V_{BIAS} stays constant, even if the temperature changes and so does V_{DD}. Thus V_{GS} of the two mismatch transistors stays constant over temperature. This assumption is not far from reality since a constant V_{BIAS} can be realized by using constant voltage sources that are temperature compensated. If the two mismatch transistors match too well for PUF purposes and their I_D vs.V_{GS} characteristics are similar, it may happen that the I_D curves cross at some V_{GS} as it is shown in Figure 7.3a. This may happen, if mobility mismatch exists. If the temperature is increased, the mobility decreases and thus the two I_D vs. V_{GS} curves become flatter. This change of current behavior of the transistors causes a change of the position of such a crossing as indicated by the small markers in Figure 7.3a. Since V_{GS} is forced to be constant over temperature, it may happen that the crossing of the curves passes the point of the actual V_{GS} when temperature changes. Thus, the transistor with the initially higher current now becomes the weaker transistor. The output of the comparator changes and a temperature induced deterministic error occurs. These errors are common and also appear in the simulation. They produce the linear increase in error rate which can be seen in Figure 7.1b. In the case of the example circuit the error rate varies about 2 % due to such errors.

7.2.3. Mismatch of Temperature Coefficient

The second type of deterministic errors that can occur is the errors due to temperature coefficient mismatch. These errors arise from the fact that not only the parameters show a certain variability but also their temperature coefficients. In the case of MOS transistors the parameters that vary

due to local mismatch are the threshold voltage parameter V_{TH0} and the mobility parameter U0. Both have temperature coefficients. In the BSIM model the coefficients are called KT1 and UTE. The effect of mismatching KT1 and UTE on the temperature behavior of a circuit like the one in Figure 7.2 can be seen in Figure 7.3b. At a low temperature (upper curves) the transistor that is represented by the solid line provides less current than the second transistor for V_{GS} higher than about 0.9 V. If the temperature increases, the curves become flatter. The degree of change depends on the temperature coefficients. In the case of Figure 7.3b the coefficients of the two transistors are not the same and therefore at some point in temperature the second transistor starts to provide more current. The output of the PUF changes. Since this error always occurs, if the crucial temperature is crossed, this error is a deterministic error. Looking at the PUF circuit in Figure 7.2 the situation can be described as follows: P_2 provides more current than P_1 and thus a '0' appears at the output. If the temperature increases, P_1 suddenly gets stronger which results in a '1' and thus a error occurs. The problem is depicted in Figure 7.4.

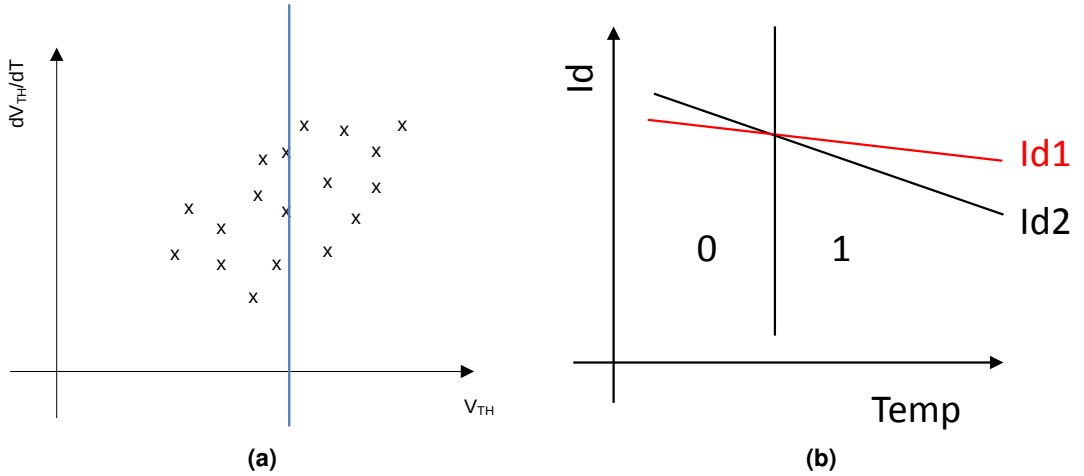


Figure 7.4.: (a) Sketch of measurement results: same V_{TH} but different temperature coefficients. (b) If the temperature coefficients differ, crossings cause errors: OUT=0 (left), OUT=1 (right).

Figure 7.4a is a sketch of the measurement results of real transistors. V_{TH} versus the absolute value of its temperature coefficient dV_{TH}/dT is depicted. Two properties stand out. Firstly, there is a correlation between V_{TH} and dV_{TH}/dT. Secondly, there are transistors having the same V_{TH} but different dV_{TH}/dT. That means that there does not only exist a mismatch of V_{TH} but also dV_{TH}/dT varies between the transistors. The effect of this phenomenon can be seen in Figure 7.4b: as soon as the temperature of the crossing is passed due to a temperature shift, the PUF output changes deterministically. Compared to other analog circuits PUF circuits are especially sensitive concerning temperature coefficient mismatches since PUFs measure the small differences between transistors and naturally therefore react on small changes. Since temperature coefficient mismatch is not included in the models these kind of errors do not occur in the simulation and thus the estimated error rate gets too small.

The residual chapter deals with this kind of errors. The reasons for them are analyzed more detailed. Finally, temperature coefficient mismatch explains the high error rate which occurs in real circuits (Figure 7.1a).

7.3. Properties of a Halo-Implanted Transistor

Local mismatches of transistors are caused by manufacturing variations. There are different kind of variations which include the final dimensions of the transistors, the thickness of the gate oxide, or the doping concentration in the different regions of the transistor. It turns out, that the doping concentration inside the region between the gate oxide and the bulk contact has crucial impact on the behavior of the transistor and is responsible for a big part of the local mismatch between transistors. This mainly happens due to the fact, that the doping concentration in this part is involved decisively in defining the threshold voltage and partly in the mobility in the channel region. Moreover, the doping concentration cannot be defined exactly since due to scattering effects during implantation the doping atoms are distributed following a Poisson distribution. Thus, the distribution of the doping atoms varies by definition between the single transistors which finally builds-up the foundation of most silicon PUFs. Therefore, in the following section a typical PUF transistor is introduced which is later used to explain the effect of doping concentration variation and its influence on the temperature behavior of PUF circuits.

PUF circuits need to show strong local variations in order to allow a preferably stable behavior. If it is possible therefore, small technologies are used to increase the local mismatch between the transistors which results mainly from the fact that a mismatch in the small number of doping atoms has a big effect compared to transistors with large numbers. In this case a technology is chosen that allows a minimum channel length of 60 nm. To stabilize the threshold voltage also for short channel devices the so-called halo or pocket implants are used. In an additional processing step the halos are implanted into the transistor on top of the substrate/well doping to build a region of higher doping concentration. The process of implantation can be seen in Figure 7.5.

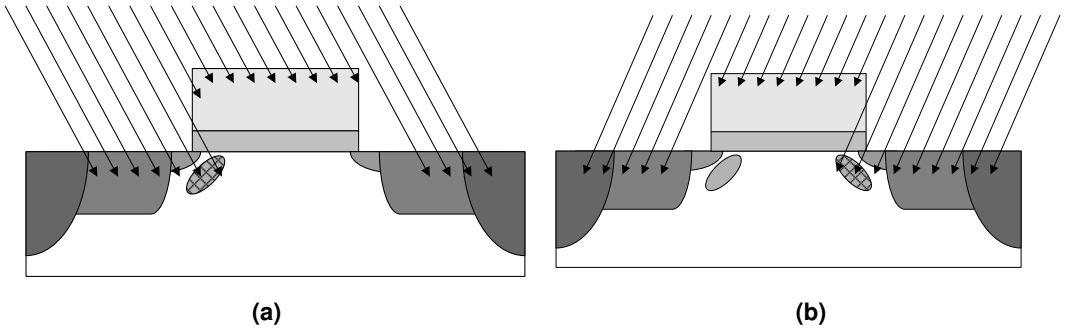


Figure 7.5.: Introducing the halo implant during processing [26].

The implantation energy is chosen in a way that the doping atoms settle in the channel region near the source and drain extension. The extra doping shifts the threshold voltage to the preferred value. Especially, the threshold voltage drop for small channel devices is compensated by the halo implant.

As mentioned above, small channel lengths are needed for PUF devices. Thus, in the case of the example transistor the minimum channel length is chosen which is 60 nm. For minimum size transistors the halo implants do not build-up two different regions but touch or even overlap each other. Now, there are not three but only two regions left which define the doping profile bellow the oxide vertically. One region which lies under the channel is defined by the halo doping and the substrate/well doping. This region is called *channel region*. The second region lies bellow the first one and is defined only be the substrate/well doping concentration. This region is called *substrate region*. The doping profile of such a minimum channel transistor can be seen in Figure 7.6.

To determine the doping concentration in the channel region (DC_{ch}) the doping concentration

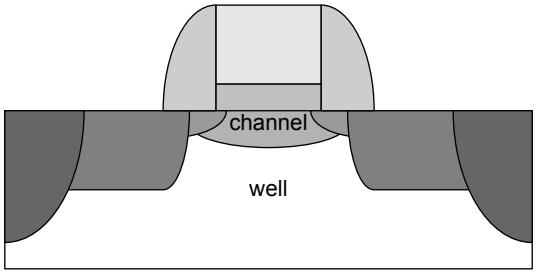


Figure 7.6.: For short channel transistors, the halo implants touch/overlap and a homogeneous channel doping concentration is formed.

of the substrate (DC_{sub}) and the doping concentration of the halo implant (DC_{hal}) have to be summed up. In the region bellow the channel region the doping concentration will stay constant independently of the halo doping concentration. In the case of the example transistor the nominal substrate doping concentration is set to 1.0E18$\frac{1}{\text{cm}^3}$. The nominal halo doping concentration is chosen to be 0.5E18$\frac{1}{\text{cm}^3}$. Thus the nominal doping profile bellow the oxide of the example minimum channel transistor can be determined as follows:

$$\begin{aligned} DC_{ch} &= DC_{sub} + DC_{hal} \\ &= 0.5E18/cm^3 + 1.0E18/cm^3 \\ &= 1.5E18/cm^3 \\ DC_{sub} &= DC_{sub} \\ &= 0.5E18/cm^3 \end{aligned} \tag{7.1}$$

The example above shows that DC_{sub} can be replaced by DC_{well} for the case of PMOSTs or NMOSTs in a triple well process. This transistor can be simulated in the TCAD environment. TCAD determines the transistor properties using finite element methods to solve the equations. Thus, very accurate analysis can be performed using this tool. The mismatch behavior of the transistor is simulated providing different doping profiles to the simulator. In this case, DC_{sub} and DC_{hal} were altered by up to 20 % to simulate local mismatches. Since these are also the parameters that are varied during common Montecarlo mismatch simulation, the threshold voltage V_{TH} and the electron mobility μ was obsered during the simulations. A sketch of the result of the analysis is shown in Figure 7.7.

The figure shows the characteristic behavior of V_{TH} and μ if either the substrate doping concentration or the channel doping concentration is varied. In itself, both variations follow the same logic. V_{TH} increases as soon as either region gets higher doped. μ behaves differently. It decreases as soon as one of the regions' doping concentration increases. Nevertheless, the effect on V_{TH} is stronger if DC_{ch} is increased. The effect on μ is stronger, if DC_{sub} is increases (the detailed TCAD analysis is presented later). This result is important since it visualizes the basic problem. Due to the two different slopes depending on the region that is altered there are combinations that show same V_{TH} but different μ. And the other way around: There exist doping profiles that show different V_{TH} but the same μ. The first situation is indicated by the dashed line in Figure 7.7. Furthermore, such transistors also have different temperature behavior. Regarding such combinations it happens that there exist transistors which show very similar behavior. These transistors may show crossings between their I_D curves either at some V_{GS} or at some temperature. The influence of doping concentration mismatch at the temperature behavior can also be seen at a TCAD simulation result of two different transistors which can be seen in Figure 7.8. The threshold voltage of the two transistors is chosen to be minimal. This could be reached by altering DC_{sub}

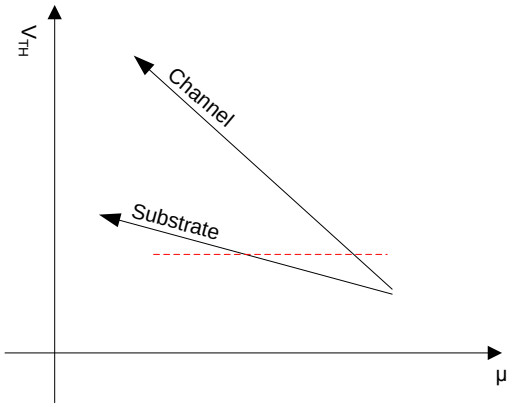


Figure 7.7.: Effect of doping concentration variation on V_{TH} and μ of a halo-implanted short channel transistor. Different combinations can cause the same V_{TH} (dashed line).

by a factor that is minus two times that altering factor of DC_{hal}. The used concentrations can be seen in Tab. 7.1. In this configuration V_{TH} stays nearly constant but the temperature coefficients differ strongly which explains the altered relation between the currents of the two transistors in Figure 7.8. At -50 °C the transistor, that is represented by the solid line, provides the higher current. At 27 °C and 120 °C the dotted line dominates. Thus, the PUF output would change and produce an error.

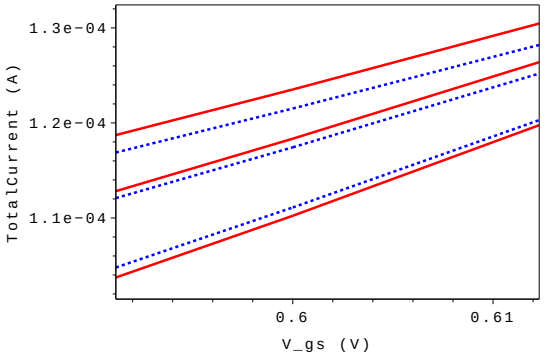


Figure 7.8.: TCAD: Current output of two different transistors at three different temperatures (bottom-up: -50°C, 27°C, 120°C). At -50°C, the sign of the ratio between the currents changes which would lead to an erroneous PUF-cell output.

In the example above the problem of halo-implanted devices can be seen clearly. Due to the doping concentration mismatches between transistors and the fact that different doping profiles may lead to very similar transistors the PUF behavior can be unpredictable. Crossings over the temperature and the gate source voltages occur that produce deterministic errors in the PUF cell. And not only PUF cells but also other circuits that depend on good matching over a wide temperature range are affected. For example, constant voltage sources that rely on matched transistors suffer from this temperature dependence. To get a better understanding of that problem the next section shows more results from the TCAD analyses of a minimum channel length halo-implanted transistor in a 60 nm technology.

Table 7.1.: Doping concentration of example transistors.

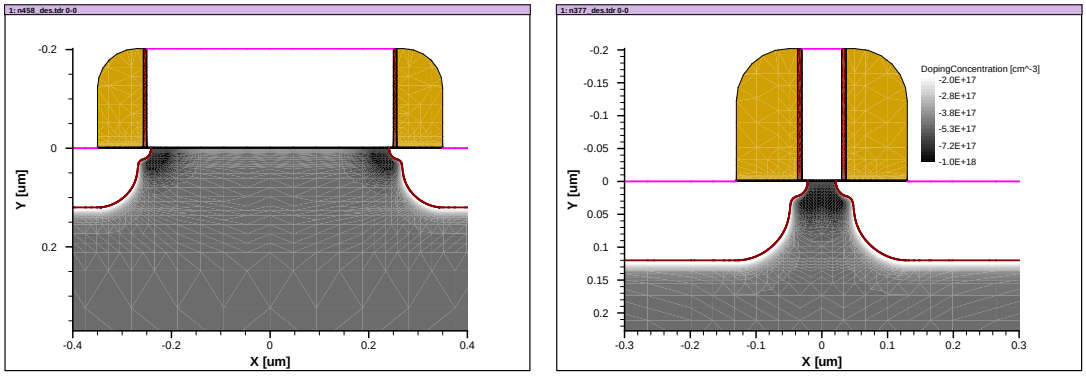
Doping concentration	Transistor 1	Transistor 2
$DC_{hal}/\frac{1}{cm^3}$	1.2E18	0.9E18
$DC_{sub}/\frac{1}{cm^3}$	0.4E18	0.6E18
$DC_{ch}/\frac{1}{cm^3}$	1.6E18	1.5E18
$DC_{w}/\frac{1}{cm^3}$	0.4E18	0.6E18

7.4. TCAD Analysis

7.4.1. Problem

To understand the temperature behavior of PUF circuits it is interesting to analyze the temperature behavior of the involved transistors more detailed. In the case of PUFs the used transistors should provide maximal mismatch to stabilize the behavior of the cells. From the designer's perspective this can be done by minimizing the size of the transistors. Thus, in PUF design the crucial transistors are in general minimum size devices. Furthermore, in modern technologies so-called halo-implants or pocket-implants are used to stabilize the threshold voltage also for short channel devices. For that reason TCAD analyses of halo-implanted minimum length transistors are presented in the following section. The results should give useful hints for deeper circuit understanding.

The doping profile of long channel device (1 µm) can be seen in Figure 7.9a.



(a) (b)

Figure 7.9.: (a) Long channel device (1 µm) and (b) minimum channel device (60 nm) with halo implants.

At the transitions to the extensions of the source and the drain regions the higher doping concentration can be seen as two pocket-like regions. When reducing the channel length to the minimum size the two halo regions touch and the doping profile looks like as it is shown in Figure 7.9b. Now, the halo implant defines the channel doping concentration (a closer look at the doping profile can be taken in Figure 7.10).

Thus, it is the doping concentration variation of the channel and the well which define the behavior of the transistors. By varying the doping concentration of the two regions in the TCAD simulation the reaction of the transistor on the variations can be analyzed.

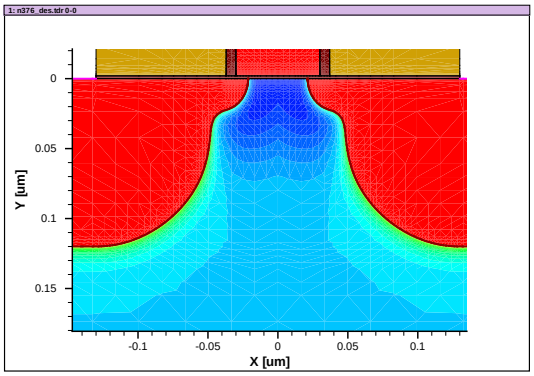


Figure 7.10.: Doping profile of a halo-doped minimum channel device.

Since mainly V_{TH} and the mobility μ define the temperature behavior of the transistors and which also are the parameters that are varied during Montecarlo mismatch simulations they are the parameters that are considered in the analysis.

The nominal values of the doping concentration are 1.0E18 $\frac{1}{\text{cm}^3}$ for the halo implant and 5E17 $\frac{1}{\text{cm}^3}$ for the substrate. Thus the channel doping concentration DC_{ch} gets 1.5E18 1/cm^3. The residual region is defined by the substrate doping concentration alone: 5E17 $\frac{1}{\text{cm}^3}$ (see Eqns. 7.2). During the mismatch analysis, the doping concentration of the channel and the substrate region was altered. The channel region took on values from 1.4E18 $\frac{1}{\text{cm}^3}$ up to 1.65E18 $\frac{1}{\text{cm}^3}$. The doping concentration in the substrate was changed between 4.0E17 $\frac{1}{\text{cm}^3}$ and 5.5E17 $\frac{1}{\text{cm}^3}$. Thus DC_{ch} was changed maximally by 10 % and DC_{sub} by maximally 20 %. The results of the analysis of this transistor are presented within the following section. The pure TCAD results are partly analyzed in Matlab. The first part is on the threshold voltage and its temperature coefficient. The second part is on the mobility and its temperature behavior. Unlike the threshold voltage, mobility is not available as a single value for the whole transistor but it is evaluated at different points of the transistors. Therefore, quantitative results are hard to receive. Nevertheless, a qualitative analysis is useful for both, understanding and model improvement.

7.4.2. Effect of Doping Concentration Variation on the Threshold Voltage

Different configurations of doping concentration variation of the channel and the substrate region were simulated in TCAD to analyze its effect on the threshold voltage and the temperature coefficient of the threshold voltage. The results can be seen in Figure 7.11.

Figure 7.11a shows the effect on the threshold voltage. As expected, V_{TH} increases, if either the halo or the well doping concentration is increased. Since V_{TH} depends on both doping levels, there are situations in which the same V_{TH} results from different doping combinations. The same statement is true for the temperature coefficient which can be seen in 7.11b. As shown in the case of V_{TH}, $\frac{dV_{TH}}{dT}$ depends on the doping concentration of the two regions. The absolute value of $\frac{dV_{TH}}{dT}$ increases as soon as the doping concentration increases. That means that V_{TH} and $\frac{dV_{TH}}{dT}$ are clearly correlated. This is already an important result of these simulations since this behavior is not included in the actual transistor mismatch models. The correlation between V_{TH} and the doping concentration in the regions can be seen Figure 7.11. The plot in Figure 7.11a is separated. Figure 7.12a illustrates the correlation between V_{TH} and DC_{ch}. Figure 7.12b depicts V_{TH} vs. DC_{sub}.

Examples of same V_{TH} but different doping profiles are marked by the solid line. The results in Figure 7.11 make clear that a change of the channel doping influences V_{TH} more than a change in the substrate doping. As expected, the channel doping concentration dominates the transistor's

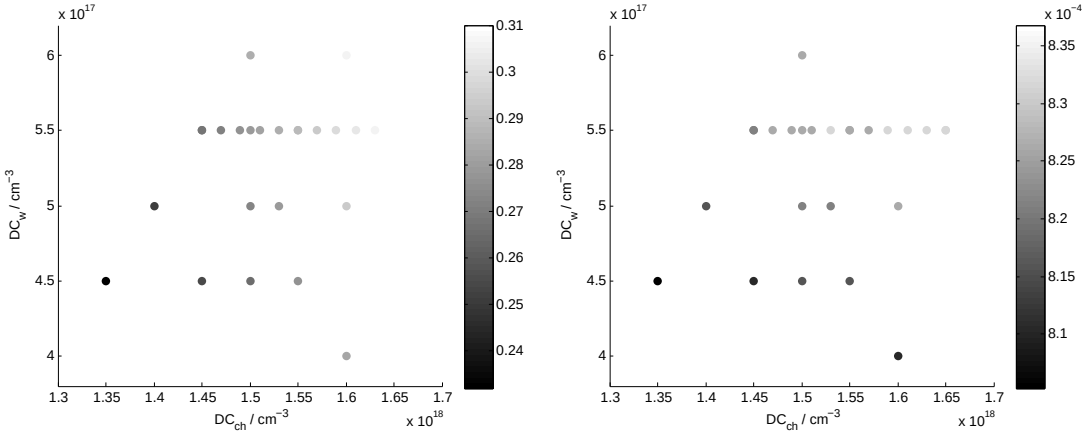


Figure 7.11.: (a) V_{TH} (gray scale) vs channel and well doping concentration. (b) $\frac{dV_{TH}}{dT}$ (gray scale) vs channel and well doping concentration.

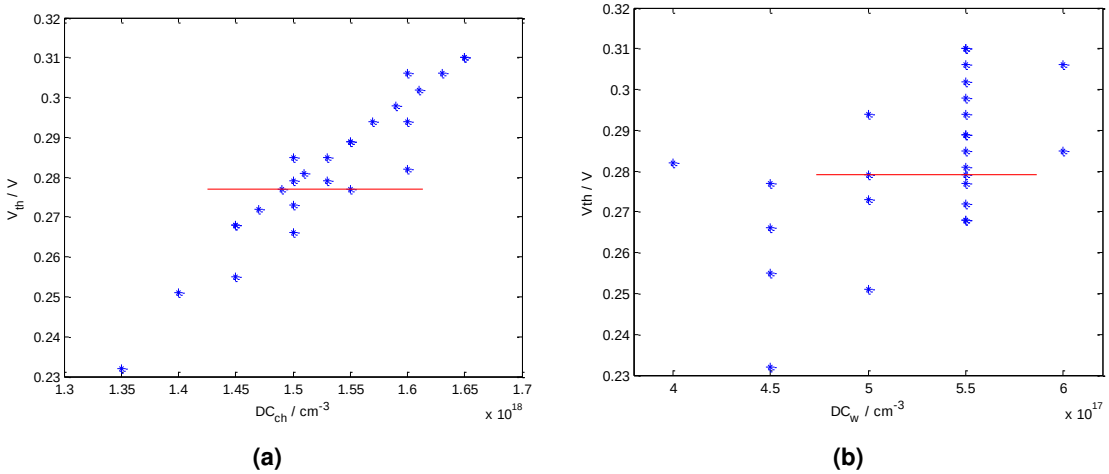


Figure 7.12.: (a) V_{TH} vs. channel doping concentration. (b) V_{TH} vs. substrate doping concentration.

behavior. A different perspective on the behavior of $\frac{dV_{TH}}{dT}$ is depicted in Figure 7.13. Figure 7.13a stresses the correlation between V_{TH} and $\frac{dV_{TH}}{dT}$. The red line shows transistors of identical V_{TH} but different $\frac{dV_{TH}}{dT}$. In Figure 7.13a $\frac{dV_{TH}}{dT}$ is shown as a function of V_{TH} and the ratio between the two doping concentrations. This plot is interesting, since it reflects precisely the case in which transistors with identical V_{TH} but different doping settings will have different temperature behavior. This is indicated by the lines in the graph.

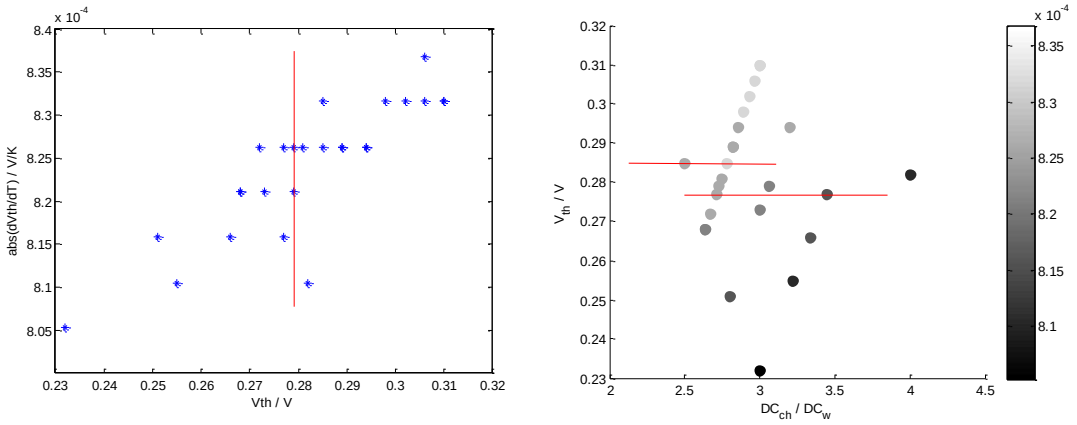


Figure 7.13.: (a) V_{TH} vs. $|\frac{dV_{TH}}{dT}|$. Red line: Same V_{TH}, different $\frac{dV_{TH}}{dT}$. (b) $|\frac{dV_{TH}}{dT}|$ (gray scale) vs. doping ratio and V_{TH}. The lines show transistors of the same V_{TH} but different temperature coefficients.

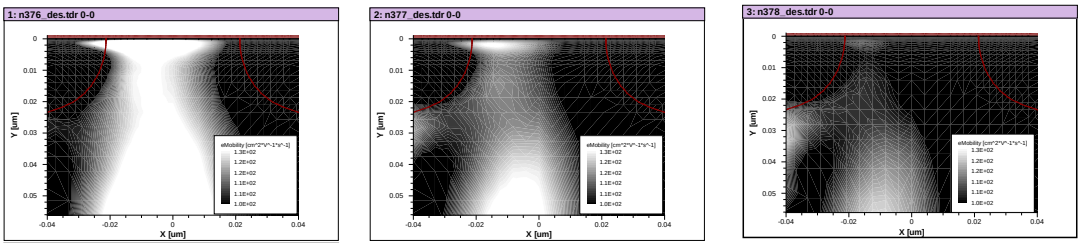
These results confirm the transistor measurement results from Figure 7.4a. This means that the mismatch effects can be explained by the variation of the doping concentration in the channel and the substrate region.

Not only a mismatch of the threshold voltage can cause deterministic temperature induced errors. A mismatch in mobility can also cause additional errors. Thus, also mobility mismatch will be analyzed.

7.4.3. Effect of Doping Concentration Variation on the Mobility

Mobility changes with temperature as well. And thus, not only the temperature coefficient of V_{TH} may vary between different transistors but also the temperature coefficient of the mobility (UTE) can be affected. Since the temperature behavior of mobility is dominant in the overall temperature behavior of a transistor in parts of the super threshold region, mismatches of UTE can have huge impact on the PUFs' temperature behavior. The effect of varying temperature on the mobility can be seen in Figure 7.14.

When the temperatures decreases the mobility increases, since the movement of the atoms decreases. If the temperature increases, the atoms start to move faster, the scattering increases, and the mobility reduces [145]. The temperature behavior of mobility is an important factor of the overall temperature behavior of a transistor. The effect of doping concentration variation on the temperature behavior is explored in the following sections.



(a) (b) (c)

Figure 7.14.: Electron mobility at (a) -73°C, (b) 27°C, (c) 117°C.

Changing the Doping Concentration in the Channel

Compared to the situation of the threshold voltage the overall mobility cannot be determined easily using TCAD. A hint how mobility behaves when the doping concentration is changed gives the two dimensional distribution of the electron mobility in the transistor. Figure 7.15 shows the electron mobility for three different channel doping concentrations. In Figure 7.15a the nominal case is shown. In Figure 7.15b the doping concentration in the channel is reduced by 6.67 % and so the mobility increases. The opposite happens in Figure 7.15c. In that case DC_{ch} is increased by 6.67 % which leads to a decreased mobility. The differences directly in the channel below the gate-oxide are so small so that it does not appear in the graphs of Figure 7.15d and Figure 7.15e. Figure 7.15d shows the electron mobility in y-direction from 0 nm (gate oxide) to 4 nm under the oxide. Figure 7.15e shows the electron mobility in x-direction $\pm$20 nm (whole channel region).

Regarding a fixed well doping concentration DC_w and varying channel doping concentration DC_{ch} the effect on the mobility could be explained as follows: if DC_{ch} is increased, the density of doping atoms increases and thus the probability of colliding charges increases. Therefore, the mobility decreases. If DC_{ch} is decreased, the density of doping atoms gets smaller and thus the probability of collision becomes smaller as well: the mobility decreases.

If the temperature is changed, the mobility changes as well. At -73°C the mobility increases clearly. Now the dependence on the doping concentration gets visible. The simulation results can be seen in Figure 7.16.

In Figure 7.16e the difference is clearly visible. The values do not match between the different doped transistors. That means that the temperature dependence is different and thus it does vary depending on the doping concentration.

Changing the Doping Concentration in the Substrate

If the doping concentration in the channel is held constant but the well doping changes, the threshold voltage changes and so does the mobility in the substrate region. The influence on the mobility in the channel region is even higher than in the case of changes in the channel doping. The TCAD results for three different doping configurations can be seen in Figure 7.17. The nominal doping is shown in Figure 7.17a. Figure 7.17b shows the mobility for the transistor of 10 % lower substrate doping concentration. In Figure 7.17c DC_{sub} was increased by 20 %. Only small differences can be recogniced between the different doping levels at room temperature.

If the temperature is decreased to -73°C (Figure 7.18), the difference between the transistors gets bigger (compare Figure 7.18e and 7.17e). This result is similar to the case of varied channel doping. Therefore, even changes in the substrate doping will have effect on the temperature behavior of the mobility in the channel region and thus have to be taken into account.

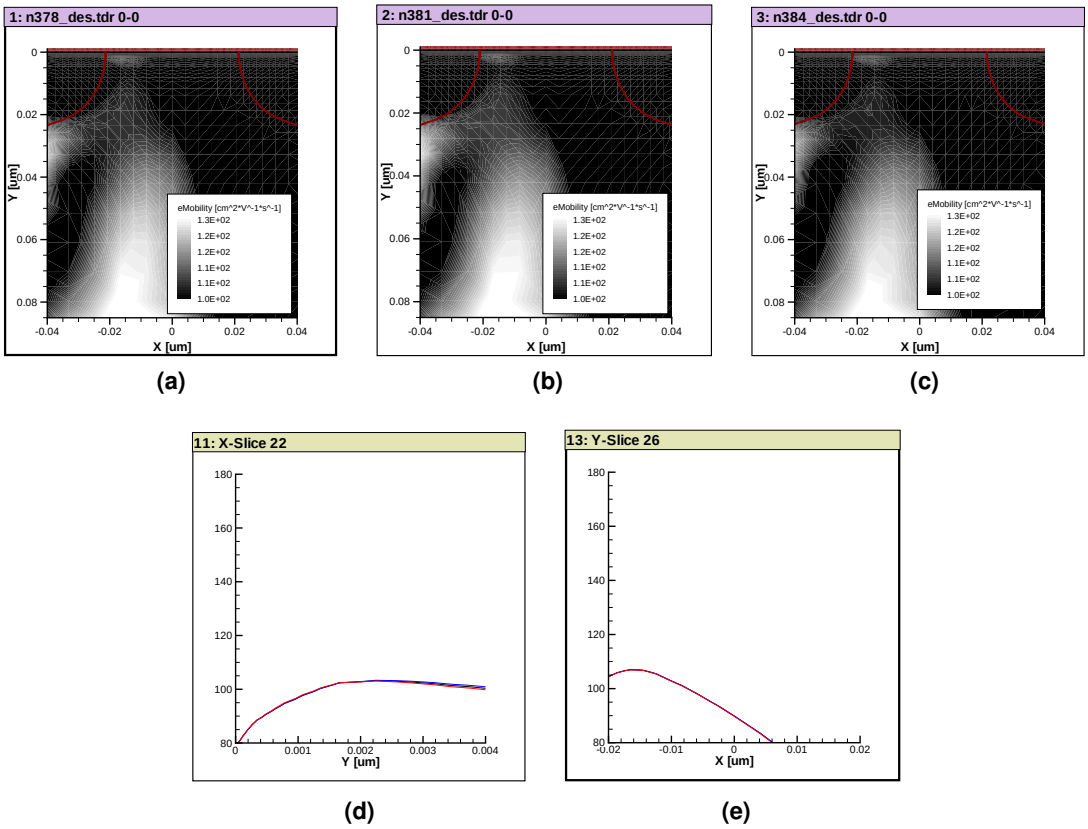


Figure 7.15.: Electron Mobility vs. DC_{ch} at 27°C: (a) nominal (green), (b) low (blue), (c) high (red). (d) Y-cut @ x=-0.001 µ, (e) X-cut @ y=0.002 µm.

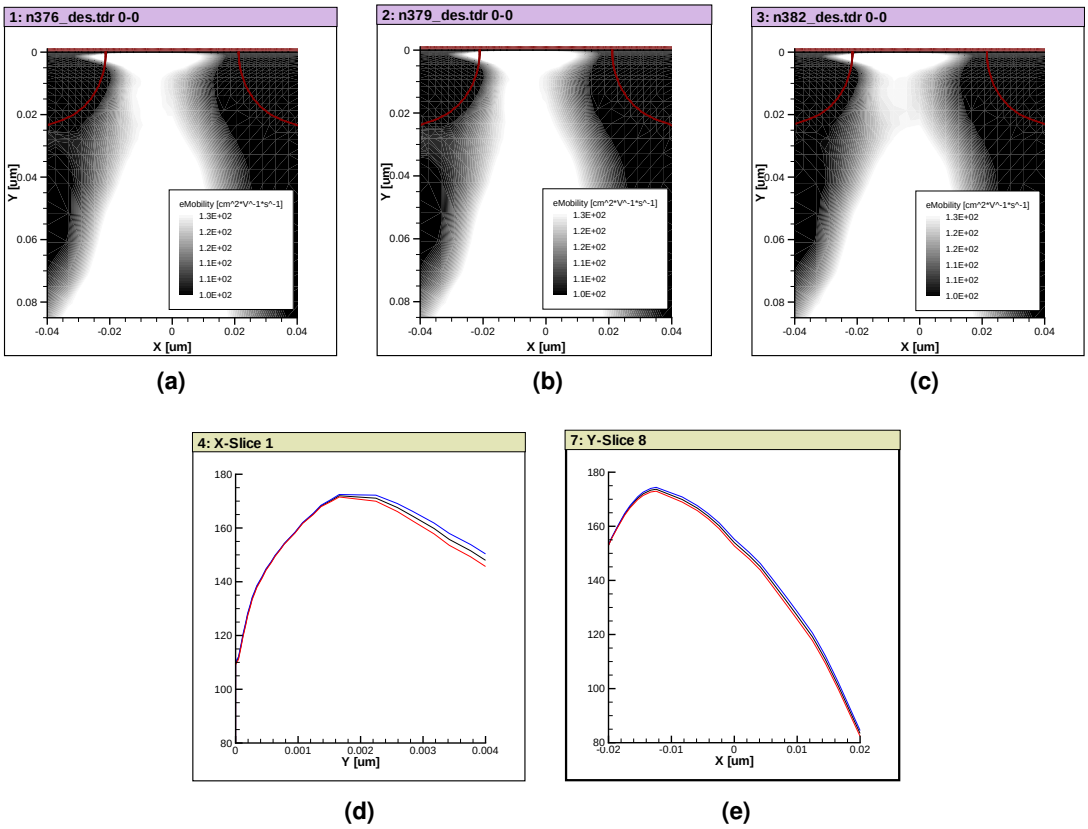


Figure 7.16.: Electron Mobility vs. DC_{ch} at -73°C: (a) nominal (green), (b) low (blue), (c) high (red). (d) Y-cut @ x=-0.001 µm, (e) X-cut @ y=0.002 µm.

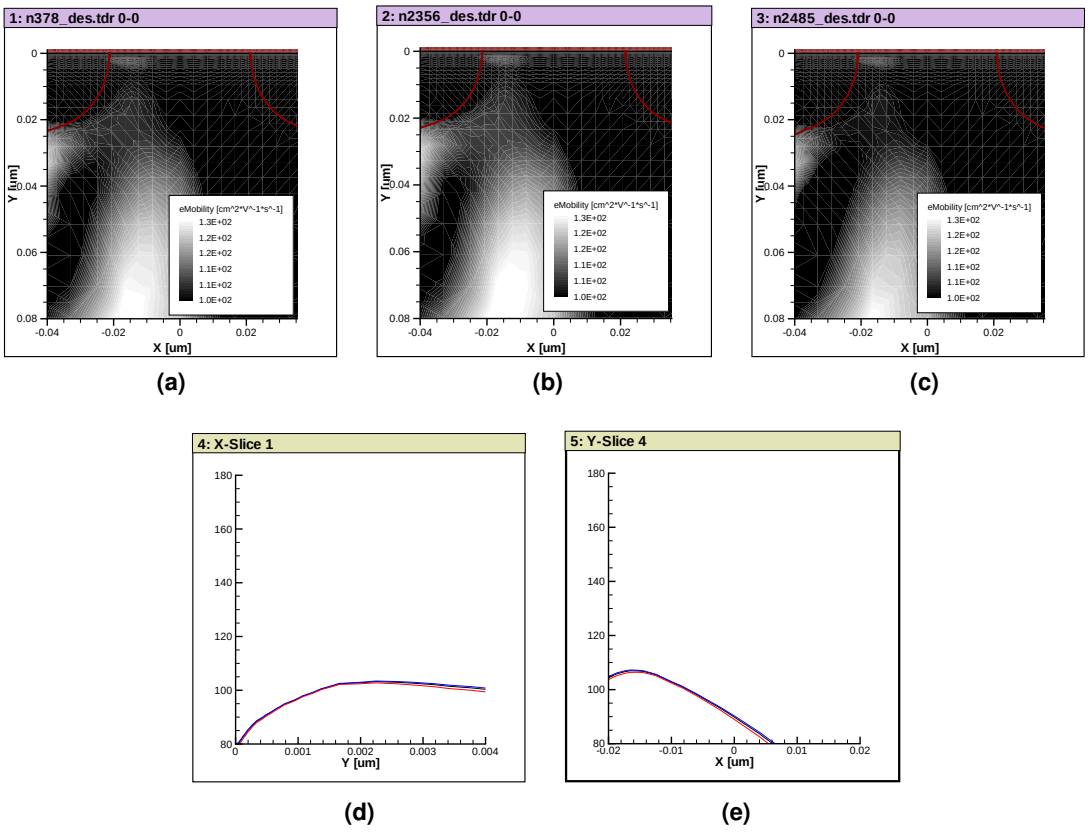


Figure 7.17.: Electron Mobility vs. DC_{sub} at 27°C: (a) nominal (green), (b) low (blue), (c) high (red). (d) Y-cut @ x=-0.001 µm, (e) X-cut @ y=0.002 µm.

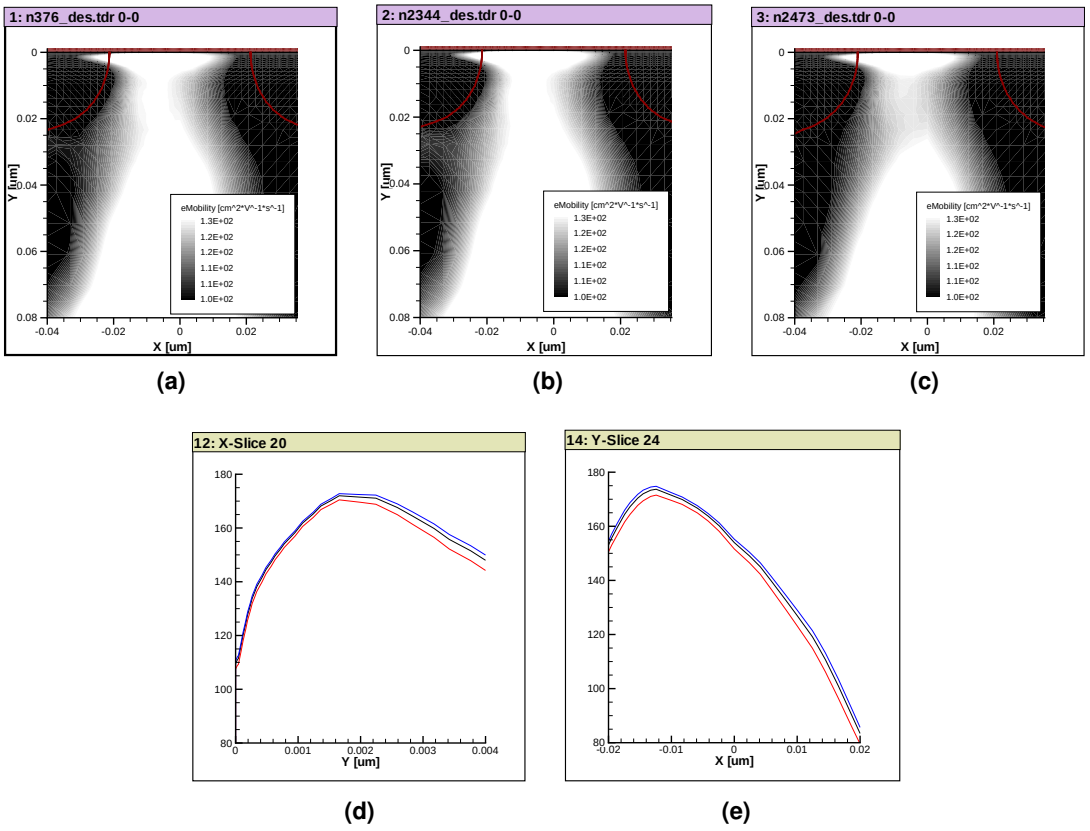


Figure 7.18.: Electron Mobility vs. DC_{sub} at -73°C: (a) nominal (green), (b) low (blue), (c) high (red). (d) Y-cut @ x=-0.001 µm, (e) X-cut @ y=0.002 µm.

Effect of Varying Doping Concentration on UTE

As shown above, not only the mobility varies when doping concentration changes but also the temperature behavior. Extracted TCAD data which were analyzed in Matlab are shown in Figure 7.19.

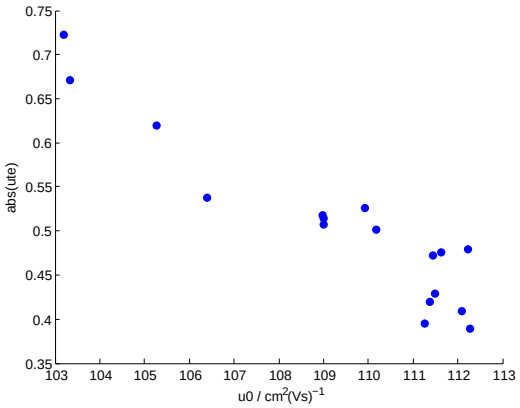


Figure 7.19.: |UTE| vs. electron mobility for different doping concentrations.

The mobility data was extracted 1 nm bellow the channel-oxide intersection. Though, no detailed results can be obtained from this data, it gives a hint to the real temperature behavior of mobility. A clear correlation between the mobility temperature coefficient UTE and the mobility μ can be observed. In the case of Figure 7.19 a 10 % increase in mobility results in 60 % decrease of the abolute value of UTE. Since this correlation is so evident and still is not included in the transistor model, it has to be taken into account when improving the model.

7.4.4. Conclusion of TCAD Analysis

The TCAD analysis of the minimum channel device showed that the temperature behavior of both, mobility and threshold voltage, vary for different doping profiles. While the threshold voltage as well as its temperature coefficient increase with increasing doping concentration the mobility decreases. Interestingly, the absolute value of its temperature coefficient increases when the doping concentration is increased. That could explain the strong increase of BER when the temperature is decreased. An overview of the TCAD results is given in Figure 7.20. It can be seen that all parameters are correlated to each other over the doping concentration in the channel. This is especially important when it comes to the Montecarlo mismatch models. This fact is not taken into account so far. There are no correlations between the two varied parameters VTH0 and U0. Even if these values cannot be assumed to correlate fully, correlation between these parameters and in succession a correlation between UTE and KT1 should be included in the models to achieve good simulation results.

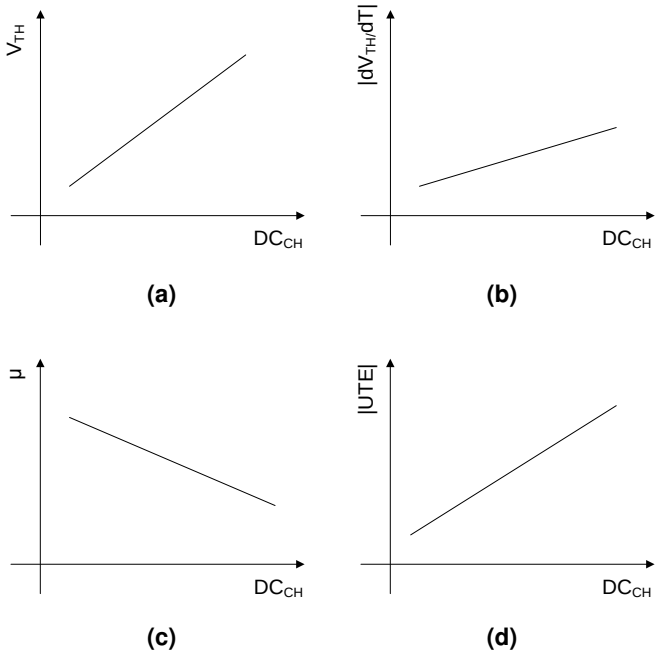


Figure 7.20.: The overview of the TCAD simulation results shows the behavior of different parameters versus a change of doping concentration.

7.5. Analytical Analysis of Temperature Coefficient Mismatch

7.5.1. Temperature Coefficient Mismatch of V_{TH}

To verify the results from TCAD the correlation between the mismatch of V_{TH} and the mismatch of its temperature coefficient KT1 is analyzed analytically in the following section. To do so, the basic V_{TH} model is used. Since V_{TH0} is the parameter that is varied in common transistor mismatch models and the bulk-source voltage V_{bs} is forced to zero in the presented PUF circuits, this is considered as a convenient approach. The difference of channel doping and substrate doping is not taken into account in this analysis.

From the BSIM 4 model the following relation between V_{TH0} and V_{TH} is given [31]:

$$V_{TH} = V_{T0} + \gamma \left(\sqrt{\Phi_s - V_{bs}} - \sqrt{\Phi_s} \right), \tag{7.2}$$

where γ is the so called body bias coefficient, Φ_s the surface potential and V_bs the bulk-source voltage. γ is defined as [31]:

$$\gamma = \frac{\sqrt{2q\epsilon_{si}N_{substrate}}}{C'_{ox}}, \tag{7.3}$$

where q is the charge of one hole ($q = 1.6 \times 10E-19$ C), ϵ_{si} the permittivity of silicon, $N_{substrate}$ the doping concentration of the substrate and C'_{ox} the oxide capacitance per unit area. V_{T0} is defined for the two terminal transistor in strong inversion which is also used for the simple model in BSIM4 [31, 145]:

$$V_{T0} = V_{FB} + \Phi_s + \gamma\sqrt{\Phi_s} \tag{7.4}$$

V_{FB} denotes the flatband voltage [145]:

$$V_{FB} = \phi_{MS} - \frac{Q'_0}{C'_{ox}}, \tag{7.5}$$

where ϕ_{MS} is the contact potential between the gate material and the substrate material. The second term covers the influence of charge, where Q'_0 denotes the effective charge per unit area of the parasitic charges within the oxide. The second term can often be neglected, since the amount of parasitic charges can be minimized by modern fabrication processes [145].

To be able to derive the threshold voltage, the value of Φ_s is needed. For the NMOS transistor case (PMOS can be derived equivalently), Φ_s at V_{TH} consists of two parts [145]:

$$\Phi_s = 2\phi_F + \Delta\phi, \tag{7.6}$$

where ϕ_F is the Fermi potential of the substrate and $\Delta\phi$ is a constant which is chosen in a way to get a good estimate of the threshold voltage. $\Delta\phi$ can vary from zero to several ϕ_F. [145] suggests five to six times ϕ_F. The Fermi potential is defined as follows:

$$\phi_F = \phi_t \ \ln(N_{substrate}/n_i), \tag{7.7}$$

where ϕ_t is the thermal voltage ($\phi_t = kT/q$, where k is the Boltzmann constant and T the temperature in Kelvin) and n_i the intrinsic carrier concentration (n_i is about 10^{10} per cm^3 at 300K (27°C)). Using the equation 7.7 ϕ_{MS} can be derived:

$$\phi_{MS} = \phi_{F,gate} - \phi_F, \tag{7.8}$$

where the Fermi voltage of the gate can be expressed for a polysilicon n-doped gate as [145]

$$\phi_{F,gate} = \frac{-E_g}{2q}. \tag{7.9}$$

E_g is the band-gap of silicon (1.12eV).

To be able to evaluate the temperature dependence of V_{TH}all the temperature dependent parts have to be taken into account. Additional to ϕ_t the intrinsic carrier concentration n_i [145] and the band gap of silicon E_g [162] also change when temperature alters.

$$n_i(T) = A_1 T^{3/2} e^{-A_2/T}, \tag{7.10}$$

where $A_1 = 7 \cdot 10^{15}/(\mathrm{K}^{3/2}\mathrm{cm}^3)$ and $A_2 = 6600\mathrm{K}$.

$$E_g(T) = E_g(0) - \frac{\alpha T^2}{T+\beta} \tag{7.11}$$

Here, $E_g(0)$, α and β are fitting parameters.

Taking all the different parts into account and considering an NMOS transistor with polysilicon n-doped gate the equation of V_{T0} gets

$$V_{T0} = \frac{-E_g(0) + \frac{\alpha T^2}{T+\beta}}{2q} + \frac{kT}{q} \ln\left(\frac{N}{A_1 T^{3/2} e^{-A_2/T}}\right) + \gamma \cdot \sqrt{2\frac{kT}{q} \ln\left(\frac{N}{A_1 T^{3/2} e^{-A_2/T}}\right)}. \tag{7.12}$$

Here it is shown that $N_{substrate}$ is substituted by N for convenience. To determine the dependence of V_{T0} on the temperature the derivative of V_{T0} towards the temperature has to be evaluated. After some calculations the following result can be derived:

$$dV_{T0} = \frac{\partial V_{T0}}{\partial T} = \frac{1}{2q}\frac{\alpha T(T+2\beta)}{(T+\beta)^2} + \frac{2k}{q}\left(\ln N - \ln\left(A_1 T^{2/3}\right) - \frac{3}{2}\right) + \frac{\sqrt{q}k\gamma}{q^2\sqrt{k}} \cdot \frac{\ln N - \ln\left(A_1 T^{2/3}\right) - \frac{3}{2}}{\sqrt{A_2 + T\ln N - T\ln\left(A_1 T^{2/3}\right)}} \tag{7.13}$$

As a second step, the derivative of $\frac{\partial V_{T0}}{\partial T}$ towards N is derived which is important to see the influence of a varying doping concentration on the temperature behavior of V_{T0}. The derivative ends up to be

$$\frac{\partial dV_{T0}}{\partial N} = \frac{k}{qN} + \frac{k\sqrt{2q^2\epsilon_{si}}}{C'_{ox}q^2\sqrt{k}} \cdot \frac{\partial}{\partial N}\left(\frac{\sqrt{N}\left(\ln N - \ln\left(A_1 T^{2/3}\right) - \frac{3}{2}\right)}{\sqrt{A_2 + T\ln N - T\ln\left(A_1 T^{2/3}\right)}}\right). \tag{7.14}$$

Since the derivative towards N is rather clumsy it is not presented in its exact formulation. Nevertheless, the first term of the equation is the dominant part of the equation and shows that the slope of dV_{T0} changes linearly when N is changed. Interestingly, the absolute value of temperature coefficient decreases when V_{TH} decreases which is contrary to the TCAD results. An explanation could be that TCAD determines the threshold voltages by extrapolating the I_D slope. Due to the strong mismatch behavior of mobility, the result could be influenced by that mismatch. Nevertheless, the analysis shows that there is a correlation between V_{TH} mismatch and the mismatch of its temperature coefficient.

7.5.2. Mismatch of Temperature Dependence of Mobility

Unlike the threshold voltage there is no convenient way to determine the mobility analytically [145]. For this reason, the temperature behavior of the mobility in dependence of the doping concentration is not solved here. So, the TCAD results are the only hint to a strong relation between the mobility mismatch and the mismatch of its temperature coefficient.

7.6. Model Parameter Adaptation

7.6.1. General Considerations

The above analyses revealed two important relations: Firstly, there are transistors that have the same $V_{TH}(\mu)$ but different $\frac{dV_{TH}}{dT}$ $(d\mu/dT)$ which can be caused by mismatches of the channel and the substrate doping. Secondly, there are correlations between the parameters and their temperature coefficients. In the next section, these results are included into the mismatch model of the transistors. The simulation results are compared to the real measurements afterwards.

To be able to estimate the error rate more realistically the BSIM4 [31] model parameters have to be adapted. Only the parameters U0 and V_{TH0} are varied during Monte Carlo local mismatch simulations using Gaussian distributions so far:

$$VTH0_{MC} = \mathcal{N}(VTH0_{NOM}, \sigma_{VTH0}) \tag{7.15}$$

$$U0_{MC} = \mathcal{N}(U0_{NOM}, \sigma_{U0}), \tag{7.16}$$

where $VTH0_{NOM}$ and $U0_{NOM}$ are the mean values and σ_{VTH0} and σ_{U0} are the variances of the Gaussian distributions. The values depend on the technology and the size of the particular

transistor. In the BSIM4 model the temperature dependence is included mainly by the following formulas:

$$\begin{aligned} V_{Th}(T) &= V_{TH}(VTH0_{MC}, TNOM) \\ &+ KT1 \cdot \left(\frac{T}{TNOM} - 1\right) \end{aligned} \tag{7.17}$$

$$U0(T) = U0_{MC}(TNOM) \cdot \left(\frac{T}{TNOM}\right)^{UTE}, \tag{7.18}$$

where in $V_{TH}(T)$ the influences of $KT1L$ and $KT2$ have been omitted.

As seen in the above section, the temperature coefficients UTE and KT1 of mobility μ and threshold voltage V_{TH} cannot be assumed to be constant for different transistors. Like V_{TH} and μ these coefficients depend on variations of the doping concentration. Since this kind of variation is not modeled in the state-of-the-art simulation environment, the mismatch simulation does not behave correctly. Especially concerning PUF circuits this leads to unexpected high error rates of the produced chips when compared to the simulation results. Looking at a temperature range from -40°C to 120°C the following discrepancy between simulation and measurement occurs: choosing the reference vector at 20°C the error rate does not exceed 2 % during Montcarlo simulation which can be handeld by common error correction codes easily. The error rate for the measured data increases up to 7 %. Since the EECs were not designed for this high error rates, the concept had to be adapted. The problem is depicted in Figure 7.21.

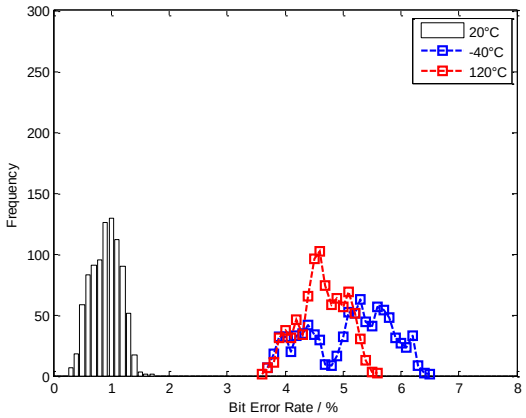


Figure 7.21.: The BER at three different temperatures for different chips is shown. The reference temperature is at 20°C. For -40°C and 120°C the error rate increases up to 7 % which is too much for simple error correction codes.

7.6.2. Temperature Coefficient Mismatch

The reason for that problem from a model perspective can be seen in Figure 7.22. It shows a sketch of V_{TH} measurements which were done for several transistors at different temperatures. The circles depict the situation in the Montecarlo mismatch simulation. There are two big differences between reality and the model: Firstly, there is a correlation between V_{TH} and its temperature coefficient. Secondly, transistors having the same V_{TH} may have different temperature coefficients. Both factors are not implemented in the model. Unfortunately, there are not any measurements for the behavior of the mobility available. Nevertheless, it can be assumed that an increase in mobility will also change the behavior during temperature variation. However, there should be at least a correlation between mobility and its temperature coefficient.

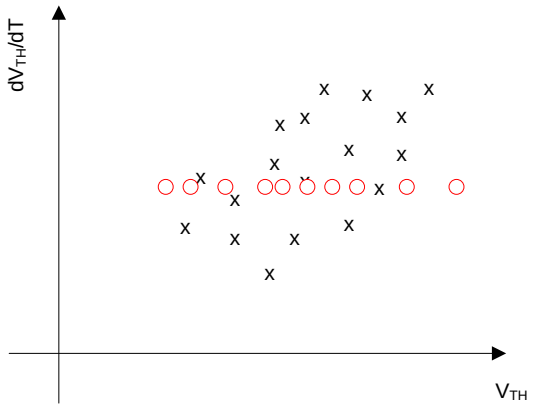


Figure 7.22.: Difference between model (○) and measurement (x) with respect to the threshold voltage and its temperature coefficient (KT). In the model, KT is fixed and not correlated with V_{TH}.

7.6.3. Measurement vs. Simulation

The simulation results of a Montecarlo simulation using the regular model can be seen in Figure 7.1. In this case -40°C is chosen as a reference vector and thus, the error rate at -40°C has its minimum. Since no noise is included in the simulation, the BER at -40°C is zero. As a reference, Figure 7.1 shows the measurement results. Beside the missing noise in the simulation there are two big differences: Firstly, the BER increases too slowly and secondly, the growth in the simulation is linear. The measurement results show that the growth rate gets smaller at higher temperatures.

7.6.4. Matlab PUF Model

To be able to solve the problem of Montecarlo simulation, a simple drain current (I_D) model of a MOS transistor in saturation region in strong inversion was implemented in Matlab. I_D without channel length modulation is defined as:

$$I_D = \beta \left(V_{GS} - VTH\right)^2, \tag{7.19}$$

where β is the amplification factor, V_{GS} the gate-source voltage and V_{TH} the threshold voltage. β can be also expressed as $\mu C_{OX} \frac{W}{L}$. Thus, the equation 7.19 results in

$$I_D = \mu C_{OX} \frac{W}{L} \left(V_{GS} - VTH\right)^2, \tag{7.20}$$

where μ denotes the mobility, C_{OX} the oxide capacitance per unit area, W the channel width, and L the channel length. This simple model is used in the Matlab simulation to build a PUF as shown in Figure 7.2 which is a simplified version of the test chip. Since the two important transistors are implemented in the Matlab model, the simplification can be assumed to approximate the real situation well. All the other components are ideal components. Since the currents through the two transistors are compared in the PUF cell, a cell can be modeled as follows:

$$OUT = I_{d1} > I_{d2}, \tag{7.21}$$

where I_{D1} and I_{D2} denote the drain current through transistor 1 and transistor 2 respectively.

The simple model already includes all parameters that are varied during the regular Montecarlo mismatch simulation. As described above, this is $VTH0$ and $U0$. These two parameters are varied

using a Gaussian distribution. They are varied uncorrelated to each other as it is normally done. $dVTH0$ which denotes the variation of $VTH0$, varies V_{TH} in Equation 7.20. $dU0$ varies the mobility μ. The standard deviations of the distributions are determined using transistor measurements and depend on $1/\sqrt{WL}$. The distribution mean is zero.

In the simple Matlab model, I_{D1} and I_{D2} are determined as follows (the following list is done for both transistors separately.):

1. Get random mumbers for V_{TH} and μ parts:

```
rand_vth = randn(1,1);
rand_u = randn(1,1);
```

2. Determine mismatch of V_{TH} and μ:

```
dvth = rand_vth * const_vth0/(sqrt(w*l));
du = u0*rand_u * const_u0/(sqrt(w*l));
```

   Here, $const_vth0$ and $const_u0$ are constants provided by transistor measurements, and w and l are the width and the length of the transistor's channel. $u0$ is the standard mobility.

3. Determine the mismatch of the temperature coefficients UTE and KT1: since the parameters are correlated, the same random numbers are used that also are used for dV_{TH} and du:

```
dkt1 = rand_vth * kt1/fact_kt1;
dute = rand_ute * ute/fact_ute;
```

   Here, `fact_kt1` and `fact_ute` define the standard deviation of `dkt1` and `dute`.

4. Then, the actual kt1 and ute are evaluated:

```
kt1_act = kt1 - dkt1;
ute_act = ute - dute;
```

   The minus sign is important to provide the correct behavior of the model.

5. Using the above results, the actual V_{TH} and μ can be determined depending on the temperature:

```
vth_act = (vth + dvth)) + kt1_act*(t/tnom - 1);
u0_act = (u0 + du).*(t/tnom)^ute_act;
```

   Here, t is the actual temperature and tnom the nominal temperature (room temperature).

6. Finally, the drain current can be evaluated:

```
id = (u0_act*K)*(vgs-vth_act)^2;
```

   Here, K is a constant that includes further model parameters.

Having done the above calculations for both transistors, the actual output can be determined:

```
OUT = id1_act > id2_act;
```

These equation include all important parts of the model.

7.6.5. Vary Temperature Coefficients in the Matlab Model

As suggested by the results above the temperature coefficients should also vary in the mismatch Montecarlo simulation. KT1 is the temperature coefficient that is mainly responsible for the temperature behavior of V_{TH} and UTE which accounts for the temperature variation of μ. Since the temperature coefficients also correlate with V_{TH} and μ respectively, KT1 and UTE are linked to their partners. The implementation of the variation is done as follows:

Varying KT1

The effect of the variation of the temperature coefficients can be seen in the following section: for the case that there is no variation of KT1 and UTE, the Matlab simulation should look quite like the Spectre simulation in Figure 7.1. The result can be seen in Figure 7.23a.

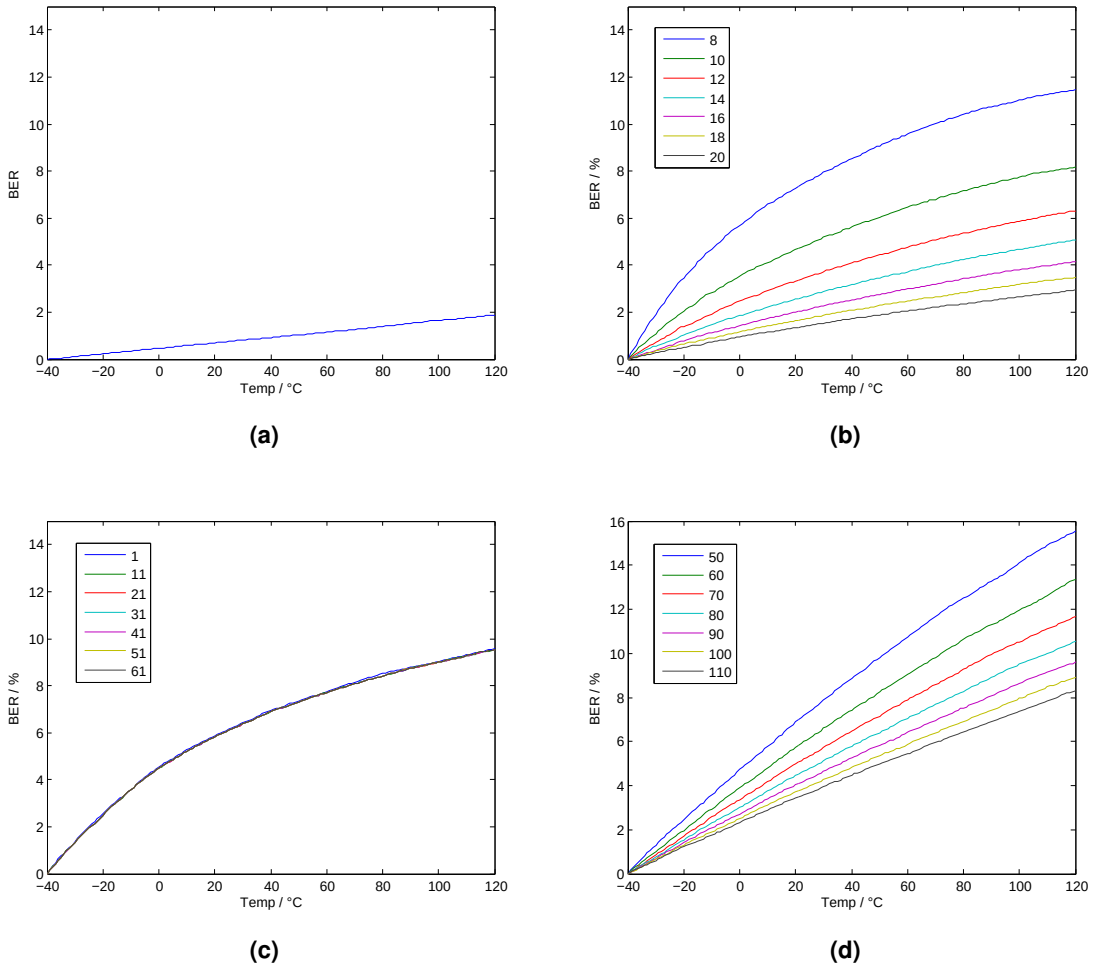


Figure 7.23.: (a) Regular mismatch model. (b) Correlated (w.r.t. VTH0 mismatch) KT1 mismatch (different fact_kt1). (c) Uncorrelated (w.r.t. VTH0 mismatch) KT1 mismatch (different fact_kt1) added to correlated KT1 mismatch. (d) Correlated (w.r.t. U0 mismatch) UTE mismatch (different fact_ute).

The BER is a little bit smaller than in the Spectre simulation. This discrepancy should not reduce the validity of the Matlab model. In Figure 7.23 only KT1 is varied. Several levels of KT1 mismatch are evaluated. In this case a level between 8 and 10 would end up in a result that matches the results of the measurements quite well.

In the case of Figure 7.23b the KT1 mismatch is chosen to correlate completely with the V_{TH} mismatch which does not fit to the results of the transistor measurements shown in Figure 7.22. In this model it is not possible that different temperature coefficients appear for the same V_{TH}. Thus, additionally to the part that correlates to V_{TH}, a second Gaussian distributed part is included which takes that effect into account. The results are shown in Figure 7.23c. The numbers show the inverse of the standard deviation of the uncorrelated part of the KT1 mismatch. The error rate

stays quite the same even for high portions of noise (number = 1). This means that for the model it is sufficient to consider the correlated part.

The growth of the error rate over temperature can be adapted in the model just by adding a KT1 mismatch to the Montecarlo parameters. This small variation of the original model increases the simulation accuracy remarkably.

Varying UTE

Adding only the UTE mismatch to the model and set the KT1 mismatch aside temporarily has the following effect on the simulation results. In this case only a mismatch is added that is correlated to the mismatch of the mobility U0. The result for different levels of mismatch can be seen in Figure 7.23d. Again, the numbers show the inverse of the Gaussian standard deviation, i.e. the higher the number gets, the smaller the influence of the mismatch will be. The correlation between $d\mu$ and dUTE has a positive sign with respect to their absolute values. This means, if the mobility increases, the temperature dependence increases as well which could be explained by the higher degree of freedom of the involved charges and thus to a higher influence of scattering effects due to temperature shifts. As shown in the case of the KT1 mismatch, varying UTE leads to an increase in error rate over temperature. But varying UTE produces a more linear increase in error rate than the pure KT1 mismatch does. Here again, varying UTE alone would lead to much more realistic results than the regular Montecarlo mismatch simulation does. But it is not able to reproduce the measured results exactly. Thus, the combination of both approaches may lead to the desired result.

Varying KT1 and UTE

In order to optimize the behavior of the model both parameters (i.e. KT1 and UTE) are varied. From measurements we know that the temperature coefficient of V_{TH} for minimum channel transistors has a standard deviation of about 3 %. Taking that fact into account, the model was optimized to fit the meaurement results of the testchip in Figure 7.24. The result of the optimization can be seen in Figure 7.24. In this case, the standard deviation of KT1 is chosen to be 33 % of the KT1 mean value. The variation of UTE is 1.7 %. The error growth starting at -40°C now adopts the characteristics of the measurement results.

7.6.6. From Matlab to SPECTRE

Putting the results from the Matlab analysis into the Spectre Montecarlo mismatch parameters, the mismatch simulation of the simple PUF cell returns the temperature behavior that is shown in Figure 7.24. In this example, the optimization was done by fully correlated variation of KT1 and UTE. KT1 is changed by a Gaussian distribution of zero mean and a standard deviation of 6.6 % of the regular KT1. The Gaussian distribution of UTE has also zero mean and a standard deviation of 2 % of the regular value. Both are correlated to the variation of their counterparts VTH0 and U0 with a sign of -1, e.g. if VTH0 is increased, KT1 is decreased. The overall result is a good approximation of the real temperature behavior of such a PUF circuit.

7.7. Summary

Regular models do not reproduce the temperature behavior of transistors in an acceptable quality to simulate PUF circuits in Montecarlo mismatch simulation. Since Montecarlo simulations are crucial in PUF design, it was important to improve the performance of the models. This could be done with the help of a Matlab transistor model basing on the common BSIM model. By varying

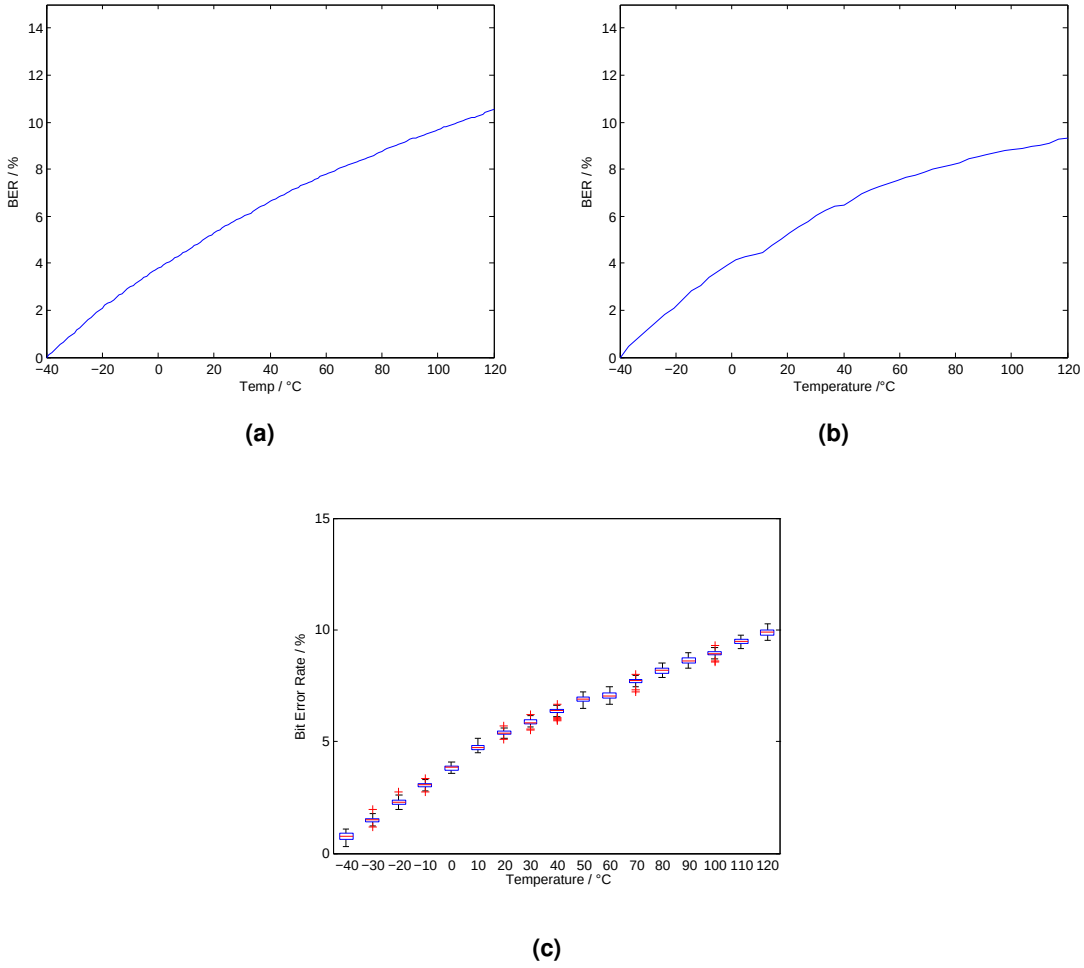


Figure 7.24.: (a) Matlab model (KT1 and UTE mismatch). (b) Adapted Spectre model (KT1 and UTE mismatch, 5000 Montecarlo runs). (c) Measurement Results.

the temperature coefficients KT1 and UTE of the BSIM model during Montecarlo mismatch simulation good results can be reached. This is especially important, if an estimation of the BER is needed for post-processing implementation.

8. Pre-Processing

by Christoph Boehm and Maximilian Hofer

As we have seen in Chapter 5, there are different problems related to error correction codes: PUF-cell overhead for redundancy, algorithm implementation complexity, power and time consumption during usage. Thus, it is a great advantage to reduce the error rate in the first place to allow for simple ECCs. In this chapter three pre-processing[1] approaches are introduced that are capable to reduce the BER of PUFs considerably. A classification of the pre-processing approaches is given in Section 8.2. Some of the disadvantages of the ECC can be reduced by using one of the different pre-processing approaches.

8.1. Fundamentals

Pre-processing steps are methods where phase the error rate is reduced by some technique during an initial. The goal of these pre-processing steps is to reduce the error rate in a way that error correction gets less complex or even unnecessary.

As in the case of error correction, additional data may be generated. In two of the three presented approaches, data that are generated during the initial phase have to be stored on an NVM. The data is later used to reduce the error rate.

Furthermore, redundancy of PUF cells may be needed, which means that more than one PUF cell is used to produce one bit of output. In the context of ECC these value was defined as redundancy R in Section 5. In the context of pre-selection it is called efficiency e:

$$e = \frac{\#\text{ Output values}}{\#\text{ all PUF cells}} \tag{8.1}$$

The higher the efficiency, the better the approach in terms of costs (area, power, etc.). As an example, if five PUF cells are needed to generate one bit of output the efficiency is 20 %. Therefore if 1024 output bits are required, 5120 PUF cells are needed.

A second interesting characteristic of pre-processing techniques is the relation between the efficiency and the stability. Typically, the higher the efficiency, the lower the stability. This relation is shown in Figure 8.1.

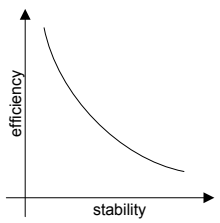


Figure 8.1.: Stability of pre-processing.

Therefore, a suitable ratio between stability and the number of required PUF cells have to be found depending on the application.

[1] pre-processing in contrast to post-processing approaches

One big advantage of pre-processing techniques compared to error correction codes (see Chapter 5) is that no additional complex calculations are necessary during the readout phase.

The idea behind all the approaches is to change the distribution of the mismatch in order to reduce the bit error rate. In the Figure 8.2a the density function of the mismatch between two components is depicted. In this case it is a Gaussian distribution of the mismatch between the transistors of the two sides of an SRAM PUF, for example. If the mismatch is not large enough, errors may occur (e.g.due to noise). It can be assumed that errors occur for cells that lie within the area between an upper and a lower threshold value MM_+ and MM_-. The gray bar marks this critical area. If the mismatch is high enough, the PUF cell is stable and errors are unlikely to occur. The density in this area should be forced to zero during the pre-processing (see Figure 8.2b). This can be done in different ways: one solution is to sort out those cells that lie within the area. Another approach is to change the properties of the cells by increasing the mismatch between the crucial components, to make sure that all the cells lie outside the critical range. This can either be done by influencing the properties of the components directly or by increasing the mismatch via summing up the mismatch of different cells by connecting a certain amount of them. In practice, a

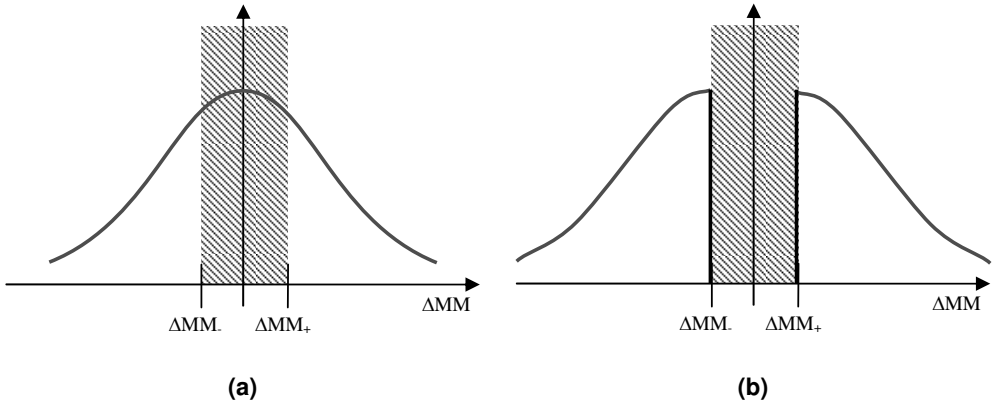


Figure 8.2.: (a) Normal distribution; (b) Desired distribution.

steep edge as it is depicted in the Figure 8.2 is not possible. Thus, the goal is to increase the slope of that edges as much as possible by means of one of the pre-processing techniques.

A formal description of the desired probability distribution can be written as follows:

$$p = 0 \; \forall \{\Delta MM_+ > \Delta MM > \Delta MM_-\} \tag{8.2}$$

$$p = f(\Delta MM) \; \forall \{\Delta MM_- \geq \Delta MM \geq \Delta MM_+\}, \tag{8.3}$$

where p is the probability of a cell having the mismatch ΔMM. Errors can occur due to unideal distributions: Within the range defined in 8.2 in reality p is not zero. Thus, there are always cells that produce errors yet after pre-processing. Furthermore, errors may occur even for cells which mismatch lies within the range defined in equation 8.3. This should make clear that the error rate can be forced to very small values but never reduced to zero.

8.2. Classification

A classification of the different pre-processing approaches is shown in Figure 8.3. By means of *PUF-parallelization* techniques a number of PUF cells having the same nominal output are connected in parallel to enforce a total increase in mismatch and thus provides a decrease of the error rate. In *pre-selection* techniques only high-mismatch cells are selected to produce the PUF

output in the future during an initial phase. *PUF biasing* techniques enforce a mismatch increase by changing the transistors' behavior directly.

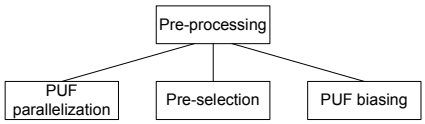


Figure 8.3.: Classification of the pre-processing approaches.

8.3. Comparison of Different Approaches

In the following section, the different pre-processing techniques are compared. Table 8.1 shows the advantages and disadvantages the three approaches. A detailed description of the different approaches can be found in the chapters 9, 10 and 11.

Table 8.1.: Comparison of different pre-processing approaches.

Approach	Hardware complexity	Pre-processing time	Requires NVM	Performance (BER)
PUF parallelization	High	Med	Yes	High
Pre-selection	Med	Med	Yes	High
PUF biasing	Med	High	No	Med

9. Parallelization of PUF Cells

by Christoph Boehm

Within the field of measurement engineering a common technique in order to reduce random errors is to determine the mean value of a number of measurement repetitions. Thus, the error can be reduced to the $\sqrt{N}$, where N is the number of repetitions. The PUF parallelization approach is based on basically the same idea, but here, not a single PUF cell is read out multiple times, but different PUF cells that provide the same nominal output are operating in parallel. As an example, a common SRAM is considered. In Figure 9.1a, the parallelization of two SRAM cells is depicted. The parallelization of two SRAM cells can be seen as a parallelization of the involved transistors.

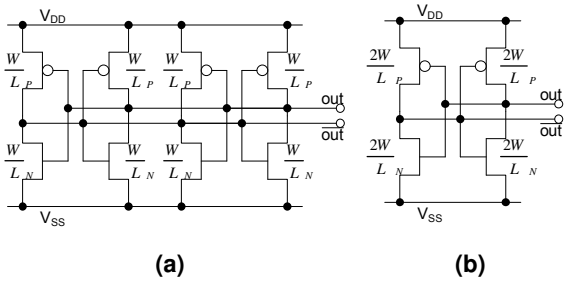


Figure 9.1.: Parallelization of PUF cells; (a) Parallelization of two SRAM cells; (b) Equivalent circuit of two parallelized SRAM cells.

Therefore, an equivalent circuit can be build from double width transistors. This circuit is depicted in Figure 9.1b. If the transistors are just put in parallel, the mismatch is reduced by the factor of $\sqrt{2}$ which is not desirable. Such a parallelization does not help to reduce the number of errors. As mentioned above, cells have to be parallelized that provide the same nominal output, i.e. cells which have the same sign of the mismatch. So, the total mismatch is increased and the error rate drops.

9.1. Introduction

As mentioned in the introduction, a PUF compares component properties to generate a binary output (e.g. threshold voltage differences of transistors). One PUF consists of a number of PUF cells. In a cell one component pair is compared providing one bit of output. The pairing of the components is done differently for the different PUF concepts. Some use the input to the PUF to define the pairing.

The difference (i.e. mismatch) between the compared components comes from local manufacturing variations. These variations can be assumed to be Gaussian distributed. Thus, the mean difference between the two components is zero.

If the mismatch gets near to zero, the error probability gets high, i.e. the output can easily be affected by disturbances. These include noise and variations of the operating point (V_{DD}, temperature, etc.). To reduce the error probability, the mismatch must be increased so that zero and near to zero mismatches become unlikely.

To increase the mismatches between components, a number of pairs can be combined electrically. By doing this, the probability of small mismatches can be reduced in any order depending on the number of combined pairs (see Figure 9.2). Table 9.1 shows the error rates for different degrees of parallelization.

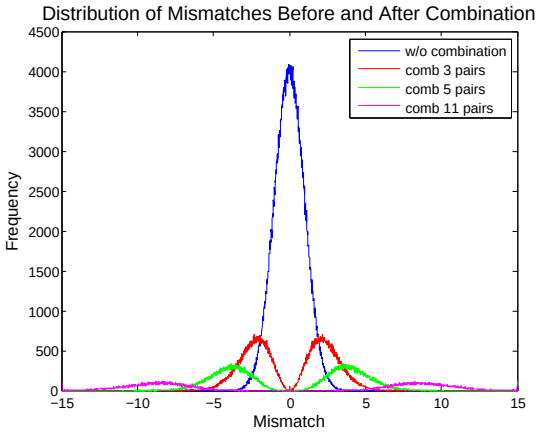


Figure 9.2.: Distribution of mismatches for different degrees of parallelization.

Table 9.1.: Error rate due to noise for different degrees of parallelization.

Pairs Per Cell	1	5	7	11
Error Rate (%)	4.75	33E-04	14E-06	0

During an initial phase, a defined number of pairs is grouped to produce one bit of output. The information is then stored in an NVM. The stored information (1bit per pair) does not reveal any information on the PUF's output and it can therefore be stored without any security risks.

9.2. Classification

In practice there are two approaches to parallelization: the first approach utilizes all cells in a block of pairs by defining - for example - the first pair as the reference pair and align the other pairs. In the second approach only these pairs are used which have initially the same sign of the mismatch as the reference pair. The other pairs are not used any longer. To use as much pairs as possible, the reference sign can be defined by a majority decision of the whole block instead of defining one pair as the reference. The classification is depicted in Figure 9.3. The two approaches are

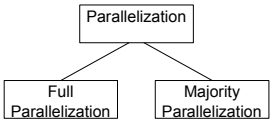


Figure 9.3.: Classification of parallelization.

explained by two example circuits in the following chapters in detail.

9.3. Full Parallelization

In the following example the compared components are two current sources. Due to the mismatch between the sources, the currents i_l and i_r differ. A sense amplifier measures the difference and takes a decision (i_l>=i_r) and outputs either '0' or '1'. The current through the sources depends mainly on the threshold voltage and the mobility of the involved transistors. These properties are influenced by the doping concentration in the different parts of the transistors. The doping concentration varies locally for it cannot be controlled exactly during production. Since these variations can be assumed to be Gaussian distributed the resulting mismatch between i_l and i_r is also Gaussian with zero mean.

To avoid current pairs with small mismatches, pairs are combined electrically. In the presented example, three pairs are combined to provide one bit of output. A schematic circuit is shown in Figure 9.4.

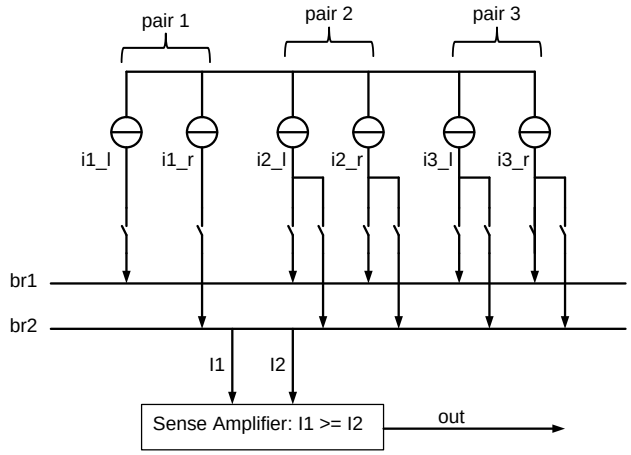


Figure 9.4.: Initial state of combination circuit.

The initialization phase consists of the following steps:

- The outputs of the three pairs are evaluated separately. During that time only the evaluated pair is connected to the two branches br1 and br2. An example is shown in Figure 9.5: Here, the second pair is evaluated.

- The output of the first pair is defined to be the reference output (could also be a majority decision or one of the other pairs). If the reference output is '1', the current source of each pair providing a larger current is connected to br1 during future read-outs. The other current source is connected to br2. If the output is '0', the current source of each pair providing a smaller current is connected to br1 and the other current source to br2 during future read-outs. This increases the total mismatch. The information of the connections is stored in an NVM (1bit/cell).

During the read-out phase the current sources are connected using the information stored in the NVM. If a '0' is stored, the current sources are connected as during the initial read-out. If a '1' is stored, the current sources are connected to the opposite branch. The example is depicted in Figure 9.6.

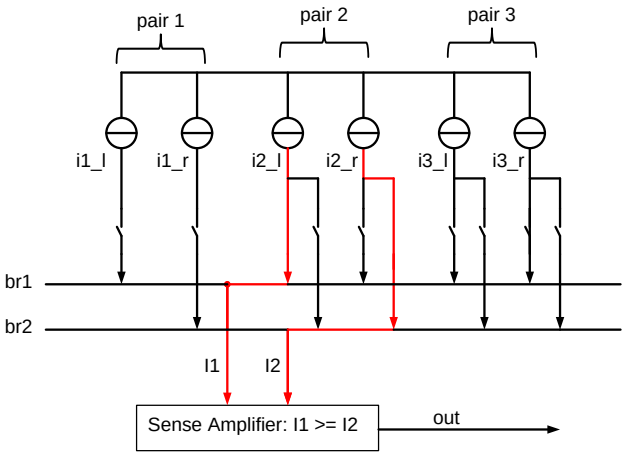


Figure 9.5.: Initial read-out of pair2.

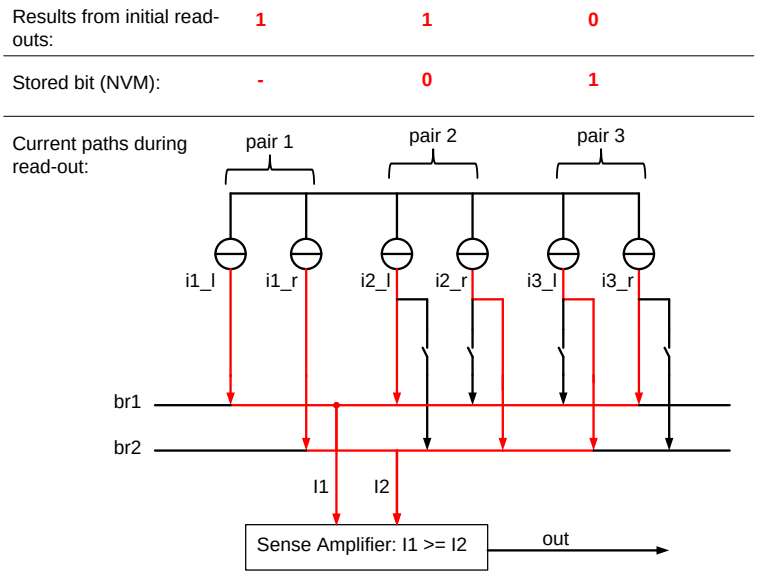


Figure 9.6.: Read-out using the data stored in the NVM.

Practical Realization

In Figure 9.7 a practical realization of the approach is depicted. Three SRAM cells are used in the example. During the initial phase the connection between the three cells is defined. The three cells are evaluated separately and the results are stored in the NVM. During the readout phase, the different SRAM cells are connected via the $cross$ and the $\overline{cross}$ switches in a way so that all have the same mismatch direction. Hence, the error rate is decreased.

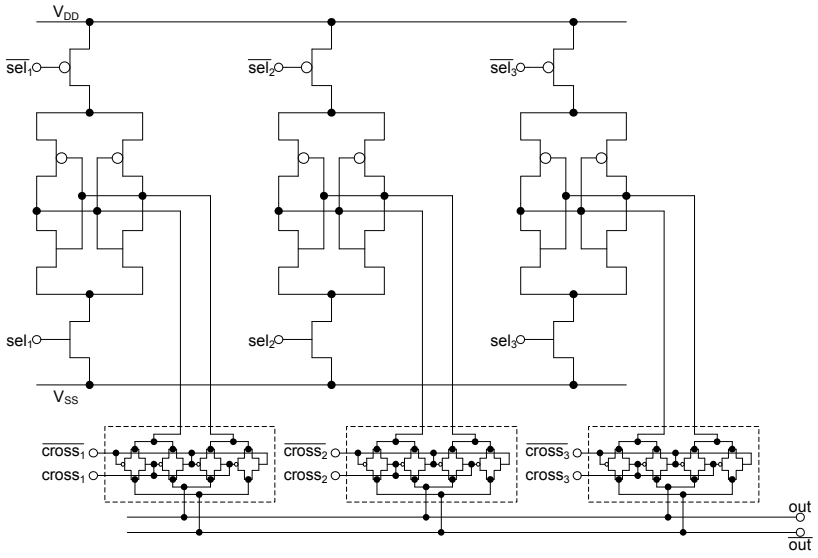


Figure 9.7.: Practical realization of full parallelization.

9.4. Majority Parallelization

As already mentioned above, instead of adjusting the sign of the pairs with respect to a reference pair, another possibility is to only utilize the matching pairs. A disadvantage of this approach is that a higher number of PUF cells is required to gain the same error rate than with the full parallelization approach. In return, the circuit complexity reduces due to the missing switches for mismatch alignment.

9.4.1. Practical Realization

Figure 9.8 shows the idea behind the approach. An example circuit was implemented in a 350 nm technology. The circuit is shown in Figure 9.9. In the example circuit five SRAM cells are connected via Q/Qn. In addition, every SRAM cell has two extra transistors to be able to activate/deactivate the cell. During an initial process each cell is powered-up once on its own. During this time, the four other cells are switched off. Thus, in this case one whole PUF cell needs at least five initial runs - one for each SRAM cell. To reduce the error probability due to noise, each cell can be powered on n-times which increases the number of runs by the factor of n. The results are stored. To increase the number of active SRAM cells, a majority decision of the outputs of the five SRAM cells is done. In an example (see Figure 9.9) the cells 1,3, and 4 return V_{DD} at outl. The cells 2 and 5 return V_{SS}. Thus, the majority decision returns V_{DD}. In the future read-outs, only the cells 1,3, and 4 are used. Due to the fact that their mismatches have the same sign, the mismatches sum up when the cells are used in parallel. Thus, the error rate decreases, since the effect of noise on the decision reduces.

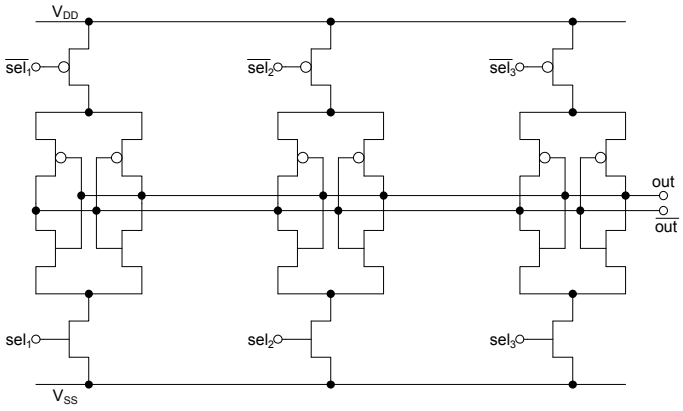


Figure 9.8.: Practical realization of majority parallelization.

Since the circuit is very simple, even if not all cells can be used, it could still be cheaper to implement than the fully parallelized version. Therefore, it depends on the principle of the used PUF which version should be preferred. In the worst case - which should be assumed to estimate the area consumption - the majority parallelizing needs double the number of cells than the full parallelization in average to provide the same error performance.

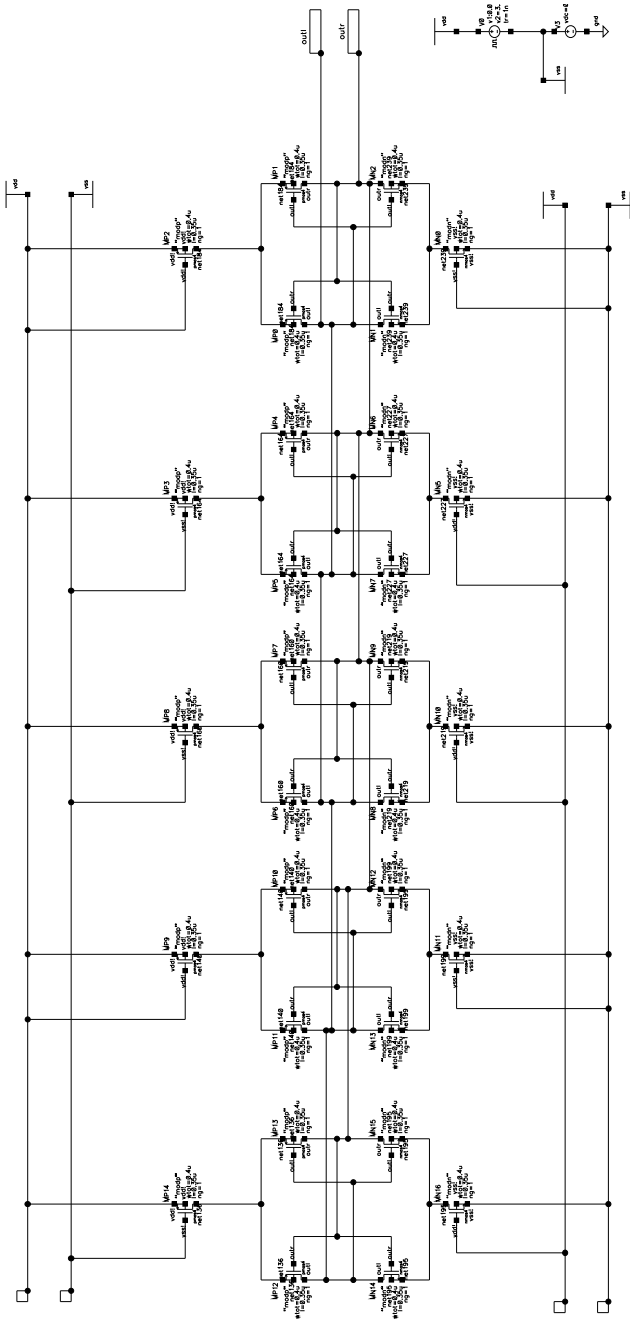


Figure 9.9.: Majority parallelizing PUF. The switches are already connected according to the parallelizing procedure.

10. Pre-Selection

by Maximilian Hofer

The idea behind the approach of pre-selection is similar to approaches known from biometric techniques: only the significant characteristics of the data are used for the identification or authentication process. During pre-selection only characteristic PUF cells are selected to provide data for the output of the PUF [62]. The characteristic cells are those cells that are unlikely to produce errors. PUF cells change their output due to the effect of noise, temperature, aging or other environment influences. Cells that change their output once are likely to change it again. PUF cells that change their output due to any reason are called unstable cells, PUF cells which always provide the same output are called stable, i.e. the HD_{intra} of a stable PUF cell is 0. The aim is to find the stable cells already during the initial phase. The concept is shown in Figure 10.1. In

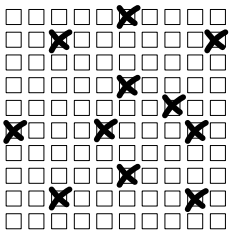


Figure 10.1.: Concept of pre-selection.

this example, 100 PUF cells are placed in an array. The unstable PUF cells are crossed out. The procedure of marking the stable PUF cells is called pre-selection. The remaining 89 PUF cells are selected for further use. In the example, the efficiency is e=89 %. After the pre-selection process the error rate should decrease. Thus, error correction becomes less complex or even unnecessary.

There are different approaches to find unstable PUF cells. A classification of pre-selection approaches is shown in Figure 10.2. Four different methods to find unstable PUF cells are being presented. In the first approach all cells are read out multiple times. Therefore, errors that occur due to noise can be detected. In the second approach called *pre-selection delta*, only PUF cells showing a pre-defined minimum mismatch are selected. The approach *pre-selection time* utilizes the amount of time that is needed for the PUF cell's decision. In the *pre-selection charge* approach the bit lines of common SRAM cells are pre-charged to find the unstable PUF cells.

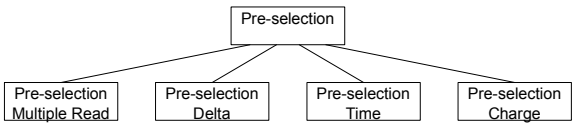


Figure 10.2.: Classification of the pre-selection.

10.1. Pre-Selection Multiple Readout

An intuitive approach is to measure the output of the PUF cells several times. Such cells which always provide the same output are marked as useful. Cells which do not always provide the same output are not used any longer. This can be done with common PUFs without any additional measurement hardware circuit needed. The software has to control the readout process during the initial phase. The information has to be stored in a NVM. At a first glance, this seems to be a good approach but in practice different problems occur. A lot of readout cycles have to find those cells which have a low error probability. If, for example, a cell provides the desired output at 99 % of the readouts and only at 1 % a wrong output there is a probability of 36 % that the cell is not detected as unstable during 100 readout cycles. On the other side, the probability to not detect a 50 % unstable PUF cell during 100 readout cycles is 7.98E-31 %. The errors caused by noise are reduced by the factor of $\sqrt{n}$, where n is the number of readouts. Another problem occurs due to a varying environment conditions. Such errors turn out to be worse than noise induced errors for most types of PUFs. For example, some cells change their output over temperature. The problem is depicted in Figure 10.3. The initial measurement is done at 25°C (Figure 10.3a). After

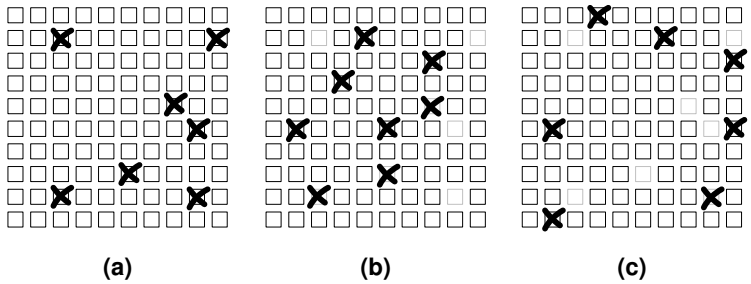


Figure 10.3.: (a) Unstable PUF Cells at 25°C; (b) Unstable PUF Cells at -40°C (c) Unstable PUF Cells at 120°C.

some readouts, seven PUF cells change their output value and so they are marked as unstable. If the temperature drops to -40°C, different PUF cells are unstable than at 25°C. Only three of the unstable PUF cells change their behavior at both temperatures (Figure 10.3b). The situation is similar at 120°C. Some of the PUF cells are unstable, but only one of these cells remains unstable at 25°C. Figure 10.4 shows an example of the mismatch behavior of PUF cells. At 25°C the overall stability is high. Due to mismatch in the temperature coefficients of the components it might be that some cells change their output value over temperature. Figure 10.5a shows the BER in dependence of the temperature at one point of pre-selection multiple readout. The maximal error rate can not be reduced significantly. To solve this problem, the initial measurements have to

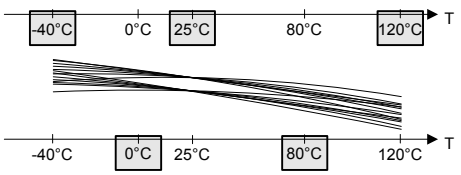


Figure 10.4.: Temperature dependence of mismatch components.

be done over the whole temperature range, for example at 25°C, -40°C and 120°C:

$$X_{25°C} = X_{-40°C} = X_{120°C} for\ all\ readouts \rightarrow \quad X = \quad stable \qquad otherwise \quad X = \quad unstable \tag{10.1}$$

where X is the output of a PUF cell. This equation can be seen as a AND connection and a OR connection. The ones in the AND connection or the zeros in the OR connection are the stable cells. So we get the following vector for the stable cells:

$$(\vec{X} \otimes \vec{X}) \oplus \overline{(\vec{X} \oplus \vec{X})} \tag{10.2}$$

If the initial readout is done as described above, the result is as shown in Figure 10.5b. By using

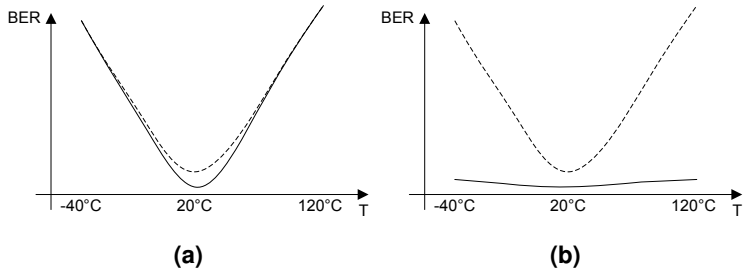


Figure 10.5.: BER in dependence of the temperature before (doted line) and after pre-selection multiple readout (solid line) (a) At one temperature (20°C); (b) At three temperatures (-40°C, 20°C and 120°C).

this pre-selection approach the BER can be reduced significantly. Unfortunately, the readout at different temperatures during the initial phase is costly and cannot be assumed to be a realistic procedure to reduce the error rate.

Another solution to this problem could be to increase the noise in the PUF cells artificially. With this approach less unstable cells can also be detected during multiple readout at room temperature. A noise generator is needed for it.

A practical realization of the approach is described in Section 13.3.

10.2. Pre-Selection Delta

10.2.1. Idea

The second approach which is presented in the following section is based on the selection of cells which provide a mismatch that exceeds a certain threshold. Thus, the approach is called Pre-Selection Delta.

For example, in the case of an SRAM PUF it is the mismatch between the involved transistors that has to be larger than a predefined threshold value. Figure 10.6 shows the threshold voltage mismatch (ΔV_{TH}) distribution of a pair of transistors schematically. As in the mismatch models the distribution is assumed to be Gaussian. The two dotted lines show the predefined positive (ΔV_{TH+}) and negative (ΔV_{TH-}) threshold values, where $|\Delta V_{TH+}| = |\Delta\ V_{TH-}|$. The threshold values split the distribution function into three areas: the central area includes all pairs whose mismatch is too small. They are marked as *not useful* (NUF). The areas to the left and to the right include all pairs whose mismatch exceeds the threshold value. Those pairs are marked either with UF_+ (positive mismatch) or UF_- (negative mismatch).

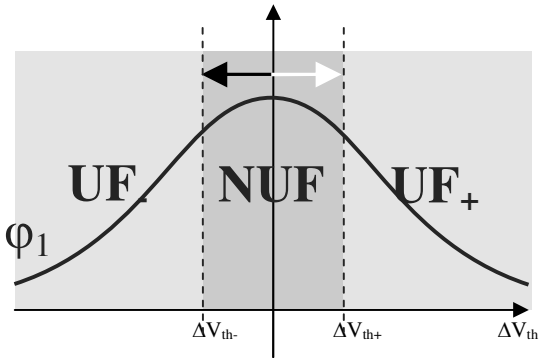


Figure 10.6.: The mismatch is divided into three classes: Useful PUF cells with positive mismatch (UF_+), useful PUF cells with negative mismatch(UF_-) and not useful PUF cells (NUF).

To provide a stable PUF output the threshold must be chosen correctly to gain the desired error rate. Choosing the threshold value is a trade of between error rate and the number of required PUF cells. If the threshold is chosen large, the error rate gets small but the number of not used pairs gets high. The other way around: if the threshold is chosen small, the number of selected pairs and the error rate get high.

One way to find the unstable cells is to measure the mismatch ΔV_{TH} by using common analog to digital converters (ADC). In Figure 10.7 a block diagram of that approach is shown. Here, the mismatching components are two diode-loaded transistors. Due to the mismatch, the voltages at the transistors differ. This voltage difference is measured at the ADC. The cell is selected, if the difference exceeds the threshold.

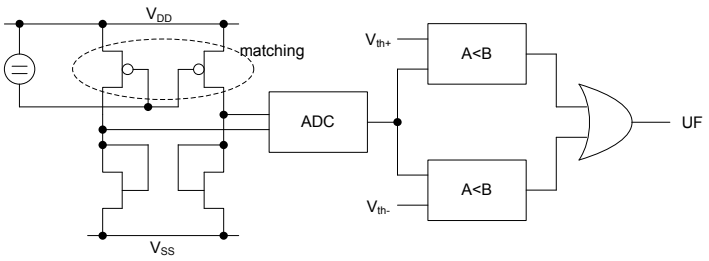


Figure 10.7.: Mismatch measurement using an ADC.

Unfortunately, there are some disadvantages related to the ADC approach. Since the ADC should add as little bias as possible on the measurement, the ADC has to be built large. Furthermore, the ADC should not depend on noise in the circuit to prevent errors. However, the resolution of the ADC may be very small. Three steps would be sufficient: UF_+, UF_-, and NUF. An exemplary circuit, which provides the desired requirements, is depicted in Figure 10.8a. The functionality of the ADC is illustrated in Figure 10.8b. The gray areas show the voltage values for which a '1' is produced at each of the outputs. If *UF* returns '1', the cell is selected.

A further approach is to add artificial mismatch during the measurements. At least two measurements are needed here to detect whether a cell is marked as useful or not. During the first measurement a negative offset is added. Thus, the threshold is set to V_{TH-}. During the second measurement, the threshold is moved to V_{TH+}. This is illustrated in Figure 10.6 by the two arrows. If the mismatch of the transistors exceeds the threshold, the PUF-cell will provide the same output for both measurements. If the mismatch is too small, the output will differ between the two measurements.

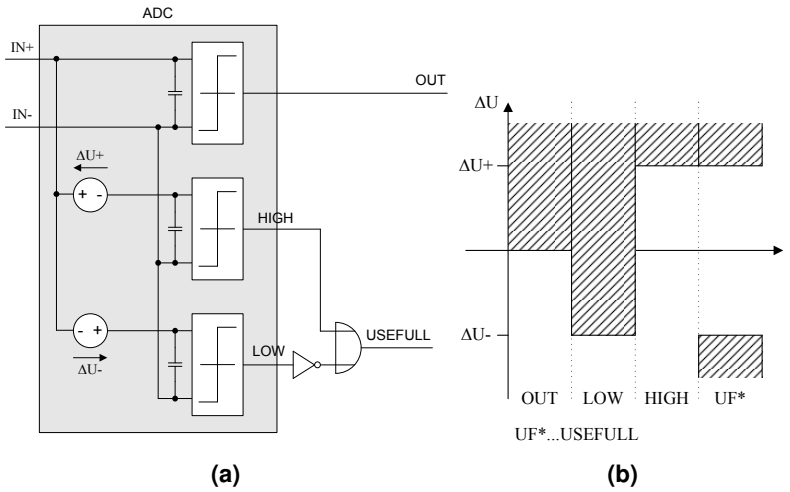


Figure 10.8.: (a) Three step ADC; (b) Diagram to illustrate the function of the ADC (gray: voltages where output gets '1').

An example of this approach is shown in Figure 10.11 in which a systematical V_{TH} offset is introduced to the circuit.

10.2.2. Modeling and Statistical Aspects

After the pre-selection process the error rate should be very small. Therefore, Montecarlo simulation is not an appropriate approach for BER estimation, since too many Montecarlo runs have to be accomplished to get accurate results. In the following section the error rate after pre-selection is estimated by analytical models.

To allow for reasonable complexity of the analysis in the remainder of the section, the distributions of the occurring random variables are assumed to be Gaussian. This includes the distribution of the mismatch and the distributions of the disturbances like noise, temperature dependence, etc [114, 115, 144].

To estimate the effect of pre-selection on the error rate, the probability density function (PDF) $f(x)$ and the cumulative distribution function (CDF) $F(x)$ of a Gaussian are needed. The PDF and the CDF are defined as follows:

$$f(x) = \phi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{10.3}$$

$$F(x) = \Phi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx, \tag{10.4}$$

where μ is the mean and σ the variance of the Gaussian distribution.

If no external factors interfere, the behavior of the PUF cell is just defined by the internal mismatches. Hence, the PUF cells return the same output again and again. For negative ΔV_{TH} the output is '0'. For positive ΔV_{TH}, the output is '1' (see Figure 10.9a). Φ_2 is the mean output value in dependence of ΔV_{TH}. In this case, Φ_2 shows the ideal decision distribution. The infinite steep slope at $\Delta V_{TH} = 0$ defines the crossing between the cells that provide zeros and the cells that provide ones.

Errors occur, if external factors like noise or temperature shifts interfere. In such a case, the behavior of the cell is not defined by the internal mismatch exclusively. The real decision distribution Φ_2 of a PUF cell can be seen in Figure 10.9b. The external factors may define the output for small values of ΔV_{TH}. Hence, the crossing between '1' and '0' output is not infinitely steep.

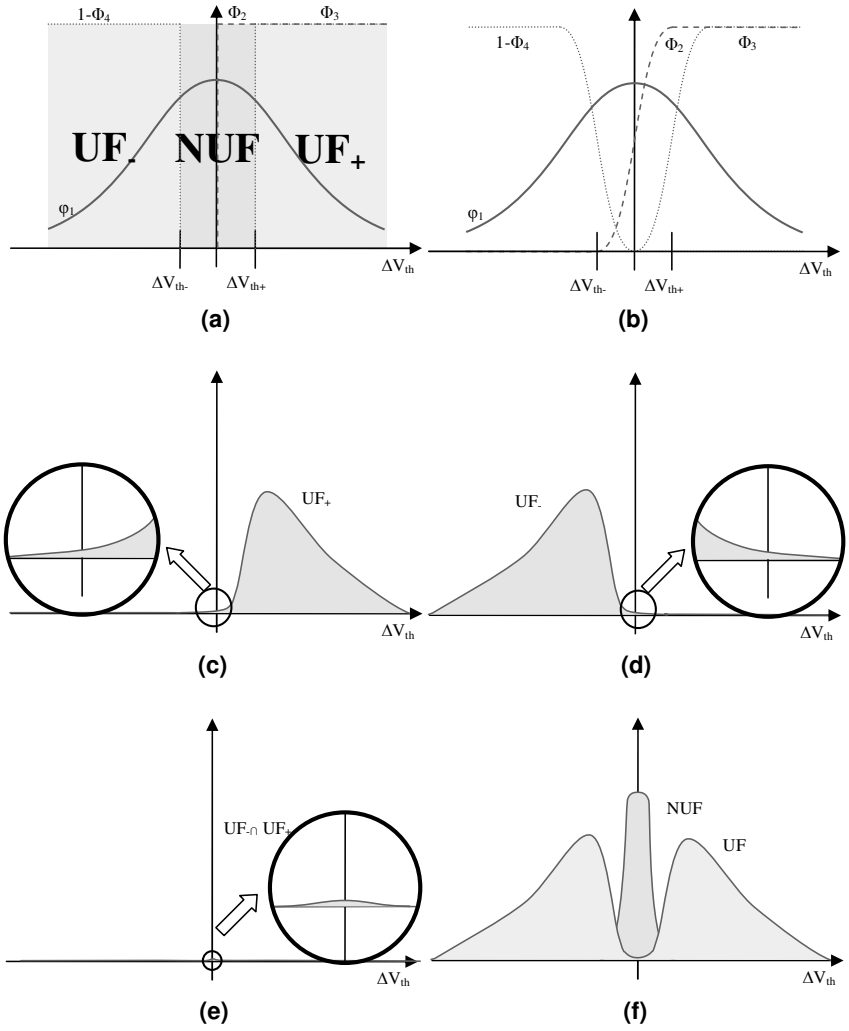


Figure 10.9.: (a) Ideal distributions $\Phi_1 - \Phi_4$.(b) Real distributions $\Phi_1 - \Phi_4$. (c) Useful PUF-cells with positive mismatch (UF+). (d) Useful PUF-cells with negative mismatch (UF-). (e) PUF-cells which occur in UF_+ and UF_-. (f) Useful PUF-cells (UF) and not-useful PUF-cells (NUF).

The CDF increases smoothly. If the PUF cell is biased either by a negative mismatch ΔV_{TH-} or by a positive mismatch ΔV_{TH+}, the CDF is shifted on the x-axis. Still, the distortion due to the external disturbances exists. After the biasing the CDF looks like shown by (Φ_3) and $(1-\Phi_4)$in Figure 10.9b. The effect of the biasing can be seen in Figures 10.9c and 10.9d. The curves show the distribution of the decision of pre-selected cells. It is the product of (ϕ_1) and one of the CDFs (e.g. (Φ_3)). Only very view cells produce a wrong output. Figure 10.9e shows those cells that are selected twice, i.e. that are declared to be a useful '1' *and* a useful '0'. Even if this case is very unlikely to happen, nevertheless it has to be taken into account. To get correct results these double-selections have to be compensated for in the analysis. Figure 10.9f shows the overall distributions of selected and not selected cells including all disturbances. The estimation of the ratio between selected and not selected cells is done below.

Ratio of Useful PUF Cells α: Probability of the occurrence of useful PUF-cells in dependence of ΔV_{TH}, V_{TH+}, and V_{TH-} (see Figure 10.9c and Figure 10.9d)

$$UF_+ = \phi_1\Phi_3 - \phi_1\Phi_3(1-\Phi_4) \tag{10.5}$$

$$UF_- = \phi_1(1-\Phi_4) - \phi_1\Phi_3(1-\Phi_4) \tag{10.6}$$

Probability of the occurrence of useful PUF-cells being selected twice in dependence of ΔV_{TH}, V_{TH+}, and V_{TH-} (see Figure 10.9e):

$$UF_+ \cap UF_- = \phi_1\Phi_3(1-\Phi_4) \tag{10.7}$$

Total probability of the occurrence of useful PUF-cells in dependence of ΔV_{TH}, V_{TH+}, and V_{TH-}:

$$\begin{aligned} UF &= UF_+ + UF_- - 2(UF_+ \cap UF_-) = \\ &= \phi_1[\Phi_3 + (1-\Phi_4) - 2\Phi_3(1-\Phi_4)] = \\ &= \phi_1[1-\Phi_4-\Phi_3+2\Phi_3\Phi_4] \end{aligned} \tag{10.8}$$

The ratio of useful PUF-cells α can be determined from UF:

$$\begin{aligned} \alpha &= \int_{\infty}^{-\infty} UF\, dV_{TH} = \\ &= \int_{\infty}^{-\infty} \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{V_{TH}-\mu_1}{\sigma_1}\right)^2} \left[1 - \frac{1}{\sigma_4\sqrt{2\pi}} \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_4}{\sigma_4}\right)^2} dV'_{TH} + \right. \\ &\quad -\frac{1}{\sigma_3\sqrt{2\pi}} \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_3}{\sigma_3}\right)^2} dV'_{TH} + \frac{2}{\sigma_3\sqrt{2\pi}} \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_3}{\sigma_3}\right)^2} dV'_{TH} \cdot \\ &\quad \left. \cdot\frac{1}{\sigma_4\sqrt{2\pi}} \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_4}{\sigma_4}\right)^2} dV'_{TH} \right] dV_{TH} = \\ &= 1 - \frac{1}{\sigma_1 2\pi} \int_{\infty}^{-\infty} e^{-\frac{1}{2}\left(\frac{V_{TH}-\mu_1}{\sigma_1}\right)^2} \left[\frac{1}{\sigma_4} \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_4}{\sigma_4}\right)^2} dV'_{TH} + \right. \\ &\quad -\frac{1}{\sigma_3} \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_3}{\sigma_3}\right)^2} dV'_{TH} + \frac{2}{\sigma_3\sigma_4\sqrt{2\pi}} \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_3}{\sigma_3}\right)^2} dV'_{TH} \cdot \\ &\quad \left. \cdot \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_4}{\sigma_4}\right)^2} dV'_{TH} \right] dV_{TH} \end{aligned} \tag{10.9}$$

In general, the following is assumed: $\sigma_2 = \sigma_3 = \sigma_4 = \sigma$, $\mu_1 = \mu_2 = 0$, and $\mu_3 = -\mu_4$. Consequently, α gets

$$\begin{aligned}\alpha &= 1 - \frac{1}{\sigma_1 2\pi}\int_{\infty}^{-\infty} e^{-\frac{1}{2}\left(\frac{V_{TH}}{\sigma_1}\right)^2}\left[\frac{1}{\sigma}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu}{\sigma}\right)^2} dV'_{TH} + \right.\\ &\quad -\frac{1}{\sigma}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu}{\sigma}\right)^2} dV'_{TH} + \frac{2}{\sigma^2\sqrt{2\pi}}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu}{\sigma}\right)^2} dV'_{TH}\cdot\\ &\quad \left.\cdot\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu}{\sigma}\right)^2} dV'_{TH}\right] dV_{TH}. \end{aligned} \tag{10.10}$$

Ratio of Not-Useful PUF Cells β: For verification purposes of α, the ratio β of not-useful PUF cells has to be evaluated as well:

$$\beta = \int_{\infty}^{-\infty} NUF\, dV_{TH} \tag{10.11}$$

$$\begin{aligned} NUF &= \phi_1[(1-\Phi_3)(1-(1-\Phi_4)+\Phi_3(1-\Phi_4))]\\ &= \phi_1[(1-\Phi_3)(\Phi_4)+\Phi_3(1-\Phi_4))]\\ &= \phi_1[\Phi_4-\Phi_4\Phi_3+\Phi_3-\Phi_4\Phi_3))]\\ &= \phi_1[\Phi_4+\Phi_3-2\Phi_4\Phi_3))] \end{aligned} \tag{10.12}$$

$$\begin{aligned}\beta &= \int_{\infty}^{-\infty}\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{V_{TH}-\mu_1}{\sigma_1}\right)^2}\left[\frac{1}{\sigma_3\sqrt{2\pi}}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_3}{\sigma_3}\right)^2} dV'_{TH}+\right.\\ &\quad +\frac{1}{\sigma_4\sqrt{2\pi}}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_4}{\sigma_4}\right)^2} dV'_{TH} - \frac{2}{\sigma_3\sqrt{2\pi}}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_3}{\sigma_3}\right)^2} dV'_{TH}\cdot\\ &\quad \left.\cdot\frac{1}{\sigma_4\sqrt{2\pi}}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_4}{\sigma_4}\right)^2} dV'_{TH}\right] dV_{TH} =\\ &= \frac{1}{\sigma_1 2\pi}\int_{\infty}^{-\infty} e^{-\frac{1}{2}\left(\frac{V_{TH}-\mu_1}{\sigma_1}\right)^2}\left[\frac{1}{\sigma_3}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_3}{\sigma_3}\right)^2} dV'_{TH}+\right.\\ &\quad +\frac{1}{\sigma_4}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_4}{\sigma_4}\right)^2} dV'_{TH} - \frac{2}{\sigma_3\sigma_4\sqrt{2\pi}}\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_3}{\sigma_3}\right)^2} dV'_{TH}\cdot\\ &\quad \left.\cdot\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_4}{\sigma_4}\right)^2} dV'_{TH}\right] dV_{TH} \end{aligned} \tag{10.13}$$

As in the above case the following assumptions are made: $\sigma_2 = \sigma_3 = \sigma_4 = \sigma$, $\mu_1 = \mu 2 = 0$, and $\mu_3 = -\mu_4$. Therefore, β can be deterimed to

$$\begin{aligned}\beta &= \frac{1}{\sigma_1 \sigma 2\pi} \int_{\infty}^{-\infty} e^{-\frac{1}{2}\left(\frac{V_{TH}}{\sigma_1}\right)^2} \left[\int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu}{\sigma}\right)^2} dV'_{TH} + \right. \\ &+ \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu_4}{\sigma_4}\right)^2} dV'_{TH} - \frac{2}{\sigma} \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu}{\sigma}\right)^2} dV'_{TH} \cdot \\ &\left. \cdot \int_{-\infty}^{V_{TH}} e^{-\frac{1}{2}\left(\frac{V'_{TH}-\mu}{\sigma}\right)^2} dV'_{TH} \right] dV_{TH}. \end{aligned} \tag{10.14}$$

Check: $\alpha + \beta = 1$ ✓

Estimation of the Error Rate e: As soon as a selected cell returns the wrong output an error occurs. In this case, the total error $e(\Delta \mathrm{V_{TH}})$ at a certain $\Delta \mathrm{V_{TH}}$ is determined by summing up the two partial errors $e_-(\Delta \mathrm{V_{TH}})$ and $e_+(\Delta \mathrm{V_{TH}})$. The partial errors can be derived as follows:

$$e_-(\Delta V_{TH}) = \Phi_2 U F_- = \Phi_2[\phi_1(1-\Phi_4) - \phi_1\Phi_3(1-\Phi_4)] \tag{10.15}$$

$$e_+(\Delta V_{TH}) = (1-\Phi_2) U F_+ = (1-\Phi_2)[\phi_1\Phi_3 - \phi_1\Phi_3(1-\Phi_4)] \tag{10.16}$$

$$\begin{aligned} e(\Delta V_{TH}) &= e_+(\Delta V_{TH}) + e_-(\Delta V_{TH}) = \\ &= (1-\Phi_2)[\phi_1\Phi_3 - \phi_1\Phi_3(1-\Phi_4)] + \\ &\quad +\Phi_2[\phi_1(1-\Phi_4) - \phi_1\Phi_3(1-\Phi_4)] = \\ &= [\phi_1\Phi_2 - \phi_1\Phi_2\Phi_4 - \phi_1\Phi_2\Phi_3 + \phi_1\Phi_2\Phi_3\Phi_4] + \\ &\quad +[\phi_1\Phi_3 - \phi_1\Phi_3(1-\Phi_4)] - \Phi_2\phi_1\Phi_3 + \Phi_2\phi_1\Phi_3(1-\Phi_4) = \\ &= \phi_1\Phi_2 - \phi_1\Phi_2\Phi_4 - \phi_1\Phi_2\Phi_3\Phi_4 + \phi_1\Phi_3 - \phi_1\Phi_3 + \phi_1\Phi_3\Phi_4 - \\ &\quad -\Phi_2\phi_1\Phi_3 + \Phi_2\phi_1\Phi_3 - \Phi_2\phi_1\Phi_3\Phi_4 \end{aligned} \tag{10.17}$$

Finally, the total error e is derived from the integration of the $e(\Delta \mathrm{V_{TH}})$ dependent errors $e(\Delta \mathrm{V_{TH}})$:

$$e = \int_{\infty}^{-\infty} e(\Delta V_{TH})\, d\Delta V_{TH} \tag{10.18}$$

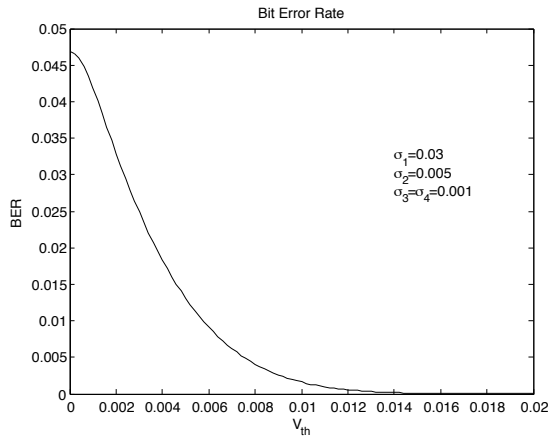
$$e(\Delta V_{TH}) = \phi_1\Phi_2 - \phi_1\Phi_2\Phi_4 - \phi_1\Phi_2\Phi_3\Phi_4 + \phi_1\Phi_3 - \phi_1\Phi_3 + \phi_1\Phi_3\Phi_4 - \Phi_2\phi_1\Phi_3\Phi_4, \tag{10.19}$$

where all ϕ_i and Φ_i depend on $\Delta \mathrm{V_{TH}}$.

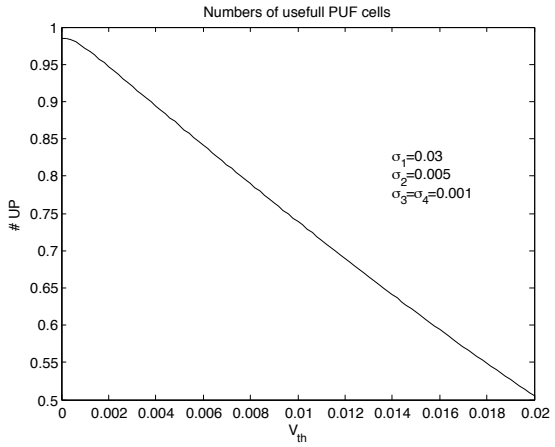
Example: To be able to estimate the error rate after pre-selection, the standard deviation of $\Delta \mathrm{V_{TH}}$ and $\phi_{2,3,4}$ have to be known. Here, the standard deviation of $\Delta \mathrm{V_{TH}}$ is set to 30 mV, the standard deviation of $\phi_{2,3,4}$ is set to 1 mV. Figure 10.10 shows the error rate e (10.10a) and the ratio of useful PUF-cells α (10.10b), both depending on the pre-selection threshold.

By selecting as much as 95 % of the cells the error rate already drops down to around 3 %. The results of different examples are shown in Table 10.1.

The results show the number of useful PUF-cells depends mainly on the ratio between the mismatch and the pre-selection threshold $\frac{\sigma_1}{\mu_{3,4}}$. The crucial parameters are the noise parameters $\sigma_{2,3,4}$ and the pre-selection parameters $\mu_{3,4}$ for the error rate e. σ_1 only shows a small influence on e. In Table 10.2 e is shown in dependence of $\mu_{3,4}$.



(a)



(b)

Figure 10.10.: Influence of the pre-selection threshold value ($\mu_{3,4}$): a) On error rate e; (b) On ratio of useful PUF-cells α.

Table 10.1.: Examples of the error rate e and the ratio of useful PUF-cells α in dependence of model parameters.

Num.	σ_1	σ_2	$\sigma_3 = \sigma_4$	$\mu_3 = \mu_4$	α	e
1	30 mV	1 mV	1 mV	5 mV	86.77	2.19E-06
2	30 mV	1 mV	1 mV	10 mV	73.90	<1E-12
3	30 mV	1 mV	1 mV	20 mV	50.82	<1E-12
4	30 mV	0.5 mV	0.5 mV	10 mV	73.89	<1E-12
5	30 mV	2 mV	2 mV	10 mV	73.94	5.09E-06
6	10 mV	1 mV	1 mV	10 mV	31.97	<1E-12
7	50 mV	1 mV	1 mV	10 mV	84.15	<1E-12

Table 10.2.: Numeric examples of the error rate e and the ratio of useful PUF-cells α in dependence of $\mu_{3,4}(\sigma_1 = 30\,\text{mV}, \sigma_{2,3,4} = 1\,\text{mV})$.

$\mu_{3,1}$	e	α	$\mu_{3,1}$	e	α
0	7,61E-03	9,85E-01	5,5	5,09E-07	8,55E-01
0,25	7,39E-03	9,84E-01	5,75	2,35E-07	8,48E-01
0,5	6,76E-03	9,81E-01	6	1,06E-07	8,42E-01
0,75	5,83E-03	9,77E-01	6,25	4,60E-07	8,35E-01
1	4,78E-03	9,72E-01	6,75	8,02E-09	8,22E-01
1,25	3,73E-03	9,66E-01	7	3,20E-09	8,16E-01
1,5	2,79E-03	9,60E-01	7,25	1,24E-09	8,09E-01
1,75	2,01E-03	9,53E-01	7,5	4,68E-10	8,03E-01
2	1,40E-03	9,53E-01	7,75	1,71E-10	7,96E-01
2,25	9,46E-04	9,40E-01	8	6,09E-11	7,90E-01
2,5	6,22E-04	9,34E-01	8,5	7,02E-12	7,77E-01
2,75	3,99E-04	9,27E-01	8,75	2,28E-12	7,71E-01
3	2,49E-04	9,20E-01	9	7,17E-13	7,64E-01
3,25	1,51E-04	9,14E-01	9,25	2,18E-13	7,58E-01
3,5	8,97E-05	9,07E-01	9,5	6,31E-14	7,52E-01
3,75	5,17E-05	9,01E-01	9,75	1,17E-14	7,46E-01
4	2,90E-05	8,94E-01	10	2,94E-15	7,39E-01
4,25	1,59E-05	8,87E-01			
4,5	8,44E-06	8,87E-01			
4,75	4,36E-06	8,74E-01			
5	2,19E-06	8,68E-01			

10.2.3. Implementation

The described pre-selection approach can be implemented in different ways. One of them is presented here. The circuit is implemented using a ordinary SRAM cell Figure 10.11. As an assumption, the transistors P_1 and P_2 are designed in a way that they are matching (large area). Hence, the mismatch between the two NMOS transistors N_1 and N_2 define the power-up behavior. The mismatch is denoted with ΔV_{TH}. To implement the pre-selection as described above 10.2.1 an additional voltage source has to be introduced at the of one NMOS transistor. This voltage source later provides the pre-selection bias (see Figure 10.11a). Instead of implementing a voltage source at the gate, at its Norton equivalent - a current source - can be introduced in parallel to one of the NMOS transistors (see Figure 10.11b). Figure 10.12 shows the equivalence between the voltage

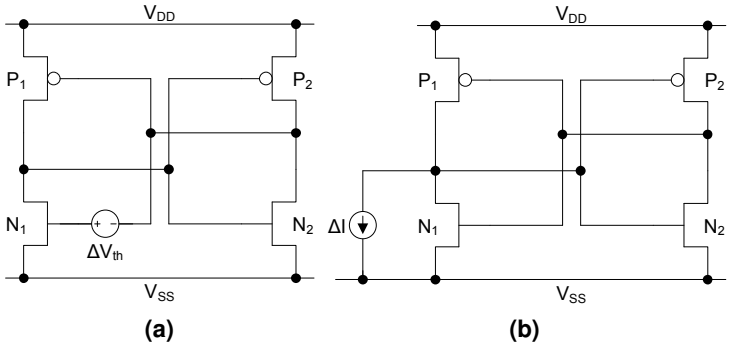


Figure 10.11.: (a) SRAM cell with additional voltage source at the gate of N_1; (b) SRAM cell with additional current source at the drain of N_1.

source and the current source. The figure shows the characteristics of two diode-loaded transistors.

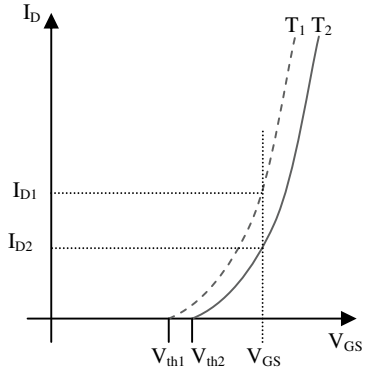


Figure 10.12.: Characteristics of two MOSFETS having different V_{TH}.

Since the two transistors have different V_{TH}, the amount of current through the transistors differs at identical V_{GS}. Thus, additional current parallel to one of the transistors is equivalent to a V_{TH} difference.

A transistor-level implementation of the concept is shown in Figure 10.13. The circuit works as follows: during the first phase N_7 is switched-off and thus no current passes N_5 and N_6. N_3, N_4 and P_1, P_2 are building are building the circuit. N_2 limits the current through the circuit. The circuit is designed in a way that the mismatch between P_1 and P_2 is so small that it does not effect the decision of the cell. Due to the fact that the transistors N_3 and N_4 are diode loaded the circuit does not latch. During the second phase, the signal *decide* is HIGH. Thus, N_7 is switched on and the circuit moves towards the preferred direction. The bias transistors P_3 and P_4, which are used

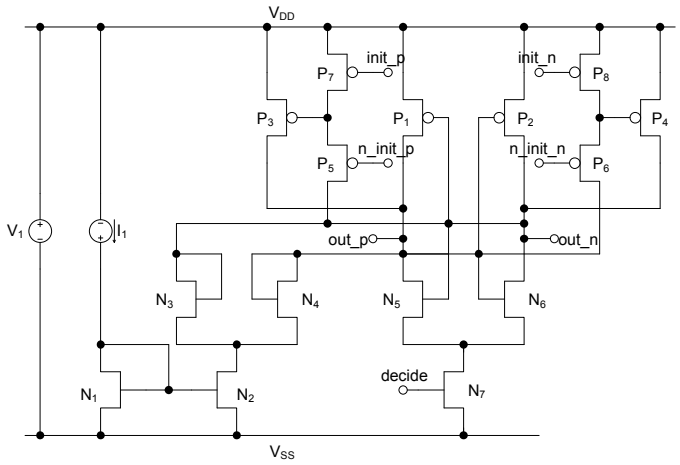


Figure 10.13.: Implementation example of an SRAM-like PUF with pre-selection transistors P_3 and P_4.

for the pre-selection process, are switched-off during the second phase (P_7 and P_8 are switched on; P_5 and P_6 are switched off).

If a offset should be added to the transistor N_4, P_8 is switched off and P_6 is closed. Now, transistor P_4 and transistor P_2 are working in parallel. Therefore, the two transistors act as one transistor having the width of P_2 plus P_4. Since current and width are proportional directly, the current through the right branch increases and thus the voltage at out_n. The same can be done at the right branch considering the transistors P_7, P_5, P_3, and N_3. If more than one pre-selection level is needed, additional transistors can be added in parallel to P_1 and P_2. The effective width is approximated by the sum of the width of all active transistors. Using this system, a binary weighted system can be built. The truth table for the control of the transistors P_5, P_6, P_7, and P_8 is shown in the Table 10.3.

Table 10.3.: Control of the transistors P_5, P_6, P_7, P_8 for adding the current bias to the circuit.

Function	n_{TH}	p_{TH}	$init_p$	$ninit_p$	$init_n$	$ninit_n$
No threshold	0	0	0	1	0	1
P threshold	0	1	1	0	0	1
N threshold	1	0	0	1	1	0

An area improvement compared to the proposed circuit can be realized by separating the mismatching transistors N_3 and N_4 from the rest of the circuit (N_5 to N_7 and P_1 to P_8). To be able to choose between the active mismatch transistors, two additional switches have to be included. With this circuit optimization one PUF cell only consists of five transistors shown in Figure 10.14. The single cells can be read out sequentially using the same evaluation circuit (sense amplifier). After the optimization, one cell is only about the size of on regular SRAM cell which is $100\,F^2$. The disadvantage of such a shared amplifier is the bias on the output that may be caused by mismatches within the sense amplifier. The problem is discussed in Chapter 13 more detailed.

A diagram of a whole pre-selection system can be seen in Figure 10.15. The system works in two different modes. The first mode is the initialization mode. The second mode is the nominal mode. During the initialization mode the pre-selection process takes place. Each cell is biased in both directions. If the output stays constant, the cell is defined as useful. The addresses of the useful cells are stored in a non-volatile memory (NVM). The initialization mode stops as soon as

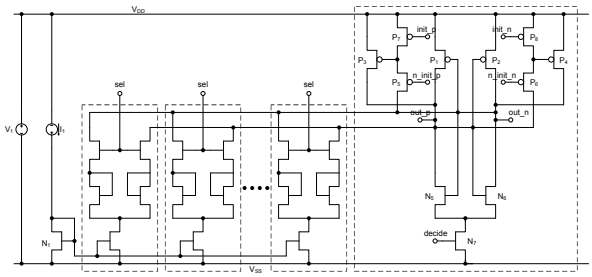


Figure 10.14.: PUF-cells with shared sense amplifier.

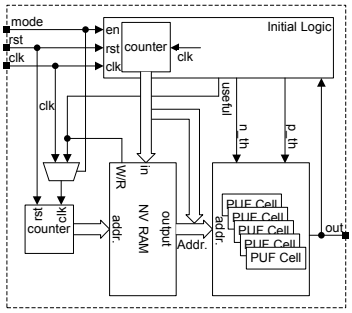


Figure 10.15.: Diagram of PUF system including pre-selection.

the necessary number of output bits is reached. It may happen that a chip does not provide enough useful cells. In this case, an error occurs and the chip is marked as defect. If too many chips are affected, the pre-selection level is chosen too high for the number of available PUF cells. Thus, a solution is to reduce the pre-selection threshold ΔV_{TH-} and ΔV_{TH+} or to increase the number of cells on the chip. If the chip works in the nominal mode, only those cells are read out that are marked as useful in the NVM.

10.2.4. Conclusion

In this section a pre-selection approach was presented that includes an SRAM-like PUF easy to be implemented. Error rates of 10E-12 are possible with this approach. Due to the strong decrease in error rate the post processing can be implemented less complex or it even becomes unnecessary. Furthermore, a smaller error rate permits less power consumption and faster readouts. The additional effort caused by the pre-selection process is small compared to the advantages, since each cell has to be read out only twice during initialization. Furthermore, the initialization can be done at one temperature. The approach was implemented on a 90 nm process. The measurement results are discussed in Chapter 14.

10.3. Pre-Selection Time

One possibility to measure the stability of PUF cells is to measure the time until the cell reaches a certain output voltage. For the following considerations a common 6T SRAM cell is used as a PUF cell (see Figure 10.16). The SRAM cell has some advantages: it is small, consists only of 6 transistors including the word line transistors and the functionality is not very complex. The idea behind the approach is that stable PUF cells need less time for the decision process than unstable cells.

The reason for this behavior lies in the mismatch of the threshold voltage between the involved

transistors. To simplify the analysis the PMOS transistors are assumed to be matching perfectly. Therefore, the decision depends merely on the two NMOS transistors. Furthermore, the error sources like noise, aging, and temperature shifts are assumed to influence the threshold voltage directly and thus, they have influence on the output. If the power is switched on, the voltage at

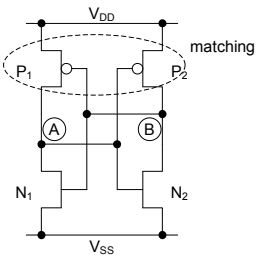


Figure 10.16.: SRAM cell.

the nodes A and B lies somewhere between V_{DD} and V_{SS}, depending on the gate capacitances of the involved transistors. If the threshold voltage of the transistor N_2 is smaller then the V_{TH} of N_1, the current through the right branch is higher than the current through the left branch. Therefore, the voltage in node B is lower than the voltage in node A. If the voltage in node B is smaller, the $V_{GS}$of P_1 is higher and the voltage in point A increases. This is a positive feedback. The higher the difference between the threshold voltages of the transistors is, the faster the decision to a stable point will be. Therefore, a detection circuit has to be implemented to measure the time needed for the decision. In Figure 10.17 the decision phases of different PUF cells are shown. It can be seen that some of the cells decide faster than others. To be able to select cells a positive and a negative threshold value (TH+, TH-) is defined as depicted in the figure. If one of the two threshold values is reached after a predefined time, the respective PUF cell is defined to be useful, otherwise the decision of the cell takes too long. In this case, the cell is not selected.

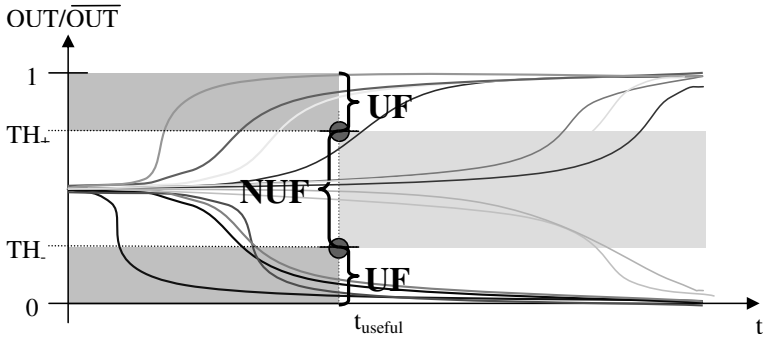


Figure 10.17.: Desicion phase of PUF cells.

10.3.1. Practical realization

In Figure 10.18 a concept of a practical realization is shown. On the left side the PUF cell can be seen. The delay measurement circuit is shown on the right side. The SEL signal activates the PUF cell. Two comparators compare the output value to the threshold voltage at the two output nodes of the PUF cell. Only a positive threshold voltages is needed. The desired output voltage will be reached after some time at one of the two comparators. The two outputs of the comparator are combined using an OR gate. This signal is fed to an arbiter. On the other input pin of the arbiter,

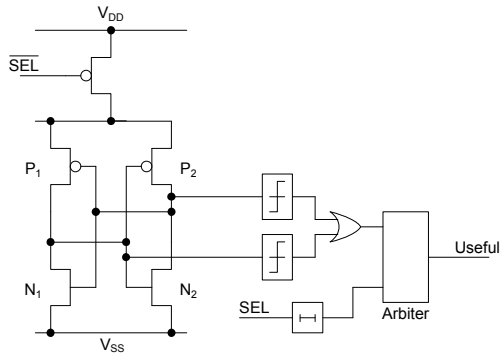


Figure 10.18.: Concept of pre-selection with time.

a delayed version of the SEL signal is provided. The SEL signal, which activates the PUF cell, is delayed by using an appropriate circuit. The arbiter compares the two signals. If the delayed SEL signal arrives later than the output of the comparator, a '1' appears at the output of the arbiter. In this case, the PUF is defined to be useful.

The delay line can be realized using inverters. To vary the delay time transmission gates are used. This is important for circuit tests to find the ideal time t_{useful}. The circuit is shown in the Figure 10.19.

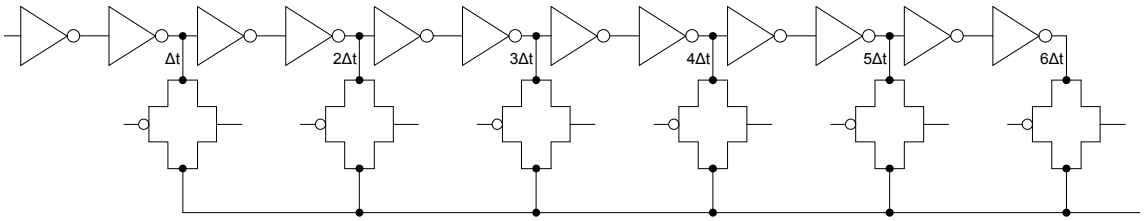


Figure 10.19.: Delay line.

The shorter the delay, the more cells are marked as useful. Furthermore, the shorter the delay, the less stable are the selected cells (see Figure 10.16). Again it is a trade-off between the number of selected cells and the error rate.

A circuit that has the same functionality as the two comparators in and the OR gate in Figure 10.18 is depicted in Figure 10.20. It is more or less a simple OTA with a maximum selector at one side of the differential pair. Transistor N_4 is the current source for the OTA. The transistors N_1, N_2 and N_3 define the differential input pair where the transistors N_1 and N_2 work as maximum selector. The transistor with the higher input voltage is the active transistor. The output of the circuit moves towards HIGH when one of the gate voltages of transistors N_1 and N_2 is higher than the reference voltage V_{TH} at the transistor N_3.

The arbiter can be realized with a NAND (not inverting inputs) RS FF and an inverter at the output. In the Table 10.4 the truth table of the circuit can be seen. If both input signals are '0' the output is reset. If the signal IN_1 arrives before the signal IN_2, the output is '1' as long as both input signals are set to '0'. The circuit which realizes this functionality is depicted in the Figure 10.21 and in the Figure 10.22. Both, the transistor level and the gate level are being shown.

In practice, it may be useful to share the delay measurement circuit.

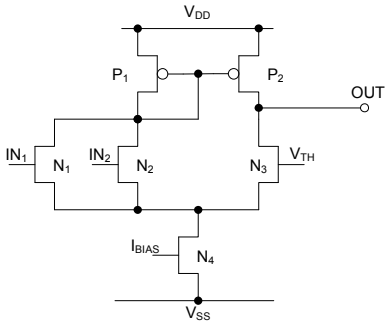


Figure 10.20.: Comparator with maximum selector.

Table 10.4.: Function of the arbiter.

IN_1	IN_2	OUT
0	0	0
0	1	0
1	0	1
1	1	No Change

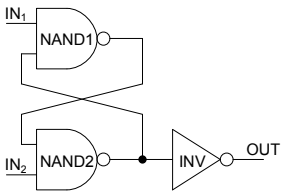


Figure 10.21.: Arbiter gate level.

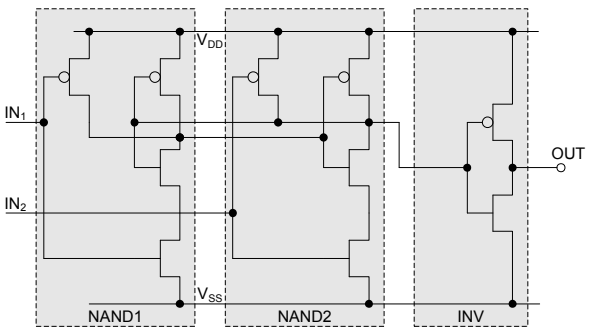


Figure 10.22.: Arbiter transistor level.

10.4. Pre-Selection Using Pre-Charging

In addition to the already described approaches for pre-selection, the PUF cells can be biased using charges which are introduced to the circuit at convenient nodes. This charges influence the decision process and help to find those cells having a small mismatch which are likely to produce errors in the future. In the following section pre-selection techniques for SRAM PUFs are introduced which are based on pre-charging. For reasons of completeness of SRAM PUF pre-selection two further pre-selection approaches are presented.

10.4.1. How to Influence SRAM PUF Behavior

As described in Section 1.6, the SRAM PUF decision depends on the mismatch between the different transistors. The V_{TH}-mismatch of the two PMOS transistors mostly defines the output of the cell. In case of small mismatches between this PMOSTs also the NMOSTs N_1 and N_2 and the various capacitances of the transistors may influence the output. To find cells of a small mismatch each cell is read out once with a biased left branch and once with a biased right branch. If the cell returns the same value for both scenarios, the cell is considered to be stable. The degree of biasing defines the number of stable PUF cells.

To bias an SRAM cell the biasing circuit has to have access to the branches of the cell separately. Biasing of the branches of an SRAM PUF can basically be done over all pins of the SRAM: V_{SS}, V_{DD}, WL, and BL. V_{SS}, V_{DD}, and WL cannot be controlled separately in the original circuit. To be able to use these pins for pre-selection the circuit has to be modified. Only the voltage of BL can be controlled separately for the two different branches which is a big advantage, since SRAMs are usually optimized with respect to the size. Hence, it makes sense to use those circuits without any modification.

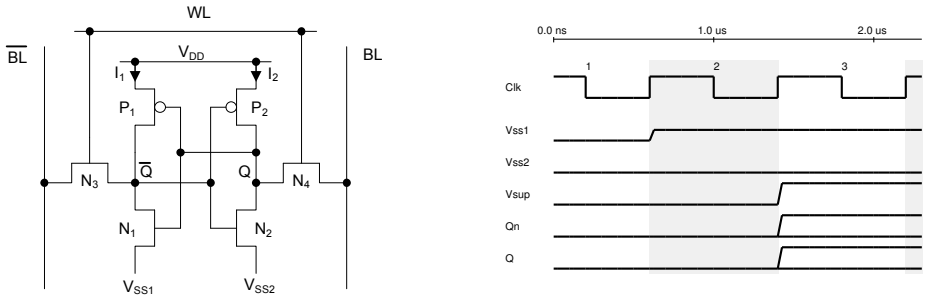
Nevertheless, in the following text all the possibilities are described to get a full overview of pre-selection of SRAM cells. It is important for all the circuits to start with $Vb = 0$ for both branches. This can be done by reseting all nodes before powering-up the SRAM PUF. The way of biasing depends on the pin that is used. A biasing voltage is used for V_{SS}, two different currents are introduced for V_{DD}, and WL and BL are utilized to pre-charge the SRAM circuit.

V_{SS}

If V_{SS} is used for biasing, V_{SS} of one of the two branches has to be increased. Decreasing is not possible, since the source/substrate diode could open. If V_{SS} is increased, V_{GS} is decreased for that transistor and the current flow through N_1/N_2 will delay for that branch. Figure 10.23a shows the circuit and Figure 10.23b shows an example of the timing diagram. Before powering-up the cell, V_{SS} at one of the branches is increased to a predefined value. The approach does not influence the behavior of the important PMOS transistors but it does influence the behavior of the NMOSTs and thus, it influences the decision process only indirectly.

V_{DD}

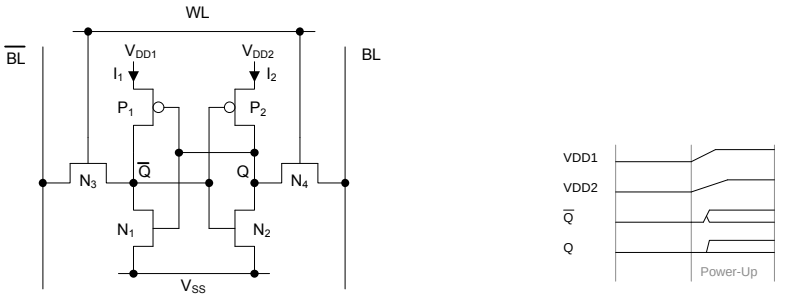
If V_{DD} is used for biasing, the V_{DD} of one branch has to be increased faster than the V_{DD} of the other branch. This is done by introducing two different currents over the V_{DD} pins of the two branches. The biased branch will reach the V_{TH} earlier than during nominal powering-up and thus, it will influence the decision of the cell. Since the powering-up of the circuit is a dynamic process, it is hard to control the current flow exactly. The capacitances of the involved PUF cells will differ mainly for the two branches and they will influence the steepness of the V_{DD} slope.



(a) Pre-selection using VSS pins. E_{med} **(b)** Timing diagram.

Figure 10.23.: Pre-selection using V_{SS} pins.

Figure 10.24a shows the circuit and Figure 10.24b shows an example of the timing diagram. The branch with the steeper slope will tend to move towards V_{DD} .



(a) Schematic. E_{med} **(b)** Timing diagram.

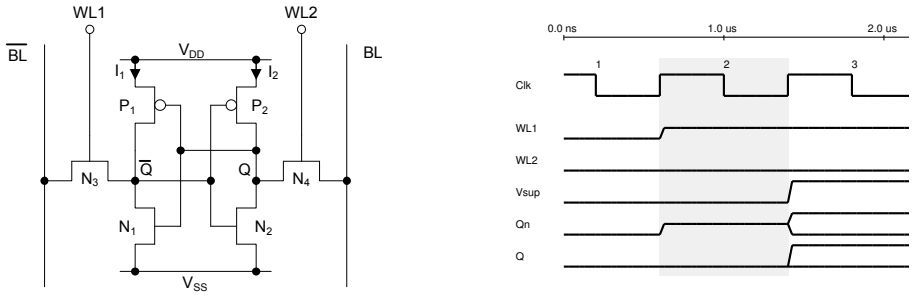
Figure 10.24.: Pre-selection using V_{DD} pins.

WL

The voltage at the WL pins can be used to produce a charge biasing at Q/Qn. To do so, after reseting the circuit, the voltage at one of the WL-transistors is increased slightly. Due to the increasing number of holes at the gate electrode, the voltage at Q/Qn increases. During power-up the V_{GS} of the PMOST of the not-biased branch is smaller and thus the transistor provides current later than under nominal condition. This will force the decision of the biased node towards V_{DD}. Additionally, V_{GS} of the NMOS of the not-biased node is increased which will also move the decision to V_{DD} . Using this approach, it is important to make sure that the V_{TH} of the WL-transistors is not exceeded since this would connect V_b to BL and it thus would destroy the functionality of the PUF. Since the capacitance between gate and source/drain is small, the pre-charge mechanism is quite insensitive to noise on the WL voltage. Figure 10.25a shows the circuit and Figure 10.25b illustrates an example of the timing diagram.

BL

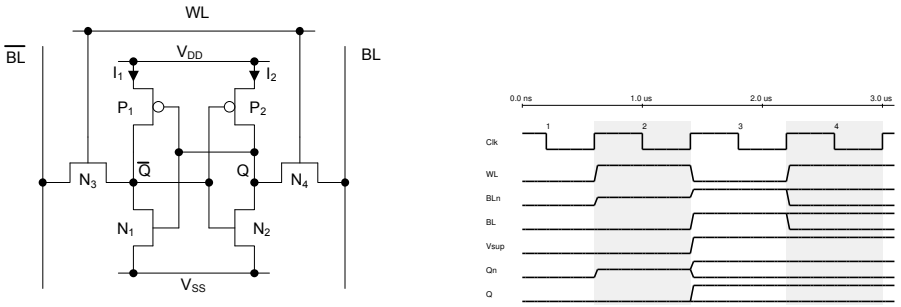
The BL pins are the only pins which can be accessed separately for the two different branches. This makes the pre-selection using BL very attractive, since the circuitry does not have to be changed. As in the case of WL-biasing the biasing using the bit line transistors is based on pre-charging the circuit at Q/Qn. The biasing is done by changing the voltage at BL, which also influences the



(a) Schematic. E_{med} **(b)** Timing diagram.

Figure 10.25.: Pre-selection using WL pins.

voltage at Q/Qn as long as the WL transistor is being opened. Figure 10.26a shows the circuit and Figure 10.26b illustrates an example of the timing diagram.



(a) Schematic. E_{med} **(b)** Timing diagram.

Figure 10.26.: Pre-selection using BL pins.

10.4.2. Closer Look to SRAM PUF Pre-selection Using BL Pin

Since the bit line BL is the only connection that is separated between the two branches of an SRAM, using BL is the most attractive approach to the pre-selection of SRAM cells. For this reason, the approach is described in the following text more detailed.

A schematic view of the circuit can be seen in Figure 10.27.

During the first phase all PUF cells are reset by opening WL and setting the BLs to V_{SS}. Moreover, V_{DD} is set to V_{SS}. Thereafter, one of the bit lines is set to a pre-charge value which changes the voltage at $Q/\bar{Q}$ to that value, since WL is still open. After that, WL is closed and the BLs are set to V_{DD} which is required for the readout process of the cells. Finally, the PUF cell is powered-up by increasing V_{DD} towards the supply voltage. Due to the biasing one branch has a 'head start' compared to the other and therefore, depending on the transistors' properties, the SRAM will change its output and the pre-selection can take place.

In addition to a regular SRAM, two additional features have to be added for pre-selection: to reset and re-powering the SRAM, the supply of the SRAM cells has to be variable (V_{SS} , V_{DD}). Furthermore, a voltage source must be included which provides the pre-charge voltage (V_{pc}) for the pre-selection process. The SRAM block itself needs no adaptation. Figure 10.26b shows the timing diagram such SRAM pre-selection. Here, $\bar{Q}$ is pre-charged and consequently the probability that $\bar{Q}$ moves to V_{DD} during powering-up is higher than moving towards V_{SS}. The diagram shows both possibilities.

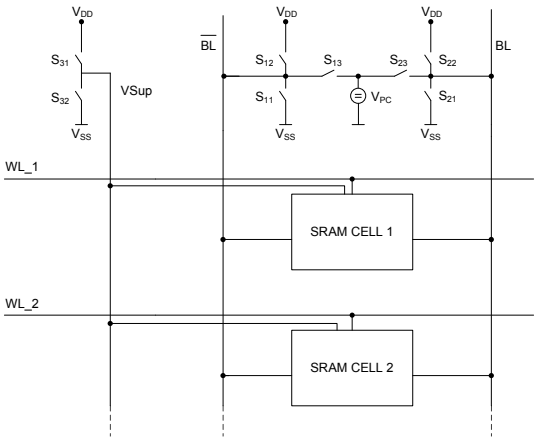
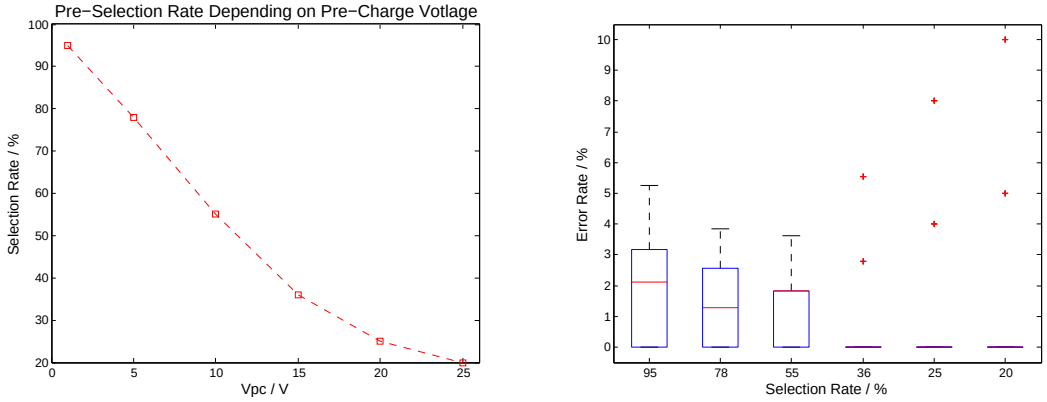


Figure 10.27.: The concept of pre-selection using BL pins.

To test the performance of the pre-selection approach the circuit was implemented in a 350 nm technology and simulated. The results are shown in Figure 10.28.



(a) Pre-selection rate vs pre-charge voltage.

(b) Error rate vs pre-selection rate.

Figure 10.28.: Simulation results using the bit lines for pre-selection purposes.

Figure 10.28a shows the number (percentage) of selected cells in dependence of the pre-charge voltage. The values of the voltage were chosen from 1 mV up to 25 mV. This was done for 100 cells (i.e. Montecarlo runs). The curve does not show any unexpected behavior.

In Figure 10.28b the error rate is shown before and after pre-selection. The simulation was done by using transient noise simulation. For that, the same 50 noise seeds were used with each of the 100 cells (Montecarlo seeds) that were utilized in the selection rate simulation. The median reduces down to 0 % of errorenous bits for the small selection rates (i.e. high pre-charge voltages). Unfortunately, some outlier appear even with strong selection. The reason for that are the properties of the SRAM cells. Noise can have a huge effect during the power-up phase since capacitances and resistances at the nodes Q/Qn are very small. Thus, noise which is composed by Flicker as well as by thermal noise of the transistors is not attenuated up to high frequencies and may influence the decision process of the PUF cell even for cells whith large mismatches.

The results show that pre-selection of SRAM PUF cells does work in general. Unlike other

approaches, outliers in error rates will still exist due to noise peeks. This makes it important to use majority decisions of multiple runs additionally to the pre-selection in order to minimize the effect of noise outliers.

10.5. Summary

Table 10.5 shows the different properties of the pre-processing approaches.

Table 10.5.: Comparison of different pre-selection approaches.

Approach	Hardware complexity	Time	Required NVM	Performance
Pre-selection multiple read	+	-	x	-
Pre-selection delta	x	x	x	+
Pre-selection time	-	x	x	x
Pre-selection charge	x	x	x	x

11. PUF Biasing

by Christoph Boehm and Maximilian Hofer

In the following chapter, the PUF biasing approach will be described. PUF Biasing is an example of pre-processing to reduce the error rate in the future of a PUF. As done in the other pre-processing approaches, biasing is performed once during an initial phase. Thus, the processing during the nominal usage can be reduced in complexity.

To increase the stability of the PUF cells' output, the mismatch between the involved transistors is increased artificially during the initial phase. Hence, the influence of disturbances like noise or shifts in temperature on the output is reduced.

In the following text an SRAM PUF is used to describe the behavior of PUF cells using the biasing technique. As already mentioned the mismatch between the transistors defines its stability in the SRAM PUF, too.

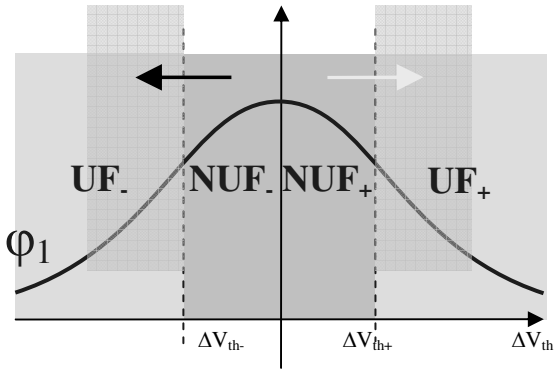


Figure 11.1.: Increasing the mismatch of the PUF cells.

In Figure 11.1 a Gaussian distribution of the mismatch between two transistors is shown. Each pair of transistors represents a PUF cell. Depending on the amount of mismatch, the cells are divided into useful (UF) and not useful (NUF) PUF cells. Additionally, the cells are defined by the sign of their mismatch: the cells are divided into positive (+) mismatch and a negative (-) mismatch pairs. The whole PDF consists of four different types of cells: UF_{-}, NUF_{-}, NUF_{+} and UF_{+}. The areas are shown in Figure 11.1. To increase the stability of the PUF more of the cells should lie in the UF areas. In order to shift all cells into the UF areas the mismatch of the cells in the NUF_{+} area has to be increased by about ΔV_{TH+}, and the mismatch between the cells in the NUF_{-} has to be increased by about ΔV_{TH-}. In a perfect world, the resulting mismatch distribution would look like as it is shown in Figure 11.2a.

Instead of increasing only the mismatch of the cells in the NUF areas, the mismatch of all cells can be changed. Hence, the mismatch of the cells in the '+' areas is increased and and the mismatch of the cells in the '-' areas is decreased. The result is shown in Figure 11.2b.

After the biasing, there are no more cells in the NUF areas. So all PUF cells can be assumed to be stable. Depending on the predefined threshold value, the cells will not be affected by external disturbances during future usage.

In practice, for the following reasons the approach does not work perfectly:

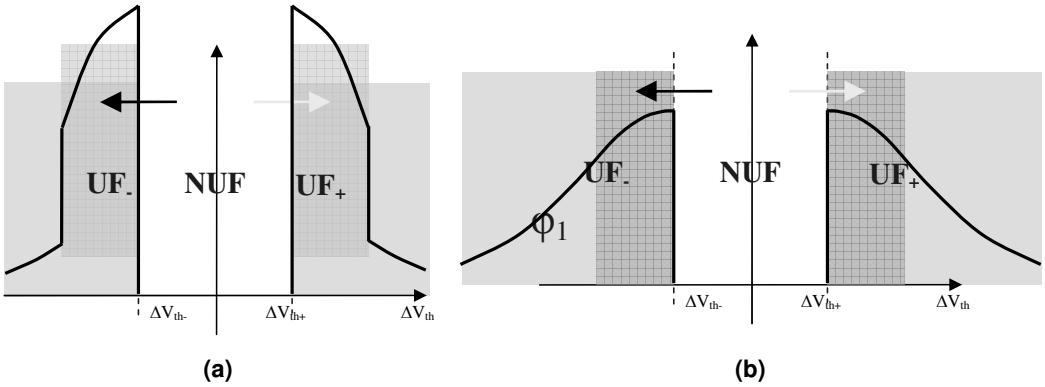


Figure 11.2.: (a) Increase the mismatch of only unuseful PUF cells; (b) Increase the mismatch of all cells.

- Due to noise, the classification between positive and negative mismatches cannot be done perfectly.
- It is not possible to change the mismatch between the transistors by a defined ΔV_{TH}.
- Also the classification between the NUF and the UF areas cannot be done perfectly.

Due to these reasons the resulting PDF would look more like as it is shown in Figure 11.3 in reality. The advantages and disadvantages of the two described biasing approaches is provided in

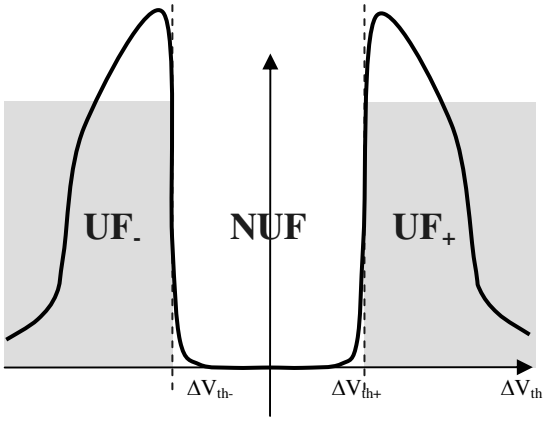


Figure 11.3.: Increase of the mismatch.

the following.

Increase the mismatch of all cells:

+ Cells must not be classified into NUF and UF cells, but only in '-' and '+' cells.
+ The stability of all cells is increased.
- If all cells are aged artificially and attackers could get knowledge about the aged cells by reverse engineering, the whole PUF output could be visible.

Increase the mismatch of the NUF cells:

+ Since only some of the cells are aged artificially, possible attackers cannot access the whole PUF output by analyzing the aged cells.
+ Time and/or energy consumption is lower.
- Cells must be classified into NUF and UF cells.
- More cells are situated within the transition region between the NUF and UF areas.

The biasing of mismatching pairs can be done by utilizing one the different aging processes of transistors. In general, transistor aging may depend on time, temperature, mechanical stress, voltages and currents through the transistors (see 6). Biasing can be done either during an initial phase or during the tests in the semiconductor fab at which the latter should be preferred. During testing, the environmental conditions can be predefined and held constant easily. Furthermore, the required energy can be delivered easily by the tester.

Requirements

To increase the aging effect it is important to be able to change the environmental conditions (temperature) artificially during the initial phase. Otherwise, the time-consumption of the aging process will be high. Furthermore, since the aging process affects all transistors in the same way, biasing should be done only at one of the transistors of a mismatch pair. In the case of the SRAM the aging should affect only one branch. Otherwise the mismatch will not be increased in general. To bias the transistors two different aging effects are presented: NBTI (PBTI) and HCI.

NBTI/PBTI (Negative Bias Temperature Instability/Positive Bias Temperature Instability)

NBTI is one of the main aging effects. NBTI occurs at transistors, if negative gate-source voltages are applied. Thus, mainly p-channel MOSFETs are affected. NBTI causes an increase of the threshold voltage. Though the transistor may recover partly after the negative bias is removed, some of the threshold voltage shift will last (see 6).

For high k-Metal transistors also PBTI occurs. If a positive gate-source voltage is applied to such a transistor, the threshold voltage is increased. Thus, mainly NMOS transistors are affected by PBTI. Apart from that, the effect of PBTI is very similar to that of NBTI (see 6).

Both effects can be used for increasing the mismatch between a pair of transistors. The time-consumption of the biasing depends on the transistor technology that is used.

HCI (Hot Carrier Injection)

HCI is another important aging effect. Especially in small channel transistors the electrons are exposed to high electric fields. Therefore, the electrons are accelerated to a high speed. Such high energetic electrons may also lead to threshold voltage shifts (see 6). Thus, high energetic electrons can be used to increase the mismatch of PUF cells. However, HCI needs high currents to change the threshold voltage effectively. It depends on the PUF circuitry if high currents over long time spans can be made available to the transistors. For SRAM PUFs, HCI-based biasing is not possible, since high currents are not available constantly.

11.1. Modeling and Statistical Aspects

To analyze the performance of the biasing approach Montecarlo mismatch simulation is not feasible, since aging effects are not included in the transistor models. Furthermore, the error rate after

biasing should be so low, so that too many Montecarlo runs would have to be executed. Therefore, the effect of biasing is analyzed theoretically.

For all further analyses, the distribution of the V_{TH} mismatch as well as the distribution of the disturbances (noise, temperature-dependent errors, etc.) are assumed to be Gaussian.

To determine the effect of biasing, the probability distribution function (pdf) f(x) and the cumulative distribution function (cdf) F(x) of the Gaussian distributions are needed:

$$f(x) = \phi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{11.1}$$

$$F(x) = \Phi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx, \tag{11.2}$$

where μ is the mean and σ the variance of the Gaussian.

In the following text the two approaches to biasing will be analyzed separately.

11.1.1. Increase the Mismatch of all Cells

As a first step, the PUF cells are classified into cells with negative and positive mismatch which is shown in Figure 11.4. In Figure 11.4a the ideal classifiaction can be seen. Figure 11.4b shows the realistic Gaussian distributed classification. Due to noise and other disturbances cells near the transition of positive and negative mismatch are partly not classified correctly. If the cells are classified using Φ_2, the classification will end up in the distributions shown in P_+ and P_- depicted in the Figures 11.4b and 11.4c. In the Figures 11.4e and 11.4f, the amount of the positive (ϕ_3) and negative (ϕ_4) mismatch increase due to biasing is shown. Here, the biasing is assumed to be symmetrical. Thus, the absolute values of (ϕ_3) and (ϕ_4) equal. In Figure 11.4g and 11.4h the mismatch distributions of the aged cells are shown separately for the positive and the negative biasing. In Figure 11.4i the joint distribution is depicted.

In the following the occurring distributions are defined:

- $\Phi_1(\sigma_1, \mu_1)$: Mismatch distribution.
- $\Phi_2(\sigma_2, \mu_2)$: Classification distribution ('+' and '-').
- $\Phi_3(\sigma_3, \mu_3)$: Biasing distribution (positive).
- $\Phi_4(\sigma_4, \mu_4)$: Biasing distribution (negative).
- $\Phi_5(\sigma_5, \mu_5)$: Readout distribution (decision '1' or '0').

The following assumptions are made with respect to the mean values: μ_1 is 0 per definition. μ_2 , μ_5 can be assumed to be 0. $\mu_3 = -\mu_4$ depends on the ageing duration.

The following equations are used to estimate the mismatch increase of the cells:

$$P_+ = \phi_1 \cdot \Phi_2 \tag{11.3}$$

$$P_- = \phi_1 \cdot (1 - \Phi_2) = P_- = \phi_1 - \phi_1 \cdot \Phi_2(allother) \tag{11.4}$$

$$P_{+AGED} = P_+ \cdot \phi_3 \tag{11.5}$$

$$P_{-AGED} = P_- \cdot \phi_4 \tag{11.6}$$

Since the Gaussian distribution is invariant with respect to convolution, the following assumptions concerning the mean values can be made:

$$\mu_{13conv} = \mu_1 + \mu_3 \tag{11.7}$$

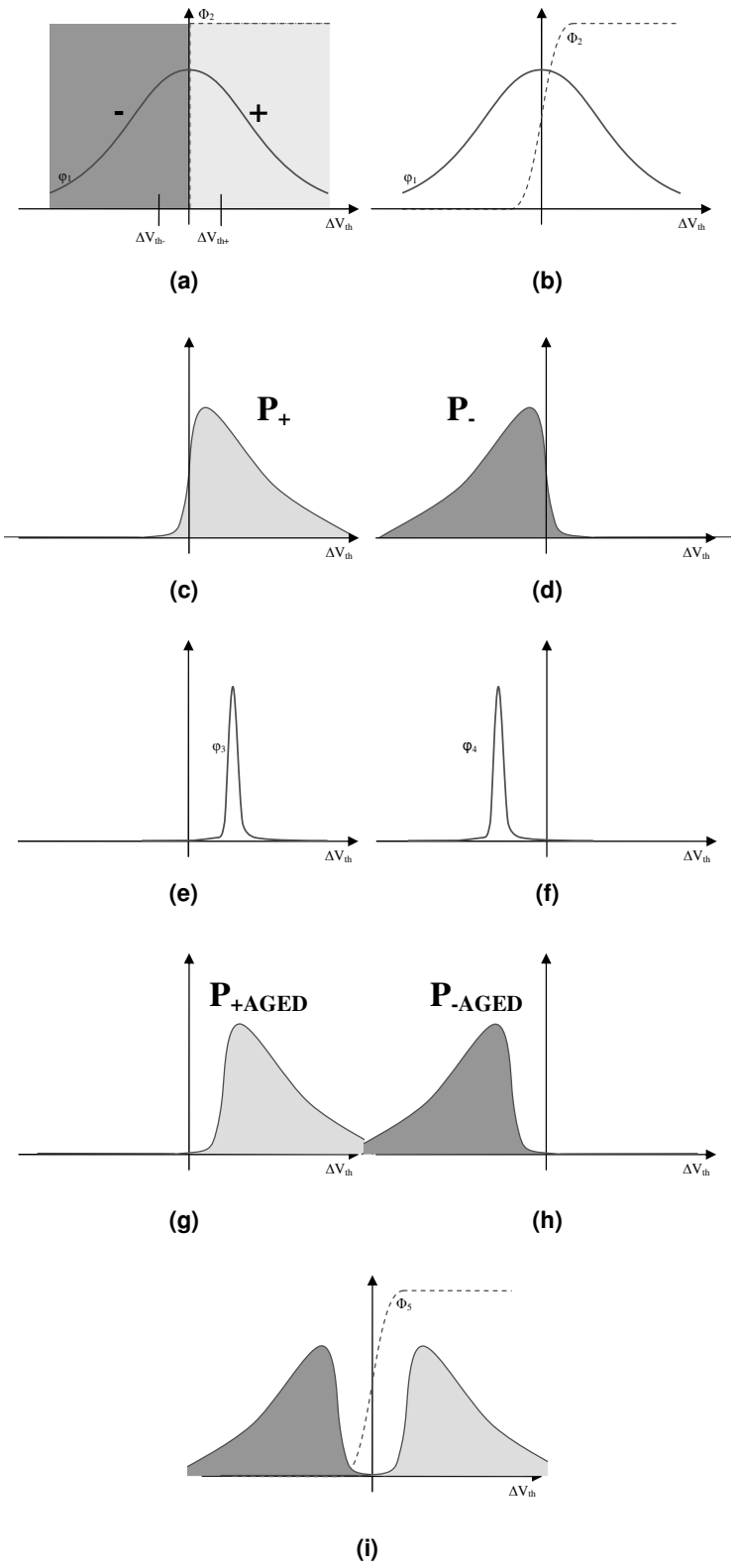


Figure 11.4.: (a) Ideal distributions Φ_1, Φ_2; (b) Real distributions Φ_1, Φ_2; (c) Positive PUF-cells(P_+); (d) Negative PUF-cells(P_-); (e) Positive increase ϕ_3; (f) Negative increase ϕ_4; (g) Positive increased area P_{+AGED}; (h) Negative increased area P_{-AGED}; (i) Distribution of all cells after aging.

$$\mu_{14conv} = \mu_1 + \mu_4 \tag{11.8}$$

$$\mu_{23conv} = \mu_2 + \mu_3 \tag{11.9}$$

$$\mu_{24conv} = \mu_2 + \mu_4 \tag{11.10}$$

$$\sigma_{13conv} = \sqrt{\sigma_1^2 + \sigma_3^2} \tag{11.11}$$

$$\sigma_{14conv} = \sqrt{\sigma_1^2 + \sigma_4^2} \tag{11.12}$$

$$\sigma_{23conv} = \sqrt{\sigma_2^2 + \sigma_3^2} \tag{11.13}$$

$$\sigma_{24conv} = \sqrt{\sigma_2^2 + \sigma_4^2} \tag{11.14}$$

Consequently, the following is true:

$$P_{+AGED} = \phi_{13conv} \cdot \Phi_{23conv} \tag{11.15}$$

$$P_{-AGED} = \phi_{14conv}(1 - \Phi_{24conv}) \tag{11.16}$$

The error after biasing can be derived as follows:

The ratio of erroneous cells is defined as

$$e = \int_{-\infty}^{\infty} \Big(P_{-AGED} \cdot \Phi_5 + P_{+AGED} \cdot (1 - \Phi_5) \Big)\, d\Delta V_{TH}. \tag{11.17}$$

The ration of error free cells is defined as

$$\overline{e} = \int_{-\infty}^{\infty} \Big(P_{+AGED} \cdot \Phi_5 + P_{-AGED} \cdot (1 - \Phi_5) \Big)\, d\Delta V_{TH}. \tag{11.18}$$

The sum of both ratios has to be 1.

$$e + \overline{e} = 1 \tag{11.19}$$

$$\begin{aligned} = \int_{-\infty}^{\infty} \Big(& P_{-AGED} \cdot \Phi_5 + P_{+AGED} \cdot (1 - \Phi_5) + P_{+AGED} \cdot \Phi_5 \\ & + P_{-AGED} \cdot (1 - \Phi_5) \Big)\, d\Delta V_{TH} \end{aligned} \tag{11.20}$$

$$\begin{aligned} = \int_{-\infty}^{\infty} \Big(& P_{-AGED} \cdot \Phi_5 - P_{+AGED} + P_{+AGED} \cdot \Phi_5 \\ & + P_{-AGED} - P_{-AGED}\Phi_5 \Big)\, d\Delta V_{TH} \end{aligned}$$

$$= \int_{-\infty}^{\infty} \Big(P_{+AGED} + P_{-AGED} \Big)\, d\Delta V_{TH}$$

$$= \int_{-\infty}^{\infty} \Big(P_{+} \cdot \Phi_3 + P_{-} \cdot \Phi_4 \Big)\, d\Delta V_{TH}$$

$$= \int_{-\infty}^{\infty} \Big((\phi_1 \cdot \Phi_2) \cdot \Phi_3 + (\phi_1 \cdot (1 - \phi_2)) \cdot \phi_4 \Big)\, d\Delta V_{TH}$$

$$= \int_{-\infty}^{\infty} \Big((\phi_1 \cdot \Phi_2) \cdot \Phi_3 + \phi_1 \cdot \phi_4 - \phi_2 \cdot \phi_1 \cdot \phi_4 \Big)\, d\Delta V_{TH}$$

$$
\begin{aligned}
&= \int_{-\infty}^{\infty} \Big(\phi_1 \cdot \Phi_2\Big)\, d\Delta V_{TH} \cdot \int_{-\infty}^{\infty} \Big(\Phi_3\Big)\, d\Delta V_{TH} \\
&+ \int_{-\infty}^{\infty} \Big(\phi_1\Big)\, d\Delta V_{TH} \cdot \int_{-\infty}^{\infty} \Big(\phi_4\Big)\, d\Delta V_{TH} \\
&- \int_{-\infty}^{\infty} \Big(\phi_2 \cdot \phi_1\Big)\, d\Delta V_{TH} \cdot \int_{-\infty}^{\infty} \Big(\phi_4\Big)\, d\Delta V_{TH}
\end{aligned}
$$

$$
= \int_{-\infty}^{\infty} \phi_1 \cdot \Phi_2\, d\Delta V_{TH} + \int_{-\infty}^{\infty} \phi_1\, d\Delta V_{TH} - \int_{-\infty}^{\infty} \phi_2 \cdot \phi_1\, d\Delta V_{TH}
$$

$$
= \int_{-\infty}^{\infty} \phi_1\, d\Delta V_{TH}
$$

Since Φ_1 is the initial distribution of all cells, the integral results in 1 which had to be shown.

e can be determined as follows:

$$
e = \int_{-\infty}^{\infty} \Big(P_{-AGED} \cdot \Phi_5 + P_{+AGED} \cdot (1 - \Phi_5)\Big)\, d\Delta V_{TH} \tag{11.21}
$$

$$
P_{-AGED} = P_{-} \cdot \Phi_4 \tag{11.22}
$$

$$
P_{-} = \Phi_4 \cdot (1 - \Phi_2) \tag{11.23}
$$

$$
P_{+AGED} = P_{+} \cdot \phi_3 \tag{11.24}
$$

$$
P_{+} = \phi_1 \cdot \Phi_2 \tag{11.25}
$$

$$
e = \int_{-\infty}^{\infty} \Big(\phi_1(1 - \Phi_2) \cdot \phi_4 \cdot \Phi_5 + \phi_1 \cdot \Phi_2 \cdot \phi_3(1 - \Phi_5)\Big)\, d\Delta V_{TH} \tag{11.26}
$$

$$
\begin{aligned}
e = \int_{-\infty}^{\infty} \Big(& \phi_1 \cdot \phi_4 \cdot \Phi_5 - \phi_1 \cdot \Phi_2 \cdot \phi_4 \cdot \Phi_5 \\
& + \phi_1 \cdot \Phi_2 \cdot \phi_3 - \phi_1 \cdot \Phi_2 \cdot \phi_3 \cdot \Phi_5 \Big)\, d\Delta V_{TH}
\end{aligned} \tag{11.27}
$$

$$
\begin{aligned}
e = \int_{-\infty}^{\infty} \Big[\;
& \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2} \cdot \frac{1}{\sigma_4\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_4}{\sigma_4})^2} \cdot \frac{1}{\sigma_5\sqrt{2\pi}} \cdot \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_5}{\sigma_5})^2} dx \\
& - \phi_1 \frac{1}{\sigma_2\sqrt{2\pi}} \cdot \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2} dx \cdot \frac{1}{\sigma_4\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_4}{\sigma_4})^2} \\
& - \phi_1 \frac{1}{\sigma_5\sqrt{2\pi}} \cdot \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_5}{\sigma_2})^5} dx \\
& + \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2} \cdot \frac{1}{\sigma_2\sqrt{2\pi}} \cdot \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2} dx \cdot \frac{1}{\sigma_3\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_3}{\sigma_3})^2} \\
& - \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2} \cdot \frac{1}{\sigma_2\sqrt{2\pi}} \cdot \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2} dx \cdot \frac{1}{\sigma_3\sqrt{2\pi}} \cdot \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_3}{\sigma_3})^2} dx \\
& - \phi_1 \frac{1}{\sigma_5\sqrt{2\pi}} \cdot \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_5}{\sigma_2})^5} dx \Big]\, d\Delta V_{TH}
\end{aligned} \tag{11.28}
$$

$$
\begin{aligned}
e = & \frac{1}{\sigma_1(2\pi)^{1.5}} \int_{-\infty}^{\infty} \Bigg[\\
& \frac{1}{\sigma_4\sigma_5} e^{(-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2 - \frac{1}{2}(\frac{x-\mu_4}{\sigma_4})^2)} \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_5}{\sigma_5})^2} dx \\
& - \frac{1}{\sigma_2\sigma_4\sigma_5\sqrt{2\pi}} e^{(-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2 - \frac{1}{2}(\frac{x-\mu_4}{\sigma_4})^2)} \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2} dx \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_5}{\sigma_5})^2} dx \\
& + \frac{1}{\sigma_2\sigma_3} e^{(-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2 - \frac{1}{2}(\frac{x-\mu_3}{\sigma_3})^2)} \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2} dx \\
& - \frac{1}{\sigma_2\sigma_3\sigma_5\sqrt{2\pi}} e^{(-\frac{1}{2}(\frac{x-\mu_1}{\sigma_1})^2 - \frac{1}{2}(\frac{x-\mu_3}{\sigma_3})^2)} \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_2}{\sigma_2})^2} dx \int_{-\infty}^{x} e^{-\frac{1}{2}(\frac{x-\mu_5}{\sigma_5})^2} dx \\
& \Bigg] d\Delta V_{TH}
\end{aligned}
\tag{11.29}
$$

Matlab Implementation: In Figure 11.5 the results of the numerical Matlab implementation are shown. The mismatch between the two transistors is assumed to be 30 mV (σ_1). This is a realistic value for a minimum size transistor in a deep sub-micron technology. The higher σ_1, the lower the error rate. Further important parameters are the distribution of the decision process (Φ_2, Φ_5) and the distribution of the aging ($\phi 3$, $\phi 4$). For the latter, the mean value (μ_3, μ_4) is the crucial parameter. The values which are used in the simulation are shown in Table 11.1.

Table 11.1.: Parameter values for Matlab simulation.

	σ	μ
Φ_1	σ_1=30 mV	μ_1=0 mV
Φ_2	σ_2=0.01 mV	μ_2=0
Φ_3	σ_3=0.01 mV	μ_3=0.02 mV
Φ_4	σ_4=0.01 mV	μ_4=-0.02 mV
Φ_5	σ_5=0.01 mV	μ_5=0

In Figure 11.6 the most interesting curves are plotted. In Figure 11.6a the original distribution (ϕ_1), the distribution of the aged cells P_{+AGED} and P_{-AGED}, and the distribution of the PUF output (Φ_5) are shown.

11.1.2. Increase the Mismatch of NUF Cells Only

The difference to the approach suggested above is that here only some of the cells are being treated. The biasing is done for such cells that are maked as NUF in the first place.

As a first step, the cells are classified into cells with positive and negative mismatch. In a second classification step, the cells are grouped into useful (UF) and not useful (NUF) cells. Due to their small mismatch NUF cells are not assumed to produce a reliable output value.

In the following list the occurring distributions are defined:

- $\Phi_1(\sigma_1, \mu_1)$: Mismatch distribution
- $\Phi_2(\sigma_2, \mu_2)$: Classification distribution (+/-)
- $\Phi_3(\sigma_3, \mu_3)$: Classification distribution (UF+/NUF+)

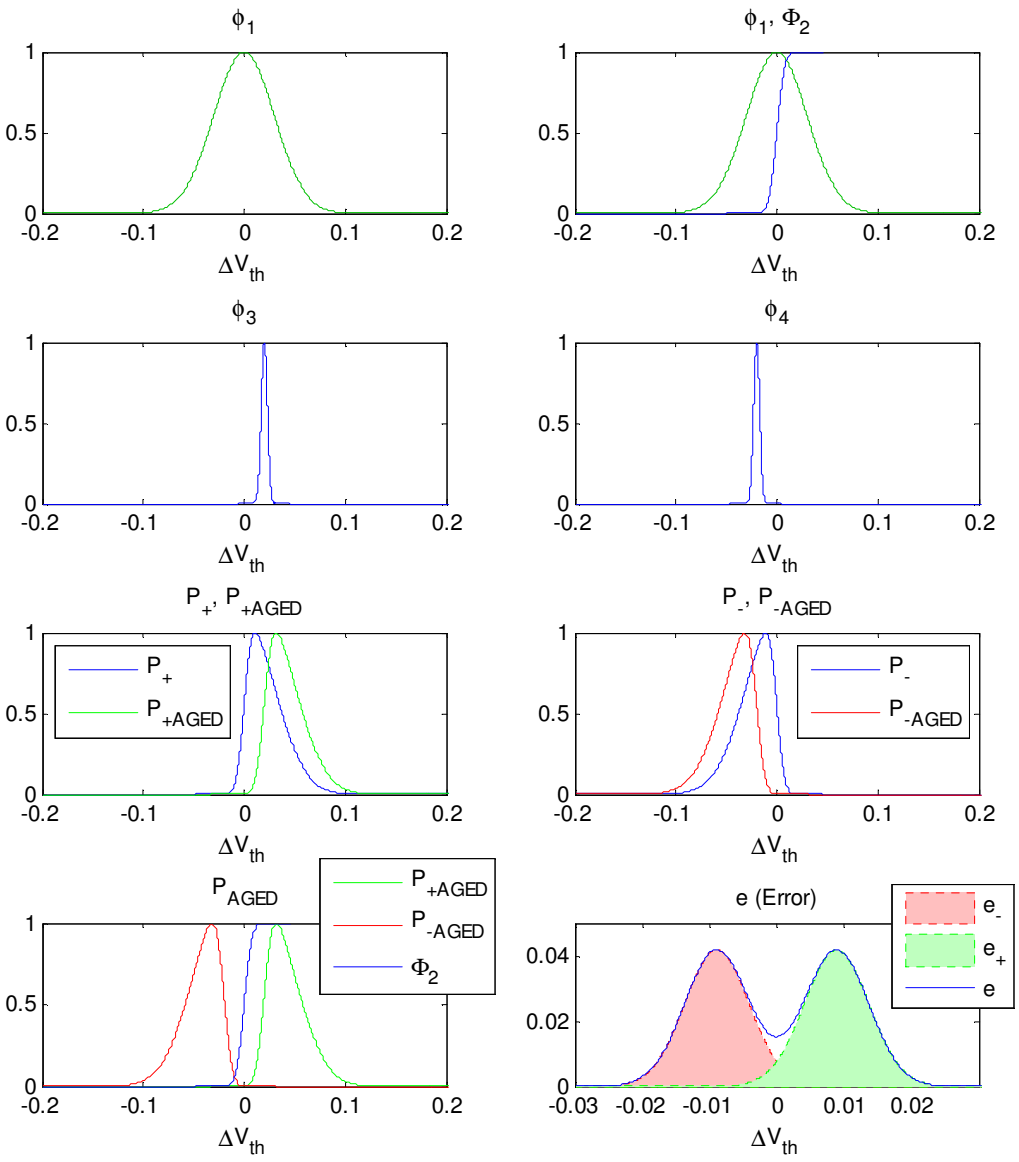
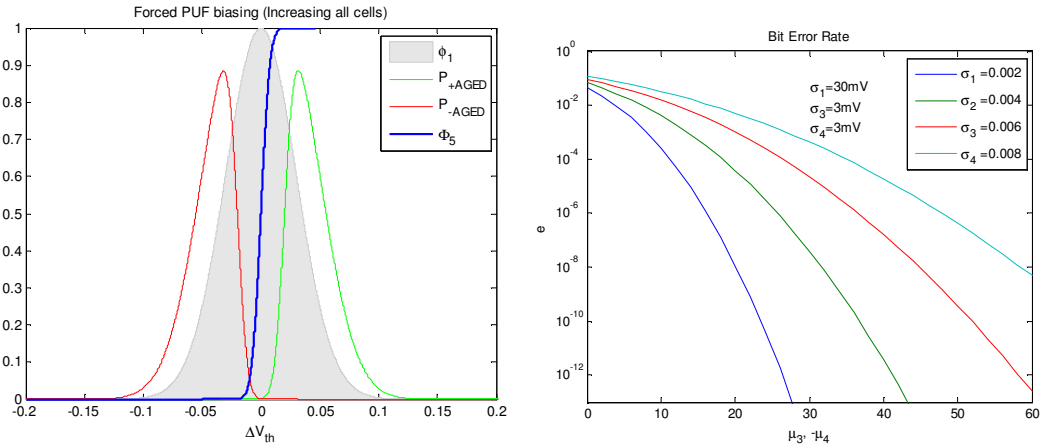


Figure 11.5.: Matlab simulation of increasing all cells.



(a) (b)

Figure 11.6.: (a) Matlab simulation results for mismatch increase of all cells; (b) Error rate.

- $\Phi_4(\sigma_4, \mu_4)$: Classification distribution (UF-/NUF-)
- $\Phi_5(\sigma_5, \mu_5)$: Aging distribution (+)
- $\Phi_6(\sigma_6, \mu_6)$: Aging distribution (-)
- $\Phi_7(\sigma_7, \mu_7)$: Readout distribution

The following equations are used to estimate the mismatch increase of the cells after biasing:

$$P_{+NUF} = \phi_1 \cdot \Phi_2(1 - \Phi_3) \tag{11.30}$$

$$P_{-NUF} = \phi_1 \cdot \Phi_4(1 - \Phi_2) \tag{11.31}$$

$$P_{+NUFaged} = P_{+NUF} \cdot \phi_5 \tag{11.32}$$

$$P_{-NUFaged} = P_{-NUF} \cdot \phi_6 \tag{11.33}$$

Since the Gaussian distribution is invariant with respect to convolution, the following assumptions concerning the mean values can be made:

$$\mu_{15conv} = \mu_1 + \mu_5 \tag{11.34}$$

$$\mu_{16conv} = \mu_1 + \mu_6 \tag{11.35}$$

$$\mu_{25conv} = \mu_2 + \mu_5 \tag{11.36}$$

$$\mu_{35conv} = \mu_3 + \mu_5 \tag{11.37}$$

$$\mu_{26conv} = \mu_2 + \mu_6 \tag{11.38}$$

$$\mu_{46conv} = \mu_4 + \mu_6 \tag{11.39}$$

$$\sigma_{15conv} = \sqrt{\sigma_1^2 + \sigma_5^2} \tag{11.40}$$

$$\sigma_{16conv} = \sqrt{\sigma_1^2 + \sigma_6^2} \tag{11.41}$$

$$\sigma_{25conv} = \sqrt{\sigma_2^2 + \sigma_5^2} \tag{11.42}$$

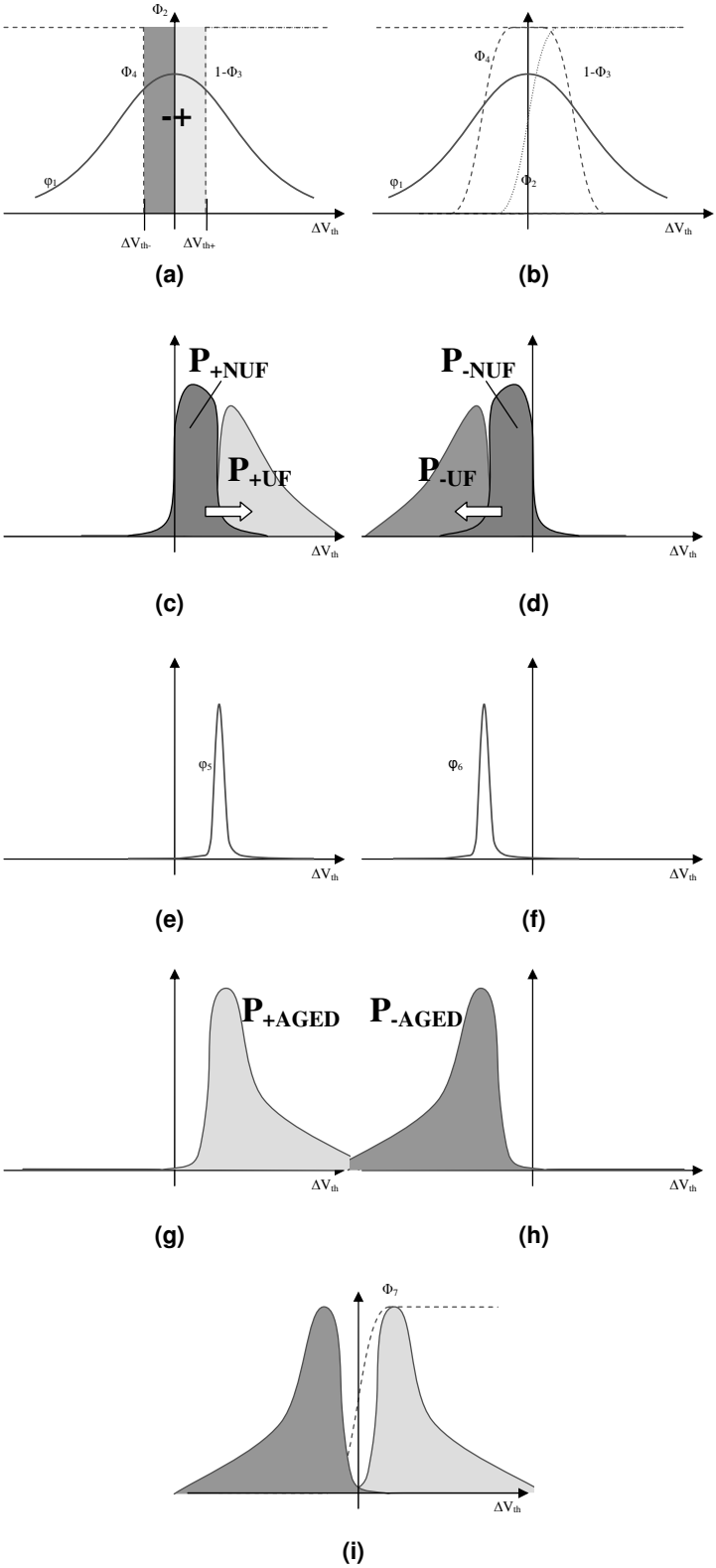


Figure 11.7.: (a) Ideal distributions $\Phi_1 - \Phi_4$;(b) Real distributions $\Phi_1 - \Phi_4$; (c) Useful positive PUF-cells(UF+); (d) Useful negative PUF-cells(UF-); (e) Positive increase ϕ_5; (f) Negative increase ϕ_6; (g) Positive increased area P_{+AGED}; (h) Negative increased area P_{-AGED}; (i) Distribution of all cells.

$$\sigma_{35conv} = \sqrt{\sigma_3^2 + \sigma_5^2} \tag{11.43}$$

$$\sigma_{26conv} = \sqrt{\sigma_2^2 + \sigma_6^2} \tag{11.44}$$

$$\sigma_{46conv} = \sqrt{\sigma_4^2 + \sigma_6^2} \tag{11.45}$$

Thus, the equations can be formulated as follows:

$$P_{+NUFaged} = \phi_1 \cdot \Phi_2(1 - \Phi_3) \cdot \phi_5 = \phi_{15} \cdot \Phi_{25}(1 - \Phi_{35}) \tag{11.46}$$

$$P_{-NUFaged} = \phi_1 \cdot \Phi_4(1 - \Phi_2) \cdot \phi_5 = \phi_{16} \cdot \Phi_{46}(1 - \Phi_{26}) \tag{11.47}$$

This simplification is important to gain a significant increase in performance of numerical calculation.

$$P_{+UF} = \phi_1 \Phi_2 \Phi_3 \tag{11.48}$$

$$P_{-UF} = \phi_1(1 - \Phi_4)(1 - \Phi_2) \tag{11.49}$$

$$P_{+AGED} = P_{+NUFaged} + P_{+UF} \tag{11.50}$$

$$P_{-AGED} = P_{-NUFaged} + P_{-UF} \tag{11.51}$$

$$e = \int_{-\infty}^{\infty} \Big(P_{-AGED}\Phi_7 + P_{+AGED}(1 - \Phi_7) \Big)\, d\Delta V_{TH} \tag{11.52}$$

$$\bar{e} = \int_{-\infty}^{\infty} \Big(P_{-AGED}(1 - \Phi_7) + P_{+AGED}\Phi_7 \Big)\, d\Delta V_{TH} \tag{11.53}$$

Matlab Implementation: In Figure 11.8 the results of an numerical Matlab implementation are shown. Again, the threshold voltage mismatch between two transistors is assumed to be 30 mV. The residual parameter values, which are used in the simulation, are shown in Table 11.2.

Table 11.2.: Parameter values for Matlab simulation.

	σ	μ
Φ_1	σ_1=30 mV	μ_1=0 mV
Φ_2	σ_2=0 mV	μ_2=0
Φ_3	σ_3=0 mV	μ_3=0 mV
Φ_4	σ_4=0 mV	μ_4=0 mV
Φ_5	σ_5=0 mV	μ_5=0
Φ_6	σ_6=0 mV	μ_6=0
Φ_7	σ_7=0 mV	μ_7=0

In Figure 11.9 the most interesting curves are depicted. Figure 11.10 shows the error rate after biasing. It becomes clear that both, the amount of aging and the classification parameters (UF/NUF) must be chosen carefully. The reason for the kink in the graphs is the dominant effect of UF/NUF classification over the biasing in the case where the amount of biasing exceeds a certain value. For this case, the unbiased cells define the error behavior.

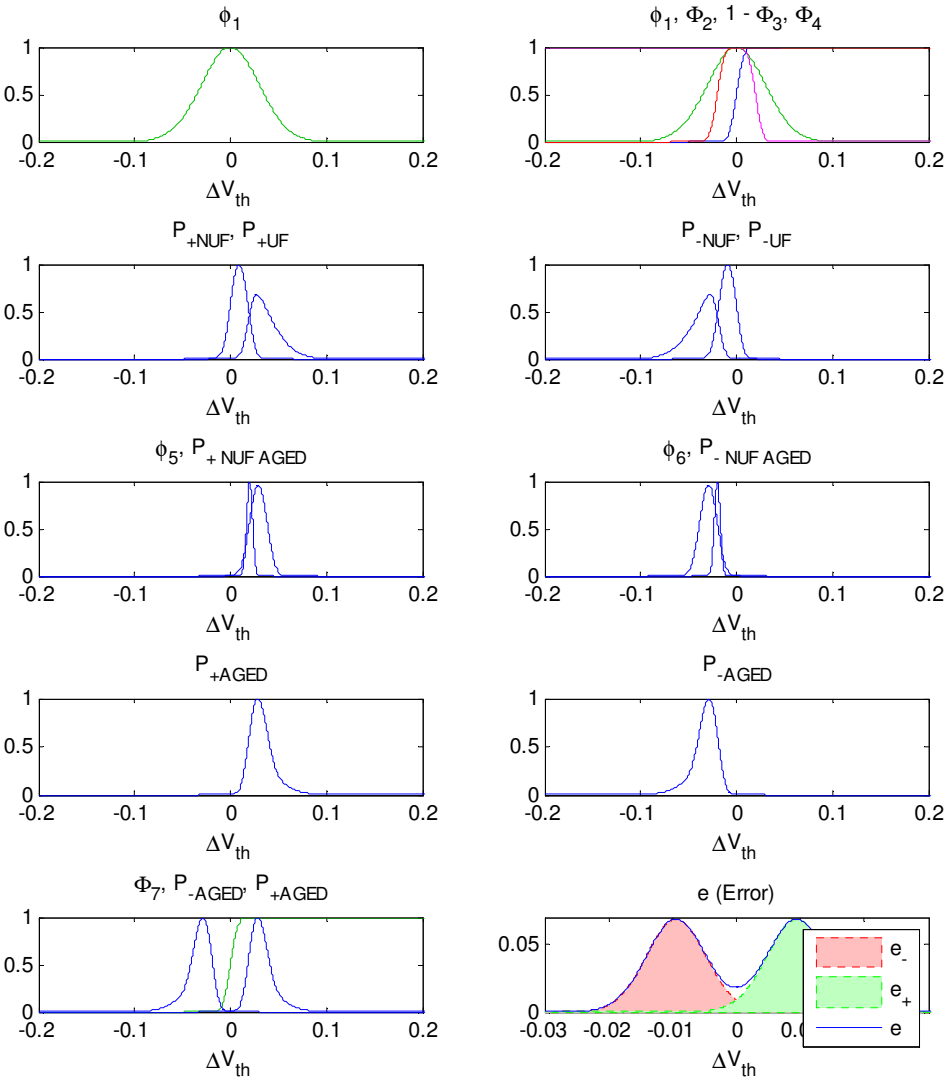


Figure 11.8.: Matlab simulation of increasing only NUF cells.

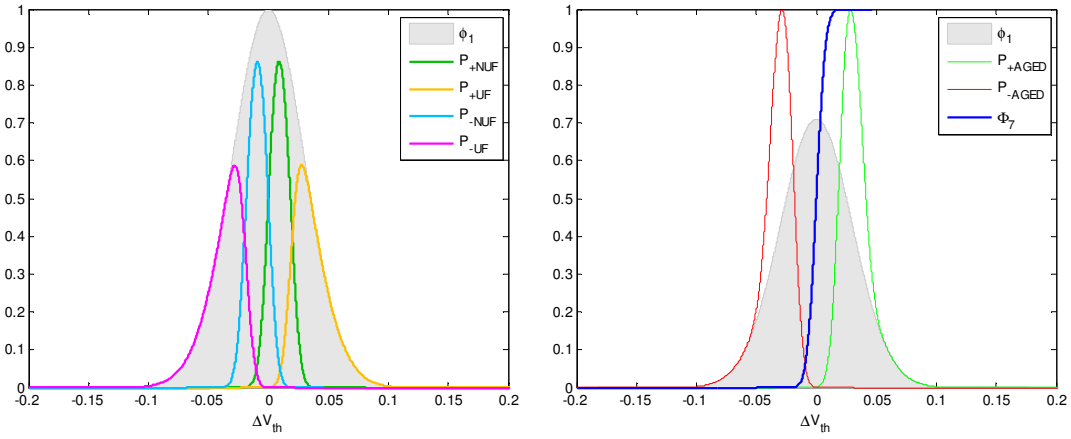


Figure 11.9.: (a) Matlab simulation results of partial biasing; (b) Matlab simulation of increasing only NUF cells.

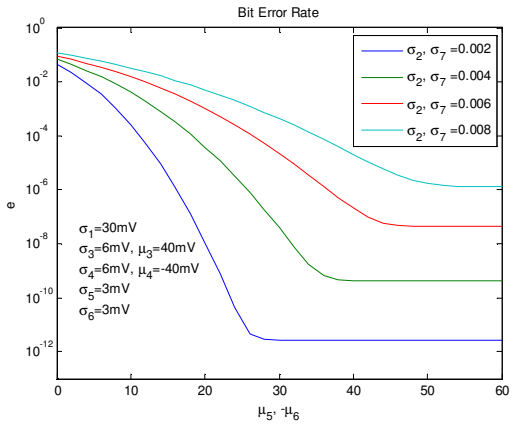


Figure 11.10.: Error rate after partial biasing.

11.2. Realization

To implement this approach the process properties have to be taken into account. Thus, it is very important to know the aging behavior of the process as accurate as possible. If this is the case, an appropriate aging process must to be chosen. To make sure that the circuit works, the following properties should be fulfilled:

- The possible V_{TH} increase must be sufficient (at least ΔV_{TH}).
- The mismatch increase must be permanent.
- The aging process should be as fast as possible (to reduce the time-consumption of the initial phase).
- The aging process should influence the other parts of the circuit as little as possible to guarantee circuit functionality.

11.2.1. Stressing Circuitry Utilizing HCI

In Figure 11.11a a circuit is shown that utilizes HCI to increase the mismatch between the transistors. The circuit in Figure 11.11b shows how it is possible to stress the transistor P_1 in a modified

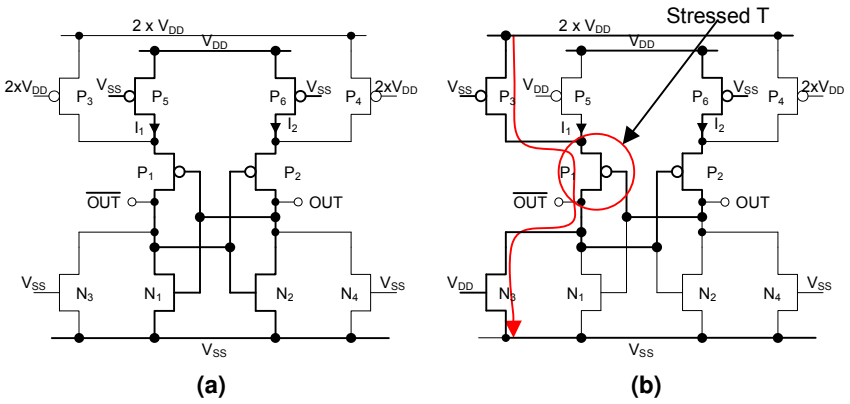


Figure 11.11.: (a) Circuit to stress the Transistors with HCI; (b) Ageing of the cell.

Table 11.3.: Stress P_1 or P_2.

	V_{DD}	Stress P_1	Stress P_2
N_3	V_{SS}	V_{DD}	V_{SS}
N_4	V_{SS}	V_{SS}	V_{DD}
P_3	$2xV_{DD}$	V_{SS}	$2xV_{DD}$
P_4	$2xV_{DD}$	$2xV_{DD}$	V_{SS}
P_5	V_{SS}	V_{DD}	V_{SS}
P_6	V_{SS}	V_{SS}	V_{DD}

SRAM cell. To do so, the transistors P_3 and N_3 are used as switches. In this example the transistor P_1 is connected to $2xV_{DD}$ minus the potential drop over the switches. The voltage at the gate of P_1 is approximately V_{DD}. A table showing the configurations for transistor stressing is shown in Table 11.3.

11.2.2. Stressing Circuitry Utilizing NTBI

An equivalent circuit can be realized for NBTI stressing. In Figure 11.12 a circuit to stress the cells utilizing NTBI is shown.

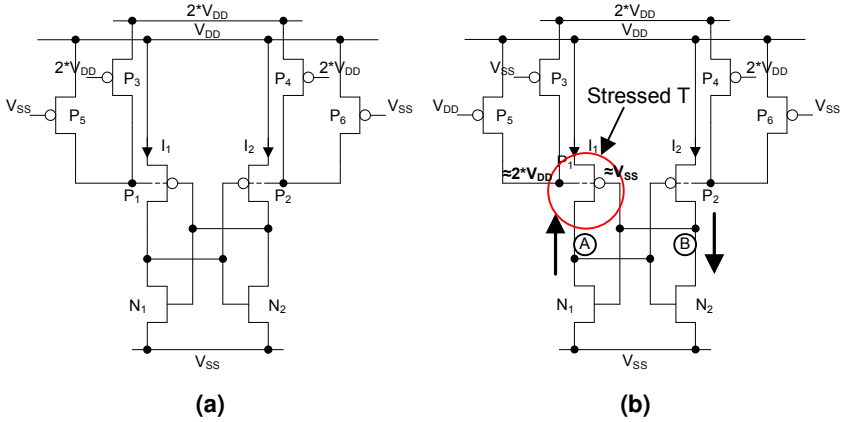


Figure 11.12.: (a) Circuit to stress the Transistors with NTBI; (b)Stress of P_1.

As an example, the aging process of P_1 is explained. The situation is depicted in Figure 11.12b. It is assumed that the cell is powered up and it therefore has already settled to a stable state: In this example, node A is at V_{DD} (minus voltage at transistor P_2: $V_{DS\,P2}$) and node B is at V_{SS} (plus voltage on transistor P_1: $V_{DS\,P1}$). The transistors P_3 and P_6 are closed and the transistors P_4 and P_5 are open. The voltage on the bulk of P_1 is $\approx 2V_{DD}$ and the voltage on the gate of P_1 is $\approx V_{SS}$. Hence, there is a perceptible difference between the gate and the bulk of these transistors. This is necessary for NBTI to appear.

In Table 11.4, the potentials at the different transistors are shown for the three types of operation: normal, stressing P_1, and stressing P_2.

Table 11.4.: Stress P_1 or P_2.

	V_{DD}	Stress P_1	Stress P_2
P_3	$2xV_{DD}$	V_{SS}	$2xV_{DD}$
P_4	$2xV_{DD}$	$2xV_{DD}$	V_{DD}
P_5	V_{SS}	V_{DD}	V_{SS}
P_6	V_{SS}	V_{SS}	V_{DD}

Circuit to Classify UF and NUF Cells

A simple circuit to divide the useful and unuseful PUF cells is presented in Chapter 10.2 (see Figure 10.13). The introduced circuit can be combined with the aging circuitry to provide full functionality.

11.3. Summary

In this chapter a method to increase the stability of PUF cells was presented. This was done by using an artificial aging process. To get an appropriate error rate a V_{TH} change of some millivolts

is necessary. One big advantage of this approach is that the stability of the cells can be increased without using an NVM.

Part III.

Practical Realizations

12. Two Stage PUF

by Christoph Boehm

The first chip that is introduced is a two-phase PUF chip. It utilizes the mismatch between two current sources to produce an output. The chip was produced in a 90 nm technology. The design was done by Marco Bucci and Raimondo Luzzi from Infineon Technologies Austria AG (see US patent application US020120072476A1). Each chip consists of 4096 PUF cells. Each of the cells provides one bit of output. The cells are read out serially. The frequency is chosen to be 1 MHz. The cell size is approximately 2500 F^2. A photograph of the chip is shown in Figure 12.1.

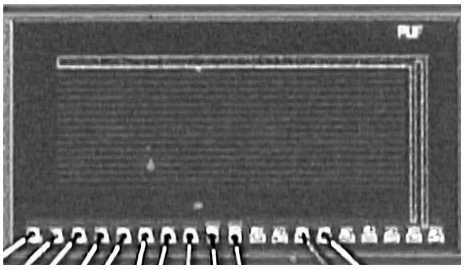


Figure 12.1.: Photograph of the testchip.

12.1. Circuit

In order to reduce the influence of noise on the PUF's output, the decision process is done within two phases: The amplification phase and the evaluation phase. Both phases are shown in the timing diagram in Figure 12.2a. Figure 12.2b provides a schematic of the circuit.

During the amplification phase the switches trig_n are closed. The mismatch ΔI is amplified at the positive feedback decision circuit (load). Thus, the differential voltage between out_p and out_n is proportional to the offset current ΔI. The very high load impedance amplifies the difference strongly but without latching the output. Unlike a SRAM PUF in which noise during the power-up phase may define the final output, the influence of noise during the amplification phase is strongly reduced.

As soon as the trigger input is set to V_{SS}(trig_ n = 0), the evaluation phase starts: The two switches below the diodes open and the positive feedback pushes the outputs towards V_{DD} and V_{SS} respectively (see Figure 12.2a). Therefore, offset amplification and decision/digitization are two separate phases.

During the amplification phase, the circuit is insensitive to noise and settles to a value which is mainly defined by ΔI. At the end of this phase, the amplified offset has reached a maximum value. This is the moment of maximal signal-to-noise ratio. Now, the signal is converted into a digital value. If noise should still be an issue, additional capacitances can be used at the outputs out_p and out_n to reduce the thermal noise during the amplification phase.

The complete schematic of the PUF cell is shown in Figure 12.3. P_1 and P_2 are the current sources which are optimized for maximum mismatch and minimal influence of gradients over the wafer (e.g. gradients on oxide thickness). This is done by minimal transistor sizes and common

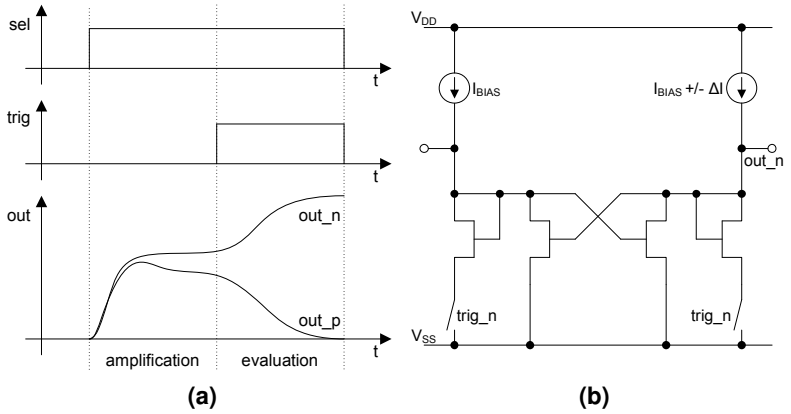


Figure 12.2.: Two stage PUF (a) Timing diagram; (b) Concept.

centroid layout. P_3 and P_4 act as cascode transistors. N_3 and N_5, and N_4 and N_6 are building the positive feedback decision circuit during the amplification phase. The transistors N_7 and N_8 are working as switches. N_9 and N_{10} are additional transistors to equalize the voltage drop at N_7 and N_8. The transistors N_5 and N_6 in combination with the current sources are building a latch during the evaluation phase.

If the cell is not selected, the transistors N_1 and N_2 are switched on and the node in between is forced to zero. Due to the fact that the transistors N_1 and N_2 are working as switches and the transistors P_5 and P_6 are cutting the current sources from the load, the output nodes are forced to V_{SS} and the cell is reseted.

12.2. Measurement Results

The measurement results of the presented test chip were used in Chapter 4 to demonstrate the different specification parameters. For the sake of completeness the most important results are presented again. The PUF chips of four wafers were tested directly on the wafer at different environmental conditions. On each wafer 80 chips were analyzed to get a huge amount of data for statistical evaluation. Furthermore, 10 chips were put in packages and measured under different environmental conditions.

Mean Value

In Figure 12.4 the mean values of the 4096 output bits are being shown. The blue curve shows the ideal binomial distribution. The mean values of the ten packaged test chips $\overline{x}$ lie within a statistical accurate range.

Error Rate

The error rate of the test chips at 20°C is shown in Figure 12.5a.

The biggest error is at 120°C. At 20°C the distribution is binomial. This is obvious because the errors occur due to the influence of noise. The BERs at -40°C and 120°C are not binomial distributed. The errors are composed of the noise (which is almost in the same range than at 20°C) and a change of the sign of the mismatch in some PUF cells. Thus, the error at upper and lower temperature corners have the same deviation but another mean. So it is better to assume the BER at the temperature corners as normal distributed.

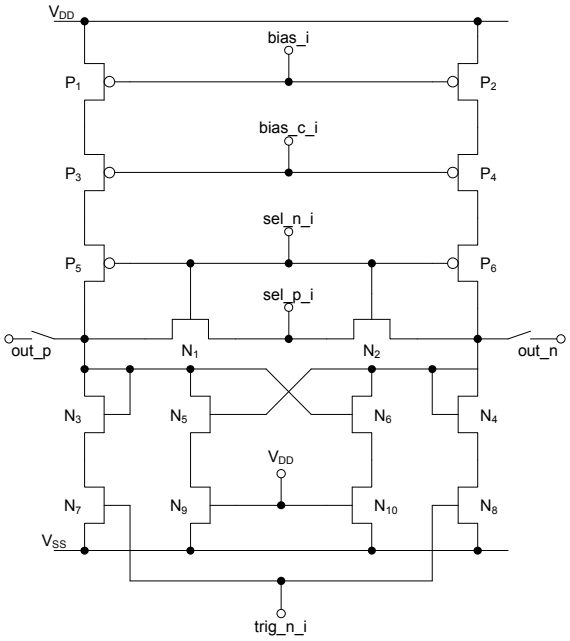


Figure 12.3.: Circuit of the two stage PUF.

In Figure 12.5b the BER between -40°C and 120°C is shown. Here, the reference vector was chosen at 20°C. The graph nicely shows the nearly linear relationship between temperature and error rate.

Correlation Between Bits

The correlation between bits is shown in Figure 12.6a. There does no correlation exist between the bits of the chips.

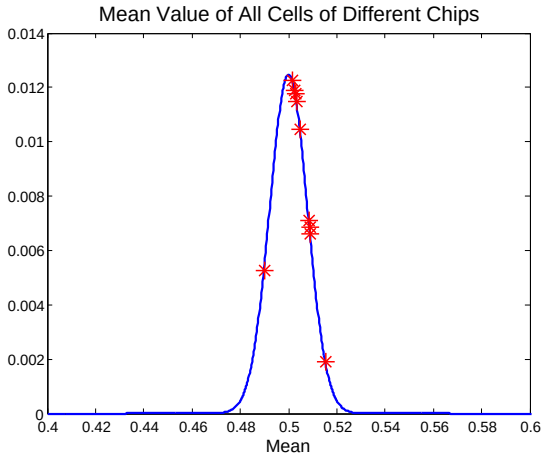


Figure 12.4.: Mean value $\overline{x}$ of the different chips of the two-stage PUF.

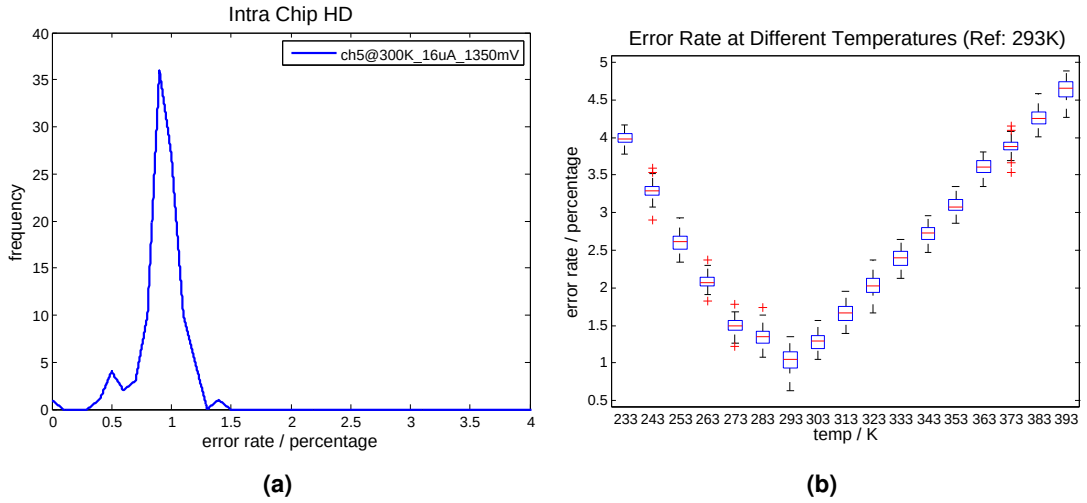


Figure 12.5.: Two stage PUF; (a) Intra chip Hamming distance HD_{intra} (100 runs); (b) HD_{intra} at different temperatures.

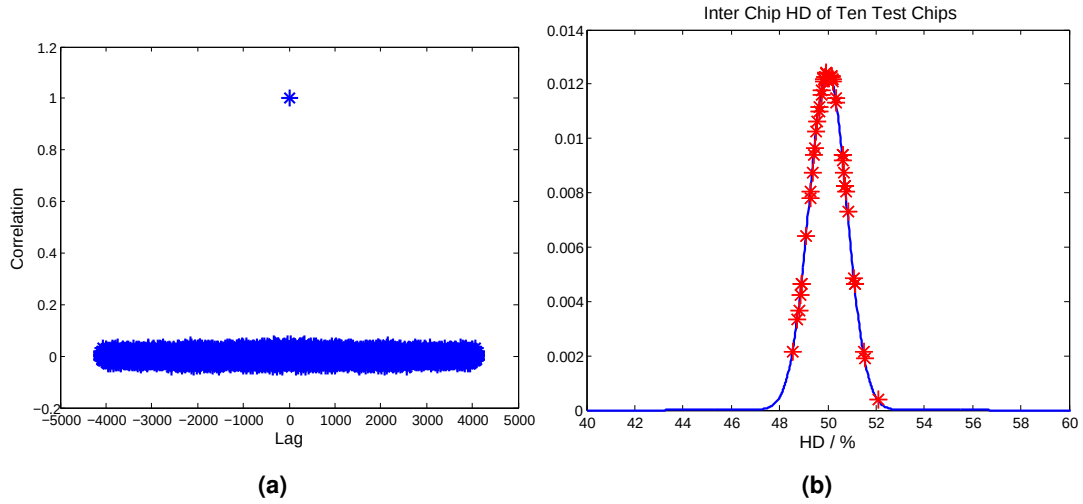


Figure 12.6.: Two stage PUF; (a) Correlation between bits; (b) Correlation between chips.

Correlation Between Chips

The correlation between the chips (inter-chip Hamming distance) is shown in Figure 12.6b . All values are laying near 50% which is the desired result.

Power and Energy Consumption

The current consumption of the test chip was 290 µA. At a voltage of 1.35 V this results in a power consumption of 391.5 µW. By using a clock frequency of 1 MHz we get a power consumption of 391.5 pJ per cell. During the design of the chip, no focus was put on energy consumption optimization. Therefore, the power consumption can easily be reduced.

12.3. Summary

An overview of the results of the two-stage PUF is given in Table 12.1.

Table 12.1.: Properties of the two-stage PUF

Property	Identifier	Test chip
Mean value	$\bar{x}$	0.51
Error rate	$\mathrm{HD}_{\mathrm{intra}(120°\mathrm{C})}$	<5 %
Corr. between bits	R_{xx}	≈ 0
Power consumption	$\frac{\mathrm{E}}{\mathrm{bit}}$	391.5 $\frac{\mathrm{pJ}}{\mathrm{bit}}$

As a conclusion, the test chip shows a good overall behavior. The results we received from the measurements show that temperature shifts are the crucial source of error. Since the temperature problem is a inherent problem of the involved transistors, no big error rate improvements can be expected from re-designs. In contrast, improvements should easily be possible with respect to size and power/energy consumption. An area-reduced version of the chip is shown in the next chapter.

13. PUF With Shared Sense Amplifier

by Christoph Boehm

In this chapter a two-stage PUF with shared sense amplifier will be presented that was developed by the author. The chip was produced in a 90 nm technology. The concept is a modification of the chip presented in chapter 12. To save area the sense amplifier is shared by 16 cells. For the same reasons no common centroid layout is used. The chip consists of 2048 PUF cells. Therefore, 128 sense amplifiers are included in one chip. The clock frequency is 1 MHz. A photograph of the chip is shown in Figure 13.1. The cell array of the PUF cells lies inside the green box. The size is approximately 440 µm x 250 µm. Thus, one cell has a size of 6630 F^2.

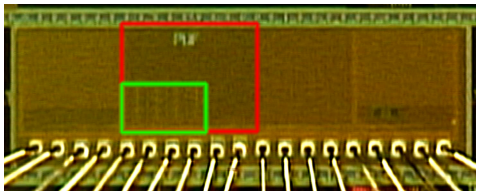


Figure 13.1.: Photograph of the test chip.

13.1. Circuit

The circuit of the approach is shown in Figure 13.2. As already mentioned, the circuit is a simplified version of the approach of Chapter 12. A shared sense amplifier helps to save area. The idea of this concept is to separate the mismatching components from those components which should have minimal mismatch. However, even if the sense amplifier is designed for minimum mismatch, mismatch may occur and have influence on the decision of the PUF cell. If a sense amplifier is shared, it happens that the mismatch of the sense amplifier influences the output of all the involved PUF cells. To reduce the risk of such kind of biasing, only a small number of PUF cells (in this case 16) shares one sense amplifier.

The following text describes the circuit: the transistors $P_{1.n}$ (n=1..N) and $P_{2.n}$ are the mismatch transistors. They work as current sources controlled by the gate voltage U_{BIAS}. The transistors $P_{3.n}$ and $P_{4.n}$ are used to connect the transistors to the sense amplifier and also as cascode transistors for the current sources. The transistors P_5 and P_6 are used to select the hole block.

N_1, N_2, N_3 and N_4 build up the positive feedback decision circuit which is used during the amplification phase. The transistors N_5 and N_6 work as switches. N_7 and N_8 are additional transistors to equalize the voltage drop at N_5 and N_6. During the evaluation phase, the transistors N_3 and N_4 in combination with the current sources build up a latch. If the cell is not selected, the transistors N_9 - N_{11} are switched on and the node in between the NMOS transistors is forced to V_{SS}. This step is necessary to avoid correlation between subsequent bits of the same sense amplifier.

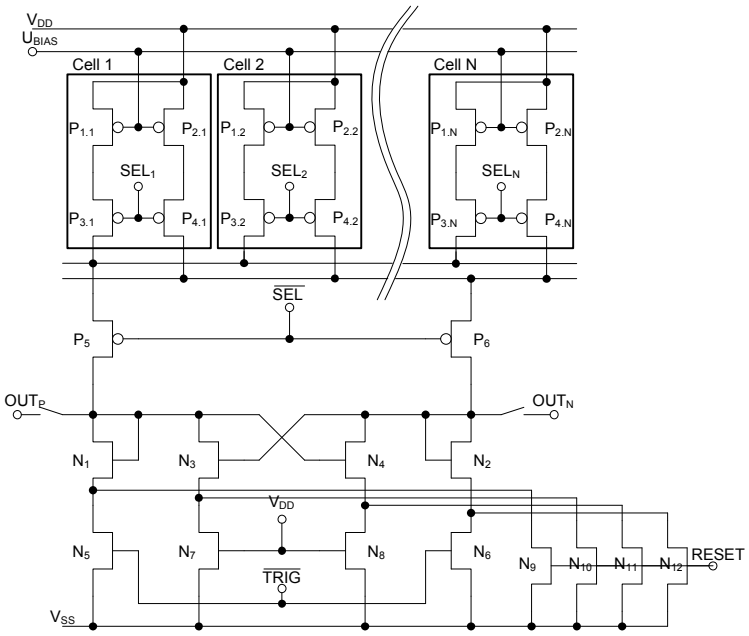


Figure 13.2.: PUF circuit with shared sense amplifier.

13.2. Measurement Results

10 chips were packaged and tested in a temperature range from -40°C to 120°C. The bias current was varied between 2 µA and 30 µA (nom. 16 µA) and the supply voltage was set to 1.25 V, 1.35 V and 1.45 V. Each chip was read out up to 1000 times at each configuration.

The results of the chip specification are shown in the following section. The specification parameters are introduced in Chapter 4.

Mean Value

The mean values of the different chips are shown in Figure 13.3.

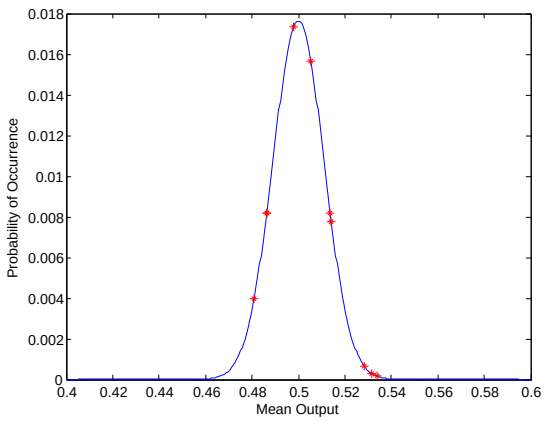


Figure 13.3.: Mean values of the different chips (PUF with shared sense amplifier).

Not all chips are within the confidential interval (CI) of 95 % (0.478 and 0.522). Three chips are outside the CI. Even if this could occur accidently, it is very likely that a systematic bias appears.

The mean value of all chips is $\overline{x} = 0.51$.

Error Rate

The bit error rate (intra-chip HD) of the chip is shown in Figure 13.4a. In Figure 13.4b its temperature dependence can be seen. As in the first example, the error rate increases rapidly as soon as the temperature changes. The value increases to more than five percent for the corner temperatures compared to the reference vector (20°C).

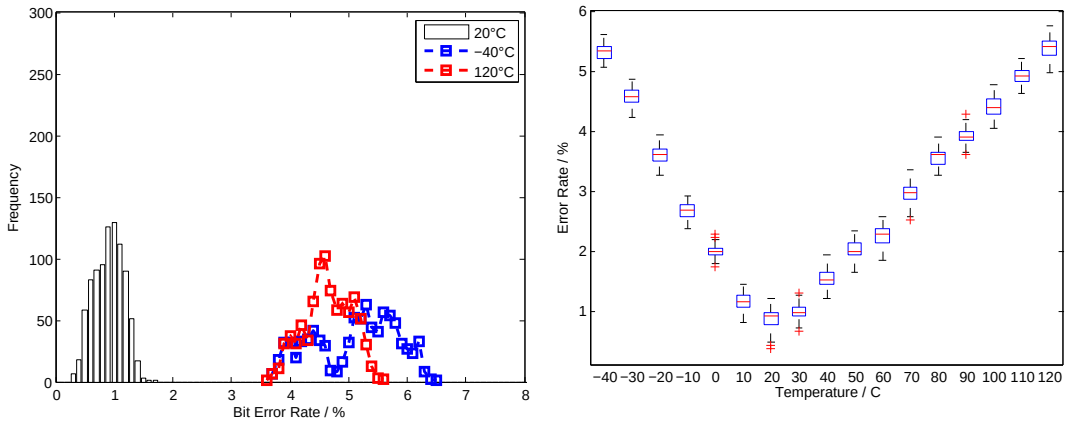


Figure 13.4.: PUF with shared sense amplifier: (a) Intra-chip Hamming distance HD_{intra} (100 runs); (b) HD_{intra} at different temperatures.

Correlation Between Bits

The correlation analysis is of special interest, since the influence of the sense amplifier on the output can be measured. Due to the fact that 16 cells share one evaluation circuit, correlations could occur within groups of 16 cells. To determine the correlation coefficients equation 4.8 is used. The result is shown in Figure 13.5a. No significant correlation can be seen between the bits. A more detailed analysis of the influence of the sense amplifier on the output is given in Section 13.2.

Correlation Between Chips

The correlation between the chips (inter-chip HD) is shown in Figure 13.5b. Since the mean value of the bits is biased towards '1', outputs including more ones than zeros are more likely. Therefore, some of the bit combinations are more likely to occur and HD_{inter} is slightly biased towards zero. The mean value of 255 chips is 49.55 % (ideal: 50 %).

Power and Energy Consumption

Due to the fact that there were two test chips in one package which all were supplied at the same time, it was not possible to measure the current consumption for the shared sense amplifier PUF alone. In simulation, the average current consumption per cell was 4 µA. At a voltage of 1.35 V this results in a power consumption of 5.4 µW. By using a clock frequency of 1 MHz the energy consumption gets 5.4 pJ per cell.

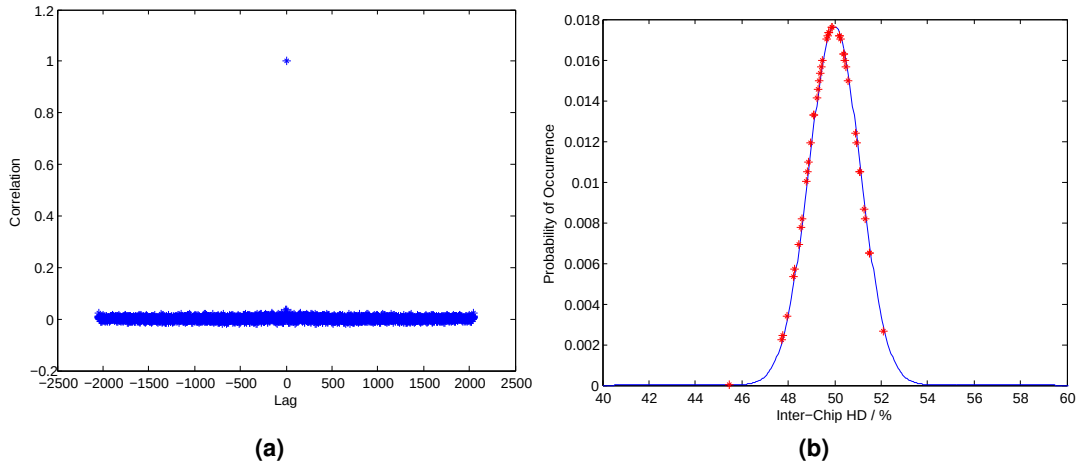


Figure 13.5.: PUF with shared sense amplifier: (a) Correlation between bits; (b) Correlation between chips.

Summary

An overview of the results is given in table 13.1.

Table 13.1.: Summary of the Chip with shared sense amplifier.

Property	Identifier	Test chip
Mean value	$\bar{x}$	0.5
Error rate	$\mathrm{HD}_{\mathrm{intra}(120°\mathrm{C})}$	0 %
Corr. between bits	R_{xx}	≈ 0
Corr. between chips	$\mathrm{HD}_{\mathrm{inter}}$	50 %
Power consumption	$\frac{\mathrm{E}}{\mathrm{bit}}$	5.4 $\frac{\mathrm{pJ}}{\mathrm{bit}}$ [a]

[a] Value from simulation.

Output Bias Due to Sense-Amplifier Bias

In the case of the presented PUF additional analysis of the correlations between the cells connected to one sense amplifier have to be carried out. As explained above, the test chip includes sense amplifier (SA) sharing. This technique will be of interest as soon as the sense amplifier is large compared to the part of the PUF cell which provides the mismatch. Since the sense amplifier should not provide mismatch to the circuit it has to be designed large. Normally, it is convenient to share the SA between a number of PUF cells. In the case of the test chip, each sense amplifier is connected to 16 mismatch cells.

If sense amplifier sharing is used, one focus of the design should cover the optimization of that part of the circuit, since influences of the SAs on the decision of the cells may lead to the situation that all the 16 cells put out the same value in a worst case scenario. This is a severe security issue, since a possible attacker can guess the output easier.

In Figure 13.6 three situations are depicted: the distribution of the number of ones in a block of 16 bits is shown. If the sense amplifier does not influence the decision of the cell and if the cells' output mean is 0.5, the output distribution is the binomial distribution (*blue dots*). If the sense

amplifier introduces a bias on the decision process, the distribution deviates from the binomial distribution and flattens. If the influence of the SA is small, the output distribution looks like the *red* distribution in Figure 13.6. Here, still the mismatch cells dominate the decision. If the influence increases and the mismatch of the SA starts to dominate, the output distribution looks like the one shown by the *black* curve. There are many blocks in which only zeros or only ones show up at the output. This has to be prevented.

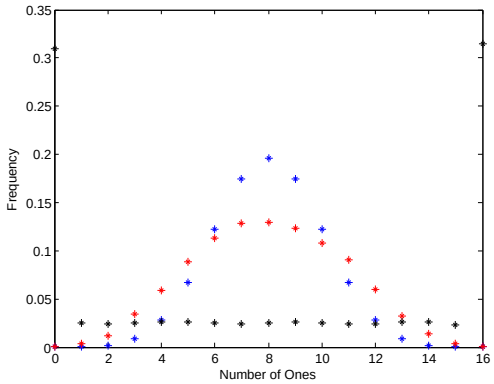


Figure 13.6.: Ideal (blue) distribution of ones in a block of 16 output bits. Small (red) and strong (black) influence of SA mismatch.

This type of plot in Figure 13.6 can also be used to analyze the behavior of the test chip. To do so, the output of the ten packaged chips is grouped in 16-cell blocks, where 16 is the number of cells that share one sense amplifier. Since one chip provides 2048 bits of output, the ten chips provide in total 265*10=2650 blocks. The result is shown in Figure 13.7. To see if the test chip provides values which are not expected by the binomial distribution, Figure 13.7 also shows the 3σ curves (black) for an ideal binomial distribution of 2650 data blocks. 99,73 % of the results should occur within that area.

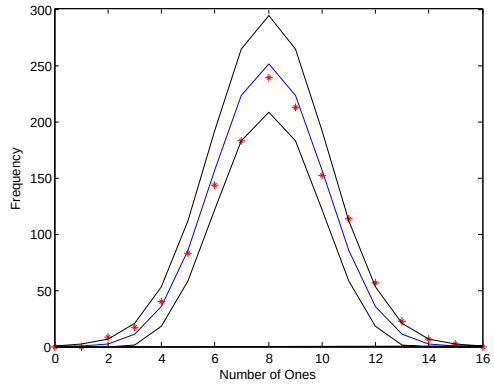


Figure 13.7.: Ideal (blue) distribution of ones in a block of 16 output bits. The red curve shows the results from the test ships. The black curves mark the 3σ (99,73 %) confidential interval.

Two effects can be observed in Figure 13.7. Firstly, the red curve shows slightly the same behavior as the red curve in Figure 13.6. This indicates a small influence of the SA on the decision. Secondly, the curve tends to the right side. That shows a gradient on the decision which produces

more ones than zeros. This behavior could already be observed in Section 13.2. If the non-ideal behavior is acceptable, has to be decided from case to case. The bias towards '1' can be reduced by post-processing.

To see the difference between a chip with sense amplifier sharing (and without common centroid layout) and a chip without sense amplifier sharing (and with common centroid layout to avoid any form of gradient at the output), Figure 13.8 shows the results of the two-stage PUF presented in Chapter 12. In this case, the number of bits per chip is 4096. Thus, for ten chips the total number of 16 bit blocks doubles and sums up to $10 * 512 = 5120$ blocks. This number is also used for the evaluation of the 3σ curves.

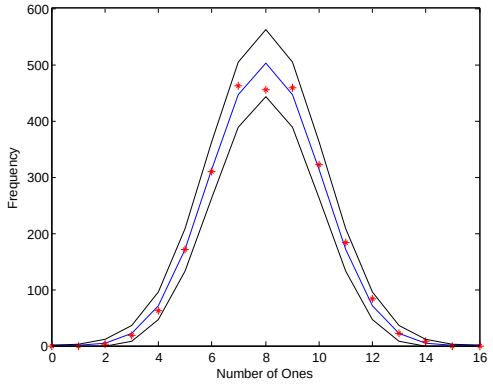


Figure 13.8.: Distribution of ones in a block of 16: First test chip (two-stage PUF).

Figure 13.8 shows that all results occur within the 3σ region and even within the 2σ region which is not shown in this figure. As expected, no influence of a shared SA can be observed and also no gradient on the output is visible. If no post-processing is provided, common centroid layout and no SA sharing should be used for high security applications.

To see the effect of common centroid layout on the gradient Figure 13.9 shows the analysis results of the second block (pre-selection) of the test chip (see Chapter 14). If the pre-selection is deactivated, the output shows the nominal distribution of ones and zeros. Again, a block of 16bit is used. This time the output of 10 chips is used each providing 2048 bits. Most of the

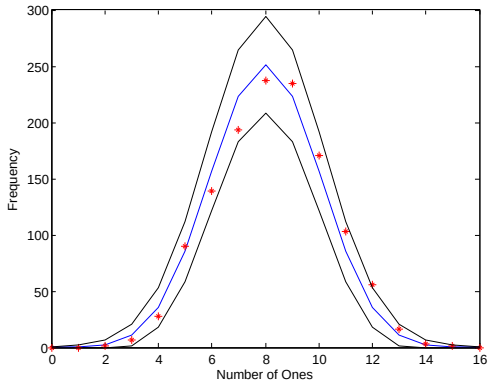


Figure 13.9.: Distribution from second block of second test chip.

points are within the 3σ area which can be expected, since this PUF does not use sense amplifier sharing. Nevertheless, a bias towards the right side (more ones than zeros) can be observed. This effect is most likely caused by the a V_{TH} gradient on the chip. Since no common centroid layout

was applied, this gradient gets visible in the results. Again, if unbiased output is strongly desired, common centroid layout should be used. If the biased output can be compensated for by post-processing, common centroid layout can be omitted in exchange for smaller area consumption.

13.3. Pre-Selection Multiple Readout

By using the presented PUF the multiple readout pre-selection approach which was described in Chapter 10.1 was realized. Due to the fact that no additional hardware is necessary, this kind of pre-selection can be implemented easily in combination with any PUF chip. To evaluate the performance of the approach four different configurations were tested:

- 1 temperature, 10 readouts each: 20°C.
- 1 temperature, 50 readouts each: 20°C.
- 3 temperatures,10 readouts each: -40°C, 20°C and 120°C.
- 3 temperatures,50 readouts each: -40°C, 20°C and 120°C.

Certainly, there are many other reasonable combinations. An initialization phase is used to pre-select the stable cells. The initialization phase has to be done before the PUF is used for the first time. The pre-selection phase can be done during the chip testing procedure, for example. The choice of the temperatures depends on the future temperature range of the product and is limited by the tester capabilities. The number of readouts depends on the available tester time. Generally, testing time is expensive and a trade-off has to be made.

In the case of the introduced PUF the efficiency (selection rate) after pre-selection at constant temperature is 97 %, the efficiency after pre-selection at three different temperatures is 81 %. The advantage of pre-selection at different temperatures can be seen by looking at the error rate with respect to temperature. Figure 13.10a (10 readouts) and Figure 13.10b (50 readouts) show the BER depending on the temperature after pre-selection at constant temperature. The results for pre-selection at three different temperatures can be seen in Figure 13.10c (10 readouts) and Figure 13.10d (50 readouts).

As expected, the multiple readout at constant temperature does not improve the error rate significantly. Only the noise induced errors can be reduced. An overview of the BER of the different settings is given in Table 13.2. There is still a worst case BER of 5.54 % (4.64 %) for the 10 (50)

Table 13.2.: BER of the different realizations.

Setting	Readout	Temperature	Efficiency	$BER_{20°C}$	BER_{MAX}
a	10	20°C	98.68 %	1.14 %	5.54 %
b	50	20°C	96.83 %	0.35 %	4.64 %
c	10	-40°C, 20°C, 120°C	88.13 %	0.33 %	0.61 %
d	50	-40°C, 20°C, 120°C	86.08 %	0.06 %	0.28 %

readouts case at the single temperature pre-selection. If the chip is meant to be used at different temperatures (which can be assumed in general), the initialization has to be done at the corners of the desired temperature range to minimize the BER. By comparing the results in Figure 13.10a and Figure 13.10b it can be seen that the multiple readout at 20°C does not reduce the error rate at -40°C and 120°C significantly. This effect indicates that the PUF cells change their stability when temperature changes as described in Chapter 7. If the pre-selection is done at multiple temperatures as shown in Figure 13.10c and Figure 13.10d, a significant improvement in terms of BER

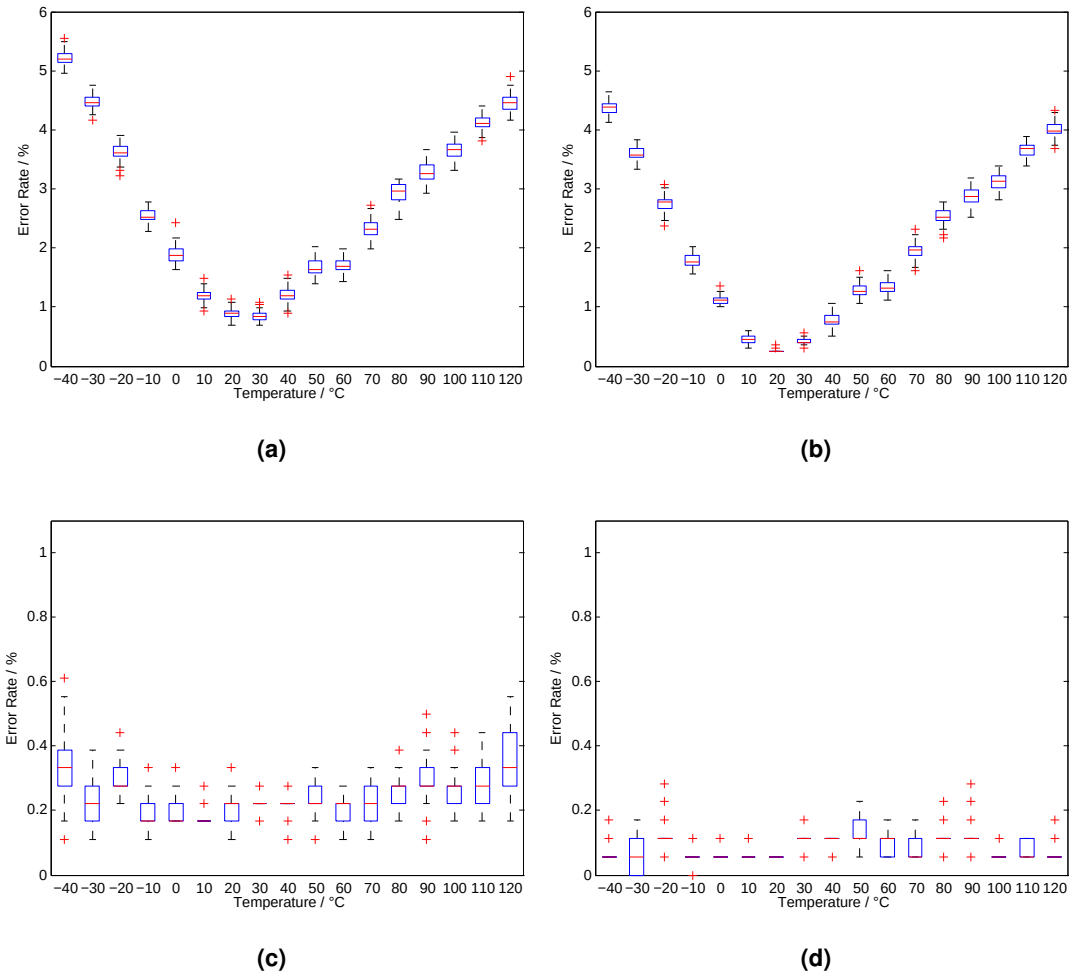


Figure 13.10.: BER of pre-selection multiple readout in dependence of the temperature (a) 1 temperature, 10 readouts; (b) 1 temperature, 50 readouts; (c) 3 temperature, 10 readouts (d) 3 temperature, 50 readouts.

can be observed. In this case, also the improvement caused by increasing the number of readouts gets visible. If the number is increased from 10 to 50, the BER reduces by nearly 50 % which also means a reduction of BER of more then twice.

The mean value, the correlation between bits, and the correlation between chips are not affected by the pre-selection process and remain quite constant compared to the results presented earlier in the chapter. Due to the reduced efficiency and the serial readout the energy consumption is increased to 5.47 pJ, 5.58 pJ, 6.14 pJ and 6.27 pJ per bit depending on the kind of readout.

After the pre-selection using multiple readouts, the BER still remains too high to be used as a key for cryptographic purposes. Nevertheless, this kind of pre-selection could be a good approach to reduce the BER in a way to allow for low-complexity error correction for the post-processing of the output data. Furthermore, this pre-selection approach could be combined with one of the other pre-processing approaches to further reduce the error rate.

13.4. Summary

The shared sense amplifier of the presented chip helps to reduce the area of the chip significantly. The number of transistors in a single PUF cell can be reduced four transistors. Nevertheless, the design of such a PUF has to be done carefully to prevent a bias on the output caused by mismatches inside the sense amplifier. This can be realized by either increasing the size of the sense amplifier or by decreasing the number of cells that share one SA. As shown in Section 13.3, the multiple readout pre-selection approach is only an efficient way to reduce the error rate if it is done at different corner temperatures. The readouts have to be done at T_{MIN} and T_{MAT} of the chip's desired temperature range to get a significant reduction in BER over the whole region of operation.

14. PUF With Pre-Selection

by Maximilian Hofer

In the following Chapter a PUF including pre-selection circuitry will be presented. As in the circuits of Chapter 12 and Chapter 13 the cell readout is also done in two phases. Additionally, each cell includes required control circuitry for the pre-selection process [62]. The chip was produced in a 90 nm technology. Each chip consists of 2048 PUF cells. The cells are read out serially at a frequency of 1 MHz. A photograph of the chip is shown in Figure 14.1. The area including the pre-selection PUF is surrounded by a red line.

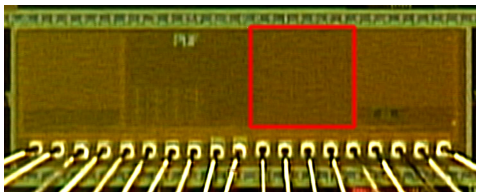


Figure 14.1.: Photograph of the test chip.

The area consumption of the PUF is approximately 570 µm x 570 µm. The size of one cell is 19600 F^2. Since the chip was not optimized with respect to the size, the area consumption is rather high. For mass production, the size could be decreased in different ways: smaller transistors, shared sense amplifier, only one level of pre-selection, optimized layout, etc. Some of them are explained later in Section 14.4. To make it comparable to other approaches the cell is first analyzed without the pre-selection feature followed by a complete analysis.

14.1. Circuit

The circuit of the cell is depicted in Figure 14.2. To make the circuit more readable the pre-selection transistors are grayed out.

The transistors P_3 and P_4 are used as switches for reseting the circuit. If the two transistors conduct, the nodes A and B are forced to V_{DD}. The transistors N_1 and N_2 are the diode connected mismatch transistors. To maximize the mismatch the size has to be minimized. N_7 works as current source and limits the current I_{MAX} through the circuit. N_5 and N_6 are the selection transistors and work as switches. If these two transistors are conductive, the cell is activated. Due to the mismatch between the transistors N_1 and N_2 the nodes A and B settle at different voltages. If the trigger signal is set, N_8 is switched on. The circuit built by the transistors P_1 and P_2, and N_3 and N_4 is a bistable circuit. Depending on the mismatches, one of the nodes moves towards V_{DD}. The other node moves towards V_{SS}. Subsequently, the transistors N_9 and N_{10} connect the nodes A and B to each of the outputs respectively and the result can be read out. A timing diagram of the different control signals is shown in Figure 14.3.

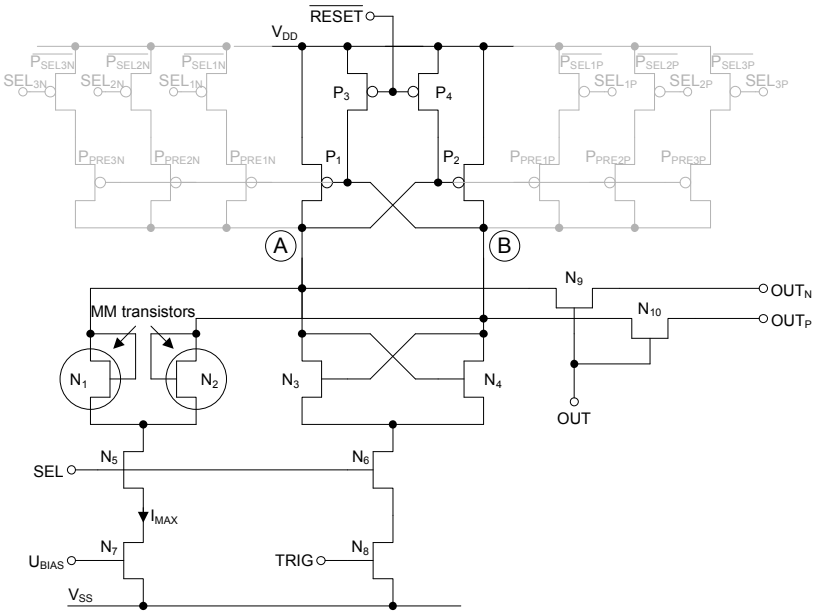


Figure 14.2.: Circuit of test chip 3.

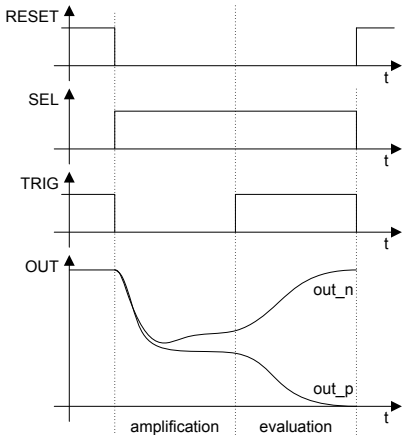


Figure 14.3.: Timing diagram of the different control signals.

14.2. Measurement Results Without Pre-Selection

Ten chips were tested in a temperature range between -40°C and 120°C, with bias currents between 2 µA and 30 µA (nom. 16 µA) and supply voltages of 1.25 V, 1.35 V and 1.45 V. Up to 1000 readouts where done for each configuration.

The chip was specified using the parameters listed in Chapter 4.

Mean Value

The mean values of the different chips are depicted in Figure 14.4. The mean value of all chips is $\bar{x} = 0.497$.

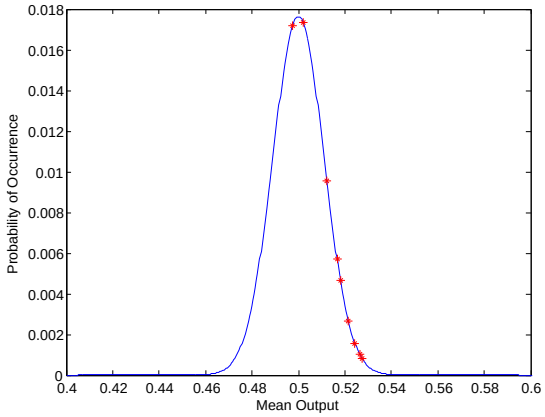


Figure 14.4.: Mean output values of 10 packaged test chips.

A clear bias towards '1' can be observed. This bias can be removed by post-processing steps for high security application. For identification purposes, a small bias does not effect the functionality of the PUF much.

Error Rate

The error rate (intra-chip HD) at the nominal temperature (25°C) and at the temperature corners (-40°C, 120°C) is illustrated in Figure 14.5a. The BER over the whole temperature range is depicted in Figure 14.5. The maximal error rate shows up at 120°C. At this temperature the BER is $\mathrm{HD}_{\mathrm{intra}(120°\mathrm{C})}$=4.43%.

Correlation Between Bits

There is no significant correlation between the different bits. Figure 14.6a shows the analysis results.

Correlation Between Chips

There is little correlation between the chips (inter-chip HD). The inter-chip Hamming distances between different chips is depicted in Figure 14.6b. The mean value of all chips is $\mathrm{HD}_{\mathrm{intra}} = 49.94\%$. The small correlation is caused by the bias towards '1' at the output of the chip.

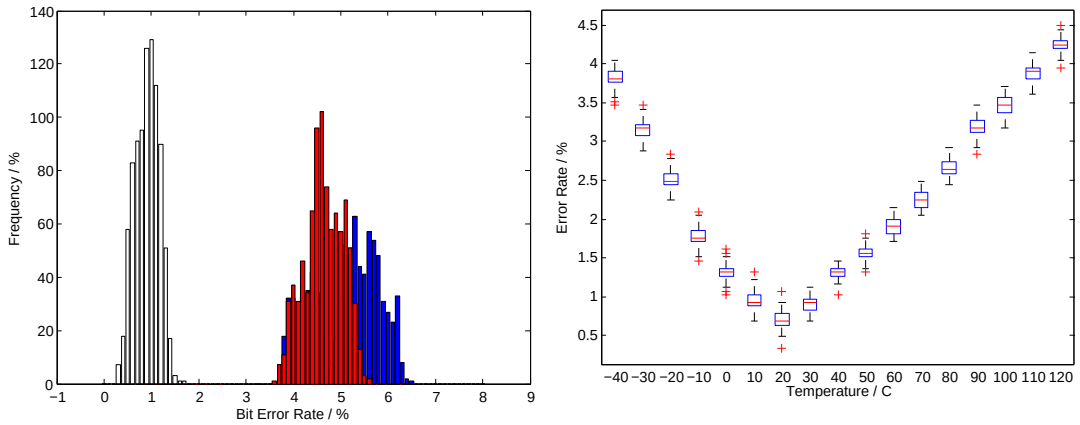


Figure 14.5.: PUF test chip 3: (a) Intra-chip Hamming distance HD_{intra} (100 runs); (b) HD_{intra} at different temperatures.

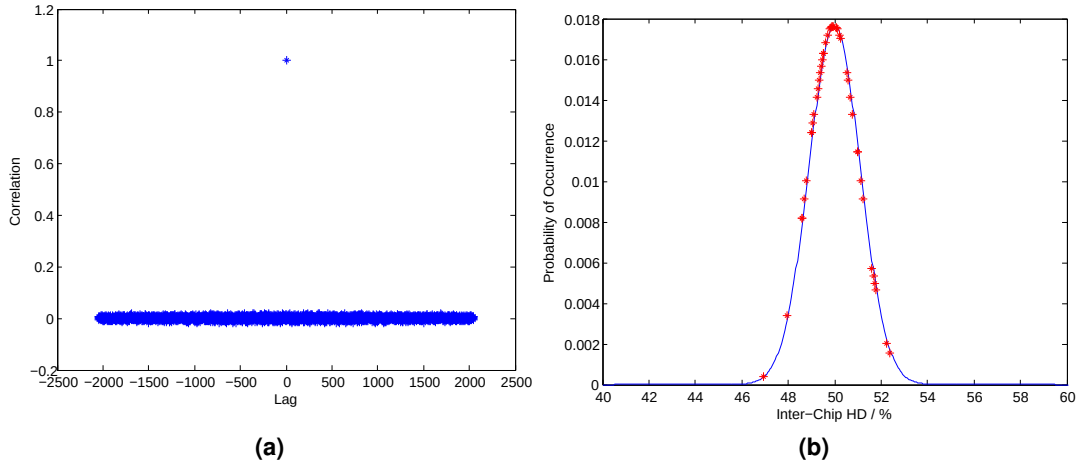


Figure 14.6.: PUF chip 3; (a) Correlation between bits; (b) Correlation between chips.

Power and Energy Consumption

Due to the fact that there were two different PUFs in one package and both used the same power supply, it is not possible to measure the current consumption of the PUF cell exactly. Circuit simulations have shown that the average current consumption per cell is 4 µA. At a supply voltage of 1.35 V the power consumption results in 5.4 µW. At a clock frequency of 1 MHz the energy consumption gets 5.4 pJ per cell.

Summary

An overview of the results is given in Table 14.1.

The presented chip without using the pre-selection functionality already shows a good error behavior. All data in Table 14.1 are comparable to the results from the other test chips. Again, the biggest problem seems to be the temperature behavior of the PUF cells and thus the error rate over temperature. To get this problem under control the pre-selection functionality has to be used. The

Table 14.1.: Summary of the chip without pre-selection.

Property	Identifier	Test Chip
Mean value	$\bar{x}$	0.497
Error rate	$HD_{intra(120°C)}$	4.43 %
Corr. between bits	R_{xx}	≈ 0
Corr. between chips	HD_{inter}	49.94 %
Power consumption	E/bit	5.4 $\frac{pJ}{bit}$ [a]

[a] Value from simulation.

results are shown in the next section.

14.3. Measurement Results with Pre-Selection Enabled

The circuit depicted in Figure 14.2 also includes a pre-selection block. Some components were added to implement the approach described in Chapter 10.2 (pre-selection delta). For evaluation purposes, 7 different levels of pre-selection bias can be added to each branch of the PUF. The selection steps has been realized using a binary weighted network. The pre-selection level can be controlled over five input pins at the test chip. The pins are listed in Table 14.2. The pin *psen* must be high to activate the pre-selection. The branch is selected with *psdir*. *psa* defines the actual level. If *psa* = 000 (level 0), no pre-selection bias is added. If *psa* = 111 (level 7), the highest amount of pre-selection bias is added. Thus, 7 pre-selection steps are available.

Table 14.2.: Different Pins to control the pre-selection on the test chip.

pin	Explanation
psen	Pre-selection enable
psdir	Pre-direction selection
psa	
psa0	Pre-selection adress 0 (LSB)
psa1	Pre-selection adress 1
psa2	Pre-selection adress 2 (MSB)

The pins are only used during the initial phase. Afterwards, during the evaluation phase the pre-selection circuit is disabled (*psen*=0).

In Figure 14.7 the PUF cell including the pre-selection circuit is depicted. The transistors P_{PRE1N}, P_{PRE2N} and P_{PRE3N} are pre-selection transistors and used to provide pre-selection bias to the left branch (A). The pre-selection transistors are switched on and off with the transistors P_{SEL1N}, P_{SEL2N} and P_{SEL3N}. At the right branch (B), P_{PRE1P}, P_{PRE2P} and P_{PRE3P} are the pre-selection transistors and P_{SEL1P}, P_{SEL2P} as well as P_{SEL3P} are used to switch them on and off. The current of the pre-selection transistors is binary weighted. The size of the pre-selection transistors is listed in Table .

The the size of the transistors P_1 and P_2 is chosen to be $\frac{500}{90}$. If the pre-selection transistors are switched by the transistors $P_{SEL(1..3)N}$ or $P_{SEL(1..3)P}$, the mean current through the particular branch is increased depending on the selected pre-selection level. This biasing causes a shift in the mean output value of all PUF cells. In Figure 14.8a this effect is shown. Figure 14.8b shows

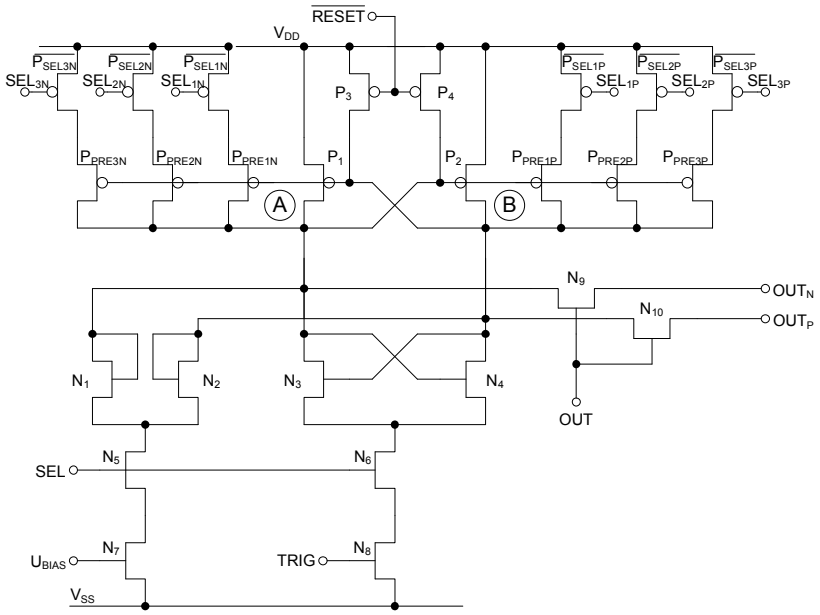


Figure 14.7.: PUF cell with pre-selection circuitry.

Table 14.3.: Size of the pre-selection transistors.

Transistor	Size $\frac{W}{L}$
P_{SEL1N}, P_{SEL1P}	$\frac{160}{400}$
P_{SEL2N}, P_{SEL2P}	$\frac{160}{180}$
P_{SEL3N}, P_{SEL3P}	$\frac{160}{90}$

the mean output of all available PUF cells in dependence of the pre-selection level. At level 1 only the transistor P_{SEL1N} (P_{SEL1P}) is switched on. At the highest level, level 7, all 3 pre-selection transistors of one of the branches are switched on. The higher the pre-selection level, the higher the shift in the mean output value.

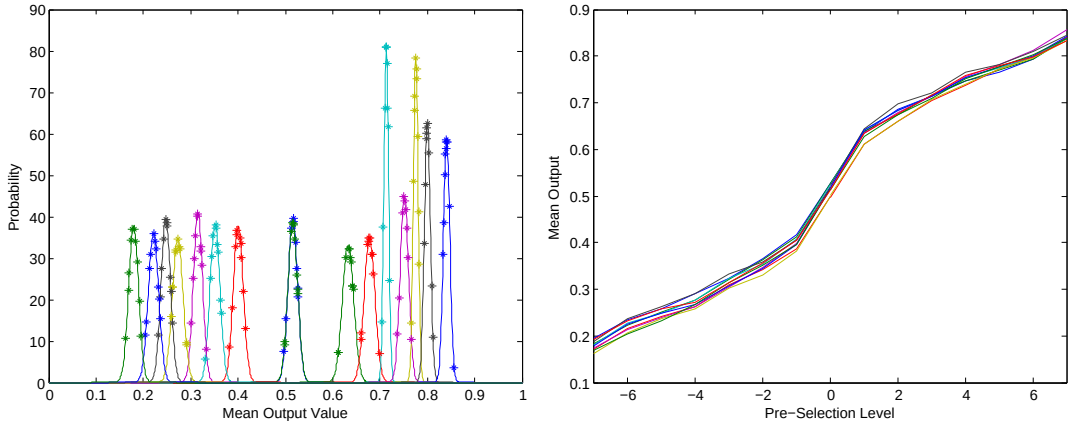


Figure 14.8.: Mean of the test chip with pre-selection: (a) Distribution of mean output at different pre-selection levels; (b) Mean output in dependence of the pre-selection level.

In Figure 14.9a the relationship between the pre-selection level and the number of selected PUF cells (efficiency) is shown. At pre-selection level of 7 the efficiency is still e$\approx \frac{1}{3}$. Figure 14.9b shows the correlation between the efficiency and the measured BER at the worst case temperatures and nominal conditions (-40°C , 20°C , 120°C). The lower the efficiency, the lower the bit error rate. The curves converge towards 0.

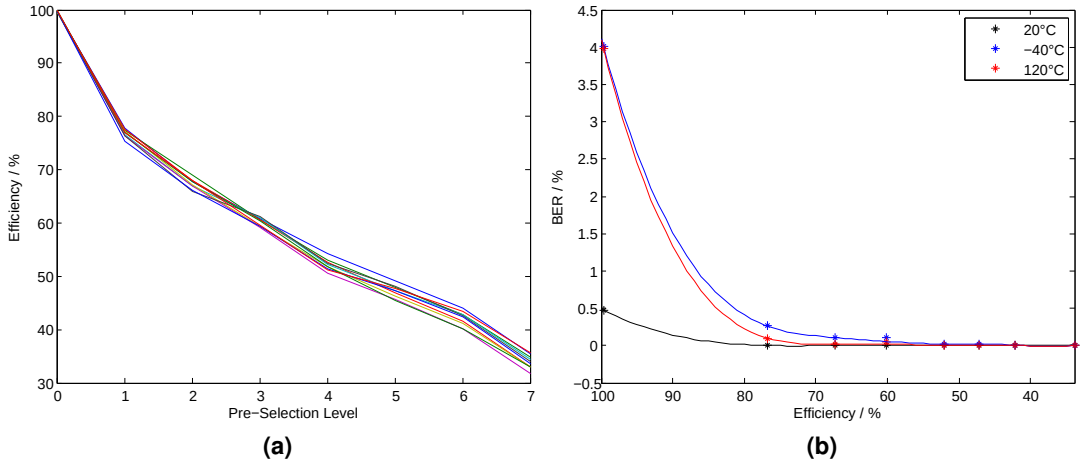


Figure 14.9.: Effects of pre-selection on test chips: (a) Level vs. efficiency; (b) Efficiency vs. BER at different temperatures.

Only at low pre-selection levels and at corner temperatures (-40°C and 120°C) a considerable number of errors occur. During all measurements at pre-selection level 2 no error occurred at 20°C. At -40°C and 120°C there where no errors at level 6 and 7. In Table 14.4 the BER is listed numerically. There are almost the same error rates at -40°C and 120°C without any pre-

selection. Measurements with pre-selection show a significant higher number of errors at higher temperatures. This is assumed to be caused by an increasing thermal noise, if temperature rises.

Table 14.4.: BER and efficiency, at 20°C, -40°C and 120°C.

Level	0	1	2	3	4	5	6	7
Efficiency	99,7 %	76,8 %	67,5 %	60,2 %	52,1 %	47,2 %	42,1 %	33,9 %
$\overline{BER_{20°C}}$	0,469 %	0,0002 %	0,0002 %	0,0001 %	0 %	0 %	0 %	0 %
$\overline{BER_{-40°C}}$	3,973 %	0,0979 %	0,0209 %	0,0462 %	0,0094 %	0,0105 %	0 %	0 %
$\overline{BER_{120°C}}$	4,004 %	0,2635 %	0,1084 %	0,1031 %	0,0232 %	0,0242 %	0,0056 %	0 %

14.4. Possible Improvements

The pre-selection circuit introduced in this chapter is convenient to illustrate the functionality of the pre-selection delta approach. The different pre-selection levels make a detailed analysis possible. The design focus has to be changed for mass production. Here, the size of the circuit plays an essential role and is one important parameter used to reduce production costs. Therefore, the circuit has to be optimized in terms of area consumption. Some possible improvements are suggested in the following part. One of the most profitable circuit improvements is to use a shared sense amplifier. A large portion of the area of the existing cell is occupied by the evaluation and pre-selection circuit. Thus, sense amplifier sharing can be used to group several PUF cells. As shown in Chapter 13 sense amplifier sharing is a good approach as long as design and layout is done with special care. As another step, it is not necessary to implement different levels of pre-selection. It is sufficient to include just one level of pre-selection on each side for the use in a high volume product. The level has to be defined in advance by using simulation and test chip measurement results. It is important to remember that the level changes between different technologies.

A circuit that includes sense amplifier sharing and also just one pre-selection level is shown in Figure 14.10.

The transistors $N_{1.n}$ and $N_{2.n}$ (n = 1...N) are the mismatch transistors. The transistors $N_{5:n}$ are working as switches. They are used to connect the different cells to the current source.

The transistor P_1, P_2, N_3 and N_4 are building up the sense amplifier as explained above. Transistor P_{PREN} and P_{PREP} are the pre-selection transistors and the transistors P_{SELN} and P_{SELN} are working as switches to connect them to the sense amplifier. It is important to make sure to have maximum mismatch between the transistors in the cells and minimum mismatch between the transistors in the sense amplifier to avoid correlations between the output bits. Hence, the cells have to be designed small and the sense amplifier with pre-selection circuit must have a certain size to keep its influence on the output as small as possible. One PUF cell consists of three transistors only. That is why very small cell sizes can be reached. The sense amplifier and the switches have to be designed very carefully in terms of mismatch, noise and leakage.

14.5. Summary

Pre-selection is a good approach to reduce the error rate of the PUF. In addition to the testchip, an optimized version was suggested. The hardware afford can be kept small when using sense amplifier sharing. This makes pre-selection a cheap alternative to complex error correction codes.

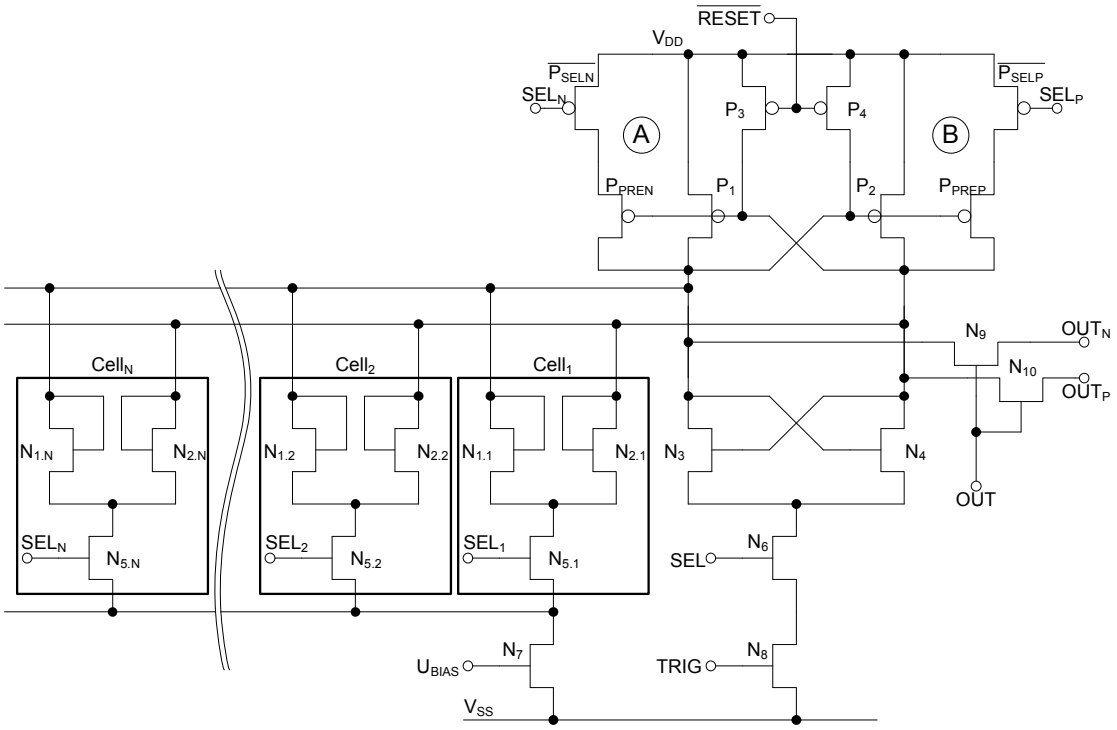


Figure 14.10.: Circuit with pre-selection and shared sense amplifier.

15. A Microcontroller SRAM PUF

by Maximilian Hofer

Microcontrollers are a crucial part in modern electronic devices. Microcontrollers are wildly used in all kinds of applications from automotive to communication systems.

Amongst others, microcontroller are used in systems in which data integrity is required. Therefore, it has to be made sure that the data is secure even in hostile environments. Especially, if the data is transmitted over public channels like radio links, the data has to be encrypted to prevent adverse access. If data encryption is required, key material is needed that must not be visible to the outside. Thus, the key material has also to be stored in a secure way.

Beside secure communications, software cloning is another problem that microcontrollers may have to cope with. The microcontroller software is commonly stored in some non-volatile memory. Such memory include flash memory, in mask programmed ROMs, or external memory chips. Unfortunately, NVMs can be read out easily by adversaries especially if the memory is placed outside the microcontroller. Thus, if there is not be taken extra care, microcontroller system software can be copied easily and the whole system can be cloned. One way to prevent cloning is to encrypt the software data before storing, instead of storing the plain text inside NVMs. The code is transfered to the system and decrypted on the microcontroller during a start up phase. Thus, the plaintext of the code never leaves the microcontroller. The described approach is shown in Figure 15.1. As in the data encryption case, the key that is needed to encrypt the plain text must not be accessible from the outside. There are ways to store key material directly on the microcontroller in non-volatile SRAMs [142]. Special hardware and a battery is required to allow for such a system.

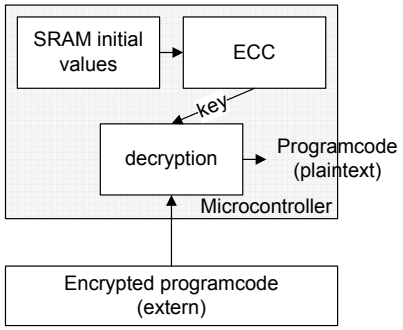


Figure 15.1.: Saving program code by encryption.

Secure key storage is also a strong requirement for other microcontroller applications. In some cases, microcontrollers are used in Pay-TV systems. This means that the encrypted data stream has to be decrypted using keys (see Section 2.2). In such systems at least a master key has to be stored directly at the microcontroller. The master key is then used to decrypt the encryption keys which are sent over a public channel to the pay-TV consumer.

As already mentioned above, different types of non-volatile memory is commonly used to store the data. Even on-chip flash memory can be used to store the key material. Nevertheless, this way of key storage can not be considered as secure. Even if the memory cannot be accessed

directly over external contacts at the microcontroller, techniques exist with which adversaries can access such data. Different approaches to attack secure flash memory is described in [1]. Similar techniques can be used to get access to data that are stored in ROMs. Thus, new ways to store key material on microcontrollers have to be found. One approach to solve this problem is to use the SRAM that is available on the microcontroller. The SRAM can be utilized as SRAM PUF which is used to generate/store the key material.

Nevertheless, it is a disadvantage that SRAM-based PUFs show high error rates. The error rates of SRAM PUFs are commonly higher than those of dedicated PUF circuits. This is due to the fact that the designer of PUF circuits can take extra care to maximize local mismatches and to minimize the influence of noise. To reduce the error rate error correction codes (ECCs) are used. Since SRAM PUFs show high error rates of ten percent and more, not all EECs are feasible. Many ECCs are too complex to be implemented in microcontrollers.

In the following chapter an easy-to-implement SRAM PUF combined with a repetition code to provide a negligible error rate is suggested [18].

15.1. Basic Concept

Microcontrollers (MCs) usually provide both, an SRAM as random access memory and a flash memory as non-volatile memory. Figure 15.2 shows the memory map of a microcontroller. In this case, the microcontroller is chosen to be an NXP LPC 1768.

By using the SRAM and the flash a PUF can easily be implemented in a microcontroller. This PUF can be used to generate keys later. The concept is described as follows: after powering up the microcontroller, the SRAM states are read out. Then the error correction data is generated and stored into the flash. Here, the on-chip NVM is not important. Since the error correction data does not include any information on the PUF output, the data can be stored externally as well. This is certainly not true for the SRAM: the SRAM must be situated inside the microcontroller since otherwise the communication between SRAM and microcontroller could be observed easily. Furthermore, it is important to make sure that the data of the internal SRAM block is not accessible from the outside. This means that no software is allowed to run that helps to provide this kind of data on one of the microcontroller pins. Once the correction data is available, the PUF can be used regularly. The software consists of two parts: the PUF is read out during the first part and during the second part the error correction data is used to remove the errors in the current PUF output.

In Figure 15.2 the concept behind the approach is shown. The initialization bit is read out during the start-up of the microcontroller. If the bit is not set yet, the PUF starts the initialization in which the start up output of the PUF is used to generate the error correction data. After that, the correction data is stored to the NVM and the initialization bit is set. If the initialization data is already available (i.e. the initialization bit is set), the readout procedure starts and the correction data is used to remove the errors from the data.

15.2. Error Correction Using the Repetition Code

As mentioned above, the error rates at SRAM-based PUFs are high due to their sensitivity to noise. Thus, not all error correction codes are feasible. Moreover, not all microcontrollers are capable to handle the complexity of some of the ECCs when it comes to very high error rates. One code that may be used in a situation like this is the so-called repetition code. Although the error correction capability is high, the complexity of the repetition code is very low. Thus, it can be implemented in many microcontrollers easily.

The principle of the repetition code is as follows: a majority decision of a odd number (N) of output bits is used to generate generate one output bit. At the initial phase, one of the bits is defined

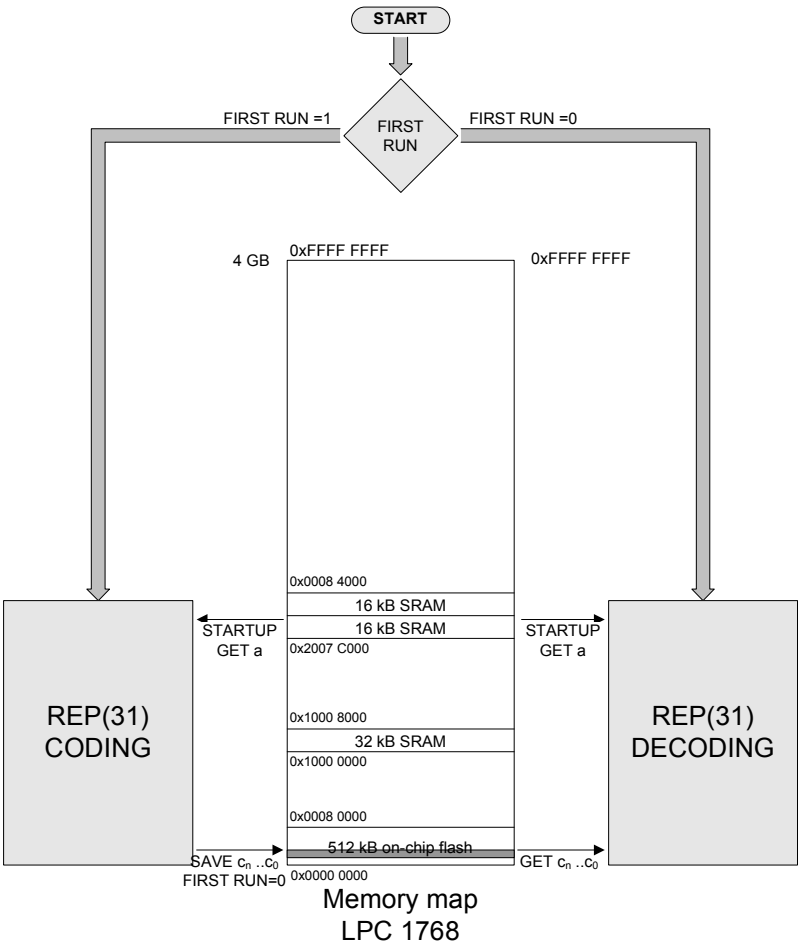


Figure 15.2.: Memory map and concept of the microcontroller SRAM PUF.

to be the nominal bit. To generate the correction data all N bits are XORed with the nominal bit. Example:

power-up values	1001011
7x first bit	1111111
error correction bits	0110100

Later, during nominal PUF readout, the correction data is used to reduce the error rate. To do so, the current PUF output is XORed with the correction data. After that, a majority decision is made. If less than half of the bits are erroneous, the majority decision results in the right value. The principle of a seven bit repetition code (Rep7) can be seen in the table below. The ideal value after XORing the SRAM output with the correction data is presented in the first column. In the second column the erroneous bits are shown. Finally, the outcome of the majority decision is listed in the last column:

$1 : 1111111$	$\rightarrow 1011011$	$\rightarrow 5 \times' 1' and 2 \times' 0' \rightarrow 1$
$0 : 0000000$	$\rightarrow 0110100$	$\rightarrow 3 \times' 1' and 4 \times' 0' \rightarrow 0$
$0 : 0000000$	$\rightarrow 1101010^*$	$\rightarrow 4 \times' 1' and 3 \times' 0' \rightarrow 1$

* too many errors

In the example of the third line too many errors show up. Thus, a '1' instead of a '0' is returned by the majority decision. In Figure 15.3 the two phases of the error correction are depicted. Figure 15.3a shows the initialization phase. In Figure 15.3b the key generation phase is depicted.

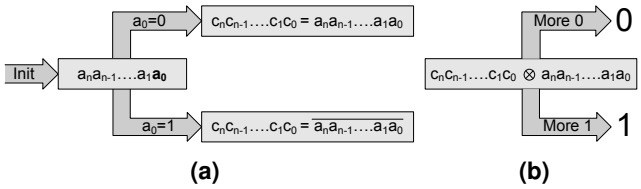


Figure 15.3.: (a) Initialization; (b) Key generation.

In Figure 15.4 the performance of three different repetition code length is shown: 11 bit, 21 bit, and 31 bit. On the x-axis the error probability per bit is assigned. On the y-axis the error probability after the error correction is shown. It can be seen for example, that a repetition code with repetition factor of 31 reduces the initial error rate of 10 % to an error rate (BER) of 6.85E-07 %. The correction capabilities of the repetition factors 21 and 11 can be seen in the BER after correction: 1.35E-04 % and 2.96E-02 % respectively.

15.3. Measurement Results

The concept was implemented in an NXP LPC1768 microcontroller. This microcontroller includes an ARM Cortex M3 core, a 64 kB SRAM, and 512 kB of flash memory. The SRAM is divided into two different blocks: 32 kB of SRAM are used for the stack and the heap memory, and 32 kB are used for the peripheral components, like the Ethernet stack or the USB stack. A memory map can be seen in Figure 15.2.

Both of the blocks were used in the concept implementation and analyzed towards their SRAM PUF feasibility. The lower SRAM block is addressed from 0x1000 0000 to 0x1000 7FFF, the upper SRAM block from 0x2007 C000 to 0x2008 3FFF. The lower SRAM block is denoted with

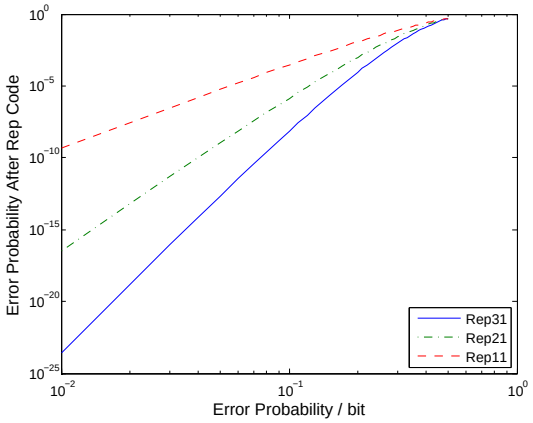


Figure 15.4.: Performance of different repetition code lengths.

block A and the upper SRAM block is denoted with *block B* in the remainder of the text. To analyze the performance the first 16 kB were used. Here, 16 kB were assumed to be sufficient to provide enough bits for data and repetition code.

To specify the performance of the SRAM blocks the specification parameters described in Section 4 are used. The following parameters are determined: mean value, error rate, the temperature behavior(error rate at different temperature corners), correlation between bits, correlation between chips and memory effect. The analysis results are presented in the following sections.

15.3.1. Mean Value

In an ideal case the mean value should be 0.5. In the case of the microcontroller PUF the mean value of Block A was 0.5205, and the mean value of block B was 0.5559. As can be seen in Figure 15.5 the results of both blocks are within an acceptable range on the binomial probability density function (pdf). The pdf of a binomial distribution can be determined by using the following equation:

$$P(k) = \binom{n}{k} p^k q^{n-k}, \tag{15.1}$$

where k is the number of ones (0 to n). n is the total number of bits ($n = 16384$) and $p = q = 0.5$ is the probability that a one/zero occurs. k is normalized by $\frac{1}{n}$.

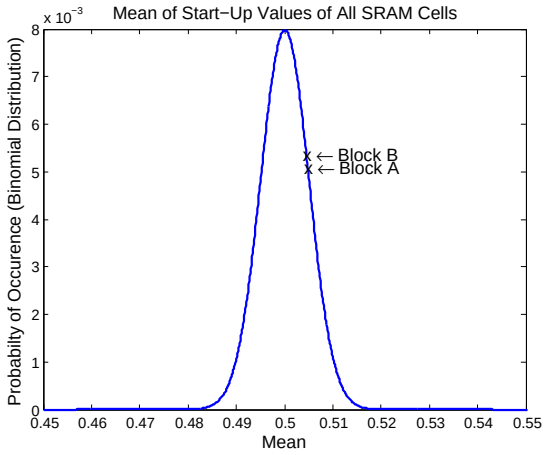


Figure 15.5.: Mean of SRAM blocks A and B on binomial pdf.

15.3.2. Error Rate

The intra-chip Hamming distance HD_{intra} was determined for both SRAM blocks. The power-down phases between the single measurements were $>10\,s$ to avoid hysteresis effects. At block A the intra-chip HD is $HD_{intraA} = 8\,\%$. At block B the intra-chip HD is $HD_{intraA} = 18\,\%$. The difference between the HD_{intra} of the two blocks indicates a different structure of the two blocks. The error rate of block A is less than half of the error rate of block B. Figure 15.6 shows the Hamming disdance of both blocks. The results are shown in Figure 15.6. The error rate was also

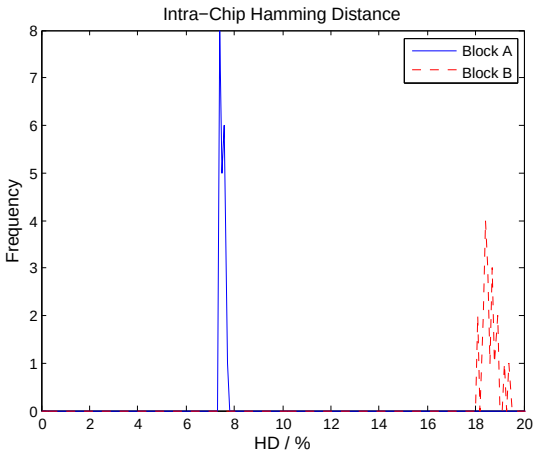


Figure 15.6.: Hamming distance of block A and block B at room temperature.

determined at two further temperatures. The temperatures were chosen to be 0°C and 80°C. The BERs were determined with respect to the reference vector which was defined at 25°C. The following HD_{intra} could be achieved:

$HD_{intraA0°C} = 8.92\,\%$
$HD_{intraA80°C} = 8.22\,\%$
$HD_{intraB0°C} = 29.40\,\%$
$HD_{intraB80°C} = 22.20\,\%$

15.3.3. Correlation Between Bits

The correlation between the different bits was determined as described in Equation 4.8. To move the mean to zero all '0' were substituted by '-1'. The correlation between the bits of block A can be seen in Figure 15.7a. The correlation between the bits of block B is depicted in Figure 15.7b. As can be seen in the figure, block A does not show any correlation between its bits. In contrast, block B has some strong correlation at gaps of 256 bits and even stronger correlation at gaps of 2048 bits which again shows (like in the case of the error rate) that the two blocks must be built up differently.

15.3.4. Memory Effect

To analyze the memory effect of the SRAM PUF, the following procedure was used: in a first step, a '0' was written in all SRAM PUF cells. After one second the chip was restarted and read out again. These two steps were repeated, but this time all PUF cells were set to '1' and then read out again. The mean values of the second read out were: $\mu_{A0} = 0.5253$, $\mu_{A1} = 0.4951$, $\mu_{B0} = 0.5588$ and $\mu_{B1} = 0.4457$. In a second test the bit error rate was evaluated: $HD_{intraA0} =$

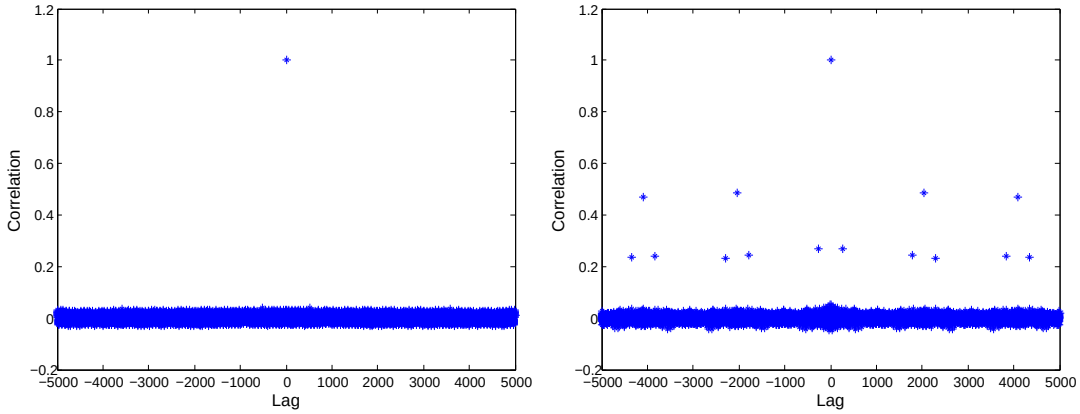


Figure 15.7.: (a) Auto-correlation of block A; (b) Auto-correlation of block B.

6.85 %, $HD_{intraA1} = 7.15\,\%$, $HD_{intraB0} = 16.61\,\%$ and $HD_{intraB1} = 17.39\,\%$. The error rate does not increase after writing in all cells being compared to the error rates in a nominal situation. This either shows that only those cells are affected by the memory effect that are responsible for the errors in first place, or that the time span in which the data was written into the SRAM was chosen too short.

15.3.5. Correlation Between the Chips

The inter-chip HD, HD_{inter}, was determined using three different microcontrollers (1-3). This is a very small number but it should at least give a hint to the real performance. For block A the following correlations were measured: $HD_{interA12} = 49.78\,\%$, $HD_{interA23} = 48.34\,\%$ and $HD_{interA13} = 49.40\,\%$. Block B: $HD_{interB12} = 50.1\,\%$, $HD_{interB23} = 49.71\,\%$ and $HD_{interB13} = 49.88\,\%$. Since the six available inter-chip HDs are close to 50 % and the ideal result would be around 50 %, the SRAM PUF seems to work well with respect to the correlation between the chips. This desired bahavior is shown by both blocks.

15.3.6. Summary: Block A and block B

The results of the BER and the correlation between the chips show that the two SRAM blocks were designed in different ways. Due to this difference, not both of the blocks show the required properties. Block A is qualified much more than Block B. The high error rate as well as the strong correlation makes block B unfeasible with respect to PUF purposes.

Table 15.1.: Measurement results of Block A and B

Property	Block A	Block B
Mean value	0.5205	0.5559
HD_{intra}	8 %	18 %
Correlation between bits	No	Yes
HD_{inter}	48.3 % -49.4 %	49.7 % -50.1 %

15.4. Error Correction Code

Block A was used to test the error correction capability of the repetion code. Even if the error rate is much smaller than that of block B, 8 % of BER is still a high value and thus a convinient repetition code length has to be defined. In this case the repetition factor was chosen to be 31. A 2048 bit key was generated using block A. The correction bits were then generated and written to the flash of the microcontroller using in-application programming (IAP) commands. The whole implementaion of PUF and error correction required 308 byte of program memory for the initial phase and 184 byte for the key generation phase, both in the program memory. To be able to store the correction data, 7.75 kbit were used for the 2048 bit key (repitition factor of 31).

To test the implementation the key generation was repeated 1000 times at different temperatures in the range between 0°C and 80°C. No errors were produced by the PUF in all of the measurement runs after error correction. This is the expected result, since the theoretical error rate of a PUF with 8 % BER of a single bit, 2048 bit key and a repetition factor of 31 can be determined to be 6.85E-07 %.

15.5. Summary

In this chapter a microcontroller-based SRAM PUF was presented. The approach utilizes the internal SRAM cells as well as the internal flash to store the error correction data. Thus, the production costs are very low with respect to the low error rates that can be achieved using the repetition code. The analysis showed that before implementing a PUF on an internal SRAM the SRAM has to be checked carefully towards its feasibility as a PUF: both, the inter-chip and the intra-chip Hamming distance have to be within a pre-defined range. Furthermore, the memory effect of cells should be checked, especially if the cells are also used during regular usage of the SRAM. In the presented case, only the SRAM block A turned out to be feasible for PUF purposes.

In future microcontroller SRAM PUFs, the error correction code could be improved by adding a block code to a smaller repetition code. Thus, the demand of NVM could be reduced. Nevertheless, a higher complexity of the error correction would lead to a higher computational effort on the microcontroller which could turn out to be a problem for some microcontrollers.

16. Conclusion

by Christoph Boehm and Maximilian Hofer

16.1. The PUF-Design Process

To conclude the work, the whole PUF design workflow is recapitulated in Figure 16.1. It includes all the steps that are required to design a silicon PUF cell and that were carried out during the design of the introduced test chips.

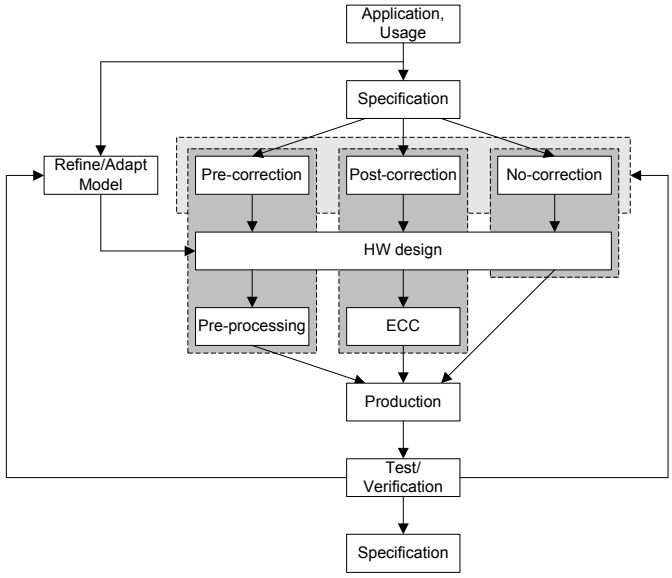


Figure 16.1.: PUF-design workflow.

First of all, the requirements must be defined (see Chapter 2, 3). If the use cases are known and the future application is defined, a specification can be created (see Chapter 4). Depending on the application and the specification, the type of PUF has to be chosen in a way to fit all the requirements and to provide a small error rate at the same time. This is important, since there are different requirements, for example between PUFs used for identification purposes and PUFs used for cryptographic purposes. Furthermore, the models for the simulation have to be adapted and refined for PUF simulation purposes (Chapter 7). The better the models are, the better the estimation of future error rates will be and thus, error correction can be dimensioned already during the design phase reliably.

Once the models are accurate and correction methods are chosen, the PUF cell can be designed (Chapter 6,8 - 11, 12 -15). After the hardware design of the PUF cell, the error rate can be estimated and so the pre-processing method or the error correction code can be implemented in one of the two design methodologies. This can either be a hardware or a software implementation. Afterwards the chips can be produced. To be able to evaluate the chip accurately, different wafer corners should be available. As soon as hardware is available, the verification should be done in the whole pre-defined region of operation (see chapter 4).

16.2. Critical Points

16.2.1. PUF Simulation

As we have seen in chapter 7 the simulated results deviate from the real results a lot. Although the existing models are sufficient enough for most circuits, the models are not suitable to predict the error rate $HD_{intra}$of the future chips, especially at the temperature corners.

16.2.2. The Simpler the Better

If the circuit is complex and many components are involved in the decision process of the PUF cell, all the components influence the behavior of the circuit. Due to the inaccuracies of all components, the mean mismatch becomes smaller and smaller with an increasing number of involved components. Due to the smaller mismatch the error rates increases. However, one important goal of PUF-cell circuit design must be to minimize the number of involved components. The best way to do this is to design a simple circuit. If the circuit only consists of a few parts, the number of parts influencing the output is also small. Due to this fact the simple statement can be declared: the simpler the better.

16.2.3. Exact Specification Parameters

For different reasons, an exact specification is necessary. It is important for the design of the error correction to predefine the maximal allowed error rate HD_{intra}, for example. For specific applications in the area of mobile devices the energy consumption may also be important. It can be seen that different parameters are crucial for different applications (see Chapter 4).

16.2.4. Using Pre- and Post-Processing

As we have shown in Part 2 and Part 3 there are different basic approaches to PUF circuits. Without any implementation to error correction like pre-processing or post-processing methods it is impossible to get stable PUF outputs. Intuitively, noise is the problem. Looking at the results from the practical realizations the influence of changing temperature often causes higher error rates than the influence of noise. Thus, if an error-free output is needed, additional steps have to be undertaken. The first one was presented in Chapter 5. According to this approach extra helper data were needed to reduce the error rate with an error correction code (ECC). Concerning high error rates of few percentages these codes can be very complex. In Chapters 8 - 11 different pre-processing approaches were introduced to reduce the error rate in order to make ECC simpler or even needless.

A. List of Publications

The following list provides the work that was published by the authors during and before the work on PUCKMAES.

- Hofer, M.; Böhm, C.:
  Using SRAMs as Physical Unclonable Functions. - in: Austochip. (2009), S. 160 - 168
- Hofer, M.; Böhm, C.:
  An Alternative to Error Correction for SRAM-Like PUFs. - in: CHES - Workshop on Cryptographic Hardware and Embedded Systems. (2010), S. 335 - 350
- Hofer, M.; Böhm, C.:
  Error Correction Coding for Physical Unclonable Functions. - in: Austrochip, Workshop on Microelectronics. (2010) In Press
- Hofer, M.; Böhm, C.; Bock, H.:
  Identifikation, Authentifizierung und Schlüsselgenerierung mittels Physical Unclonable Functions . - in: Informationstagung Mukroelektronik ME10. (2010), S. 267 - 272
- Hofer, M.; Böhm, C.; Bock, H.:
  UPM - Useful PUF marking. - in: SECSI - Secure Component and System Identification, Workshop Record. (2010) In Press
- Hofer, M.; Böhm, C.; Bock, H.:
  Identifikation, Authentifizierung und Schlüsselgenerierung mittels Physical Unclonable Functions - Übersicht und Anwendungsgebiete. - in: e u i - Elektrotechnik und Informationstechnik 127 (2010) 4 4.2010, S. 72 - 77
- Böhm, C.; Hofer, M.; Pribyl, W.:
  A Microcontroller SRAM-PUF. - in: Proceedings of NSS 2011 (2011), S. 269 - 273 International Conference on Network and System Security ; 2011
- Hofer, M.; Böhm, C.; Pribyl, W.:
  Temperature Behavior Mismatch of Halo Implanted Short Channel Transistors and its Influence on PUF Circuits. - in: ISIC2011. (2011) In Press
- Böhm, C.; Hofer, M.:
  Test Work Flow and Specification for Physical Unclonable Functions. - in: Tagungsband (2011), S. 86 - 90 Austrochip ; 2011
- Farago, P.; Jackum, T.; Böhm, C.; Hofer, M.:
  Illustration Of Automatic Digital Synthesis For Camellia-128 Cipher Algorithm . - in: Acta technica Napocensis / Electronics and telecommunications (2012) In Press
- Böhm, C.; Pernkopf, F.:
  Effective metric-based speaker segmentation in the frequency domain. - in: IEEE International Conference on Acoustics, Speech, and Signal Processing ; 2009 (2009), S. 4081 - 4084

- Hofer, M.; Mayr, B.:
  Design of a Digital Pulse Modulated Audio Power Amplifier. - in: Workshop on Microelectronics. (2008), S. 79 - 84

Bibliography

[1] Actel, "Design security in nonvolatile flash and antifuse fpgas," *Security Backgrounder*, 2002.

[2] W. Adi and B. Soudan, "Bio-inspired electronic-mutation with genetic properties for secured identification," *Bio-inspired, Learning, and Intelligent Systems for Security, 2007. BLISS 2007. ECSIS Symposium on*, pp. 133–136, 2007.

[3] K. Agarwal and S. Nassif, "The impact of random device variation on sram cell stability in sub-90-nm cmos technologies," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 1, pp. 86 –97, 2008.

[4] M. A. Alam, "On the reliability of micro-electronic devices: An introductory lecture on negative bias temperature instability," Oct 2005. [Online]. Available: http://nanohub.org/resources/193

[5] AlteraCorporation, "Protecting intellectual property through fpga design security," 2006. [Online]. Available: http://www.altera.com/literature/ads/fpgadesignsecurity.pdf

[6] ——, "An 341: Using the design security feature in stratix ii and stratix ii gx devices," 2009.

[7] R. Angeles, "Rfid technologies: Supply-chain applications and implementation issues," *Information Systems Management*, vol. 22, no. 1, pp. 51–65, 2005. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1201/1078/44912.22.1.20051201/85739.7

[8] F. Armknecht, R. Maes, A. Sadeghi, F.-X. Standaert, and C. Wachsmann, "A formalization of the security features of physical functions," in *Security and Privacy (SP), 2011 IEEE Symposium on*, may 2011, pp. 397 –412.

[9] A. Asenov, A. Brown, J. Davies, S. Kaya, and G. Slavcheva, "Simulation of intrinsic parameter fluctuations in decananometer and nanometer-scale mosfets," *Electron Devices, IEEE Transactions on*, vol. 50, no. 9, pp. 1837 – 1852, sept. 2003.

[10] D. Barr, "Security application using silicon fingerprint identification," US Patent 7 291 507, 2009.

[11] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede, "Public-key cryptography for rfid-tags," in *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, march 2007, pp. 217 –222.

[12] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer, "High-performance cmos variability in the 65-nm regime and beyond," *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp. 433 –449, july 2006.

[13] M. Bhargava, C. Cakir, and K. Mai, "Attack resistant sense amplifier based pufs (sa-puf) with deterministic and controllable reliability of puf responses," in *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, 2010, pp. 106 –111.

[14] C. Böhm and M. Hofer, "Test work flow and specification for physical unclonable functions," *Austrochip ; 2011*, pp. 86 – 90, 2011.

[15] W. Bidermann and M. Frank, "Using a time invariant statistical process variable of a semiconductor chip as the chip identifier," US Patent 7 291 507, 2007.

[16] J. Black, "Electromigration;a brief survey and some recent results," *Electron Devices, IEEE Transactions on*, vol. 16, no. 4, pp. 338 – 347, apr 1969.

[17] C. Boesch, J. Guajardo, A. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on fpgas," in *Proceedings of CHES 2008*, 2008.

[18] C. Bohm, M. Hofer, and W. Pribyl, "A microcontroller sram-puf," in *Network and System Security (NSS), 2011 5th International Conference on*, sept. 2011, pp. 269 –273.

[19] L. Bolotnyy and G. Robins, "Physically unclonable function-based security and privacy in rfid systems," *Pervasive Computing and Communications, 2007. PerCom '07. Fifth Annual IEEE International Conference on*, pp. 211–220, March 2007.

[20] J. Bringer, H. Chabanne, and T. Icart, "Improved privacy of the tree-based hash protocols using physically unclonable function," vol. 5229, pp. 77–91, 2008, 6th International Conference on Security and Cryptography for Networks, Amalfi, ITALY, SEP 10-12, 2008.

[21] K. Bulgreen, "Sildenafil tablets," 2010. [Online]. Available: http://www.pfizer.com/files/products/uspi-viagra.pdf

[22] H. Busch, M. SotÃ¡kovÃ¡, S. Katzenbeisser, and R. Sion, "The puf promise," in *Trust and Trustworthy Computing*, ser. Lecture Notes in Computer Science, A. Acquisti, S. Smith, and A.-R. Sadeghi, Eds. Springer Berlin / Heidelberg, 2010, vol. 6101, pp. 290–297, 10.1007/978-3-642-13869-0-21. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13869-0-21

[23] A. Cabbibo, J. Conder, and M. Jacobs, "Feed forward test methodology utilizing device identification," *Test Conference, 2004. Proceedings. ITC 2004. International*, pp. 655–660, Oct. 2004.

[24] H. Casier, M. Steyaert, and A. v. Roermund, Eds., *Analog Circuit Design: Robust Design, Sigma Delta Converters, RFID*. Springer Science+Business Media B.V., 2011.

[25] L. Chang, D. Fried, J. Hergenrother, J. Sleight, R. Dennard, R. Montoye, L. Sekaric, S. McNab, A. Topol, C. Adams, K. Guarini, and W. Haensch, "Stable sram cell design for the 32 nm node and beyond," in *VLSI Technology, 2005. Digest of Technical Papers. 2005 Symposium on*, june 2005, pp. 128 – 129.

[26] J. D. Cheek, "Angled halo implant tailoring using implant mask," US Patent 6 372 587, 2002.

[27] B.-D. Choi, T.-W. Kim, M.-K. Lee, K.-S. Chung, and D. Kim, "Integrated circuit design for physical unclonable function using differential amplifiers," *Analog Integrated Circuits and Signal Processing*, pp. 1–8, 2010, 10.1007/s10470-010-9563-8. [Online]. Available: http://dx.doi.org/10.1007/s10470-010-9563-8

[28] D. Clarke, B. Gassend, M. Van Dijk, and S. Devadas, "Authentication of integrated circuits," US Patent 7 840 803, 2010.

[29] G. Csaba, X. Ju, Z. Ma, Q. Chen, W. Porod, J. Schmidhuber, U. Schlichtmann, P. Lugli, and U. Ruhrmair, "Application of mismatched cellular nonlinear networks for physical cryptography," in *Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on*, feb. 2010, pp. 1 –6.

[30] Dekker, Gerard, and Johan, "Preventing cloning of receivers of encrypted messages," EP Patent 09 176 381.3, 2009.

[31] *BSIM4.6.4 MOSFET Model - User's Manual*, Department of Electrical Engineering and Computer Sciences, Univ. California, Berkeley, 2009. [Online]. Available: http://www-device.eecs.berkeley.edu/~bsim/

[32] C. Desset, B. Macq, and L. Vandendorpe, "Block error-correcting codes for systems with a very high ber: Theoretical analysis and application to the protection of watermarks," *Signal Processing: Image Communication*, vol. 17, pp. 409–421, 2002.

[33] S. Devadas and B. Gassend, "Data protection and cryptographic functions using a device-specific value," US Patent 7 818 569, 2010.

[34] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, "Design and implementation of puf-based "unclonable" rfid ics for anti-counterfeiting and security applications," in *RFID, 2008 IEEE International Conference on*, april 2008, pp. 58 –64.

[35] S. Devadas and T. Ziola, "Securely field configurable device," US Patent 7 702 927, 2010.

[36] Y. Dodis, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *Proceedings of Eurocrypt 2007*, pp. 523–540, 2007.

[37] P. Drennan and C. McAndrew, "A comprehensive mosfet mismatch model," in *Electron Devices Meeting, 1999. IEDM Technical Digest. International*, 1999, pp. 167 –170.

[38] ——, "Understanding mosfet mismatch for analog design," *Solid-State Circuits, IEEE Journal of*, vol. 38, no. 3, pp. 450 – 456, mar 2003.

[39] EuropeanTelecommunicationsStandardInstitute, "En 300 421," *European Standard (Telecommunications series)*, vol. V1.1.2 (1997-08), 1997.

[40] EuropeanTelecommunicationsStandardsInstitute, "Digital video broadcasting (dvb); extensions to the common interface specification," 1999.

[41] ——, "En 302 307," *European Standard (Telecommunications series)*, vol. V1.2.1 (2009-08), 2009.

[42] K. Finkenzeller and K. Cox, *RFID Handbook*. John Wiley & Sons, 2010.

[43] D. Forney, "Concatenated codes," *Technical report (Massachusetts Institute of Technology. Research Laboratory of Electronics)* ., vol. 440, pp. 103–104, 1965.

[44] D. Ganta, V. Vivekraja, K. Priya, and L. Nazhandali, "A highly stable leakage-based silicon physical unclonable functions," in *VLSI Design (VLSI Design), 2011 24th International Conference on*, jan. 2011, pp. 135 –140.

[45] B. Gassend, "Physical random functions," Master's thesis, Massachusetts Institute of Technology, The Stata Center, 32 Vassar Street, Cambridge, Massachusetts 02139, February 2003.

[46] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," pp. 148–160, 2002.

[47] B. Gassend, D. Lim, D. Clarke, M. v. Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency Computation: Pract. Exper.*, vol. 16, pp. 1077–1098, 2004. [Online]. Available: http://www.cse.msstate.edu/~ramkumar/fulltext.pdf

[48] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for fpga ip protection," in *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, aug. 2007, pp. 189 –195.

[49] ——, "Brand and ip protection with physical unclonable functions," *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 3186–3189, May 2008.

[50] J. e. a. Guajardo, "Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions," *Information System Frontiers*, vol. 11, pp. 19–41, 2009.

[51] J. Guajardo, B. Å koriÄ‡, P. Tuyls, S. Kumar, T. Bel, A. Blom, and G.-J. Schrijen, "Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions," *Information Systems Frontiers*, vol. 11, pp. 19–41, 2009, 10.1007/s10796-008-9142-z. [Online]. Available: http://dx.doi.org/10.1007/s10796-008-9142-z

[52] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds. Springer Berlin / Heidelberg, 2007, vol. 4727, pp. 63–80. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74735-2-5

[53] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, ser. Lecture Notes in Computer Science, M. Joye and J.-J. Quisquater, Eds. Springer Berlin / Heidelberg, 2004, vol. 3156, pp. 925–943.

[54] S. Haehn, "Secure, stable on chip silicon identification," US Patent 7 603 637, 2009.

[55] R. Hamming, "Error detecting and error correcting codes," *Bell System Tech. system*, vol. 29, pp. 147–160, 1950.

[56] G. Hammouri, E. Oeztuerk, B. Birand, and B. Sunar, "Unclonable lightweight authentication scheme," vol. 5308, pp. 33–48, 2008, 10th International Conference on Information and Communications Security, Birmingham, ENGLAND, OCT 20-22, 2008.

[57] A. Hastings, *The Art of Analg Layout*. Prentice Hall, 2001.

[58] HavocscopeLLC. (2011) Counterfeit goods ranking. Havocscope LLC. [Online]. Available: http://www.havocscope.com/black-market/counterfeit-goods/counterfeit-goods-ranking/

[59] R. Helinski, D. Acharyya, and J. Plusquellic, "A physical unclonable function defined using power distribution system equivalent resistance variations," in *Proceedings of the 46th Annual Design Automation Conference*, ser. DAC '09. New York, NY, USA: ACM, 2009, pp. 676–681. [Online]. Available: http://doi.acm.org/10.1145/1629911.1630089

[60] ——, "Quality metric evaluation of a physical unclonable function derived from an ic's power distribution system," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, june 2010, pp. 240 –243.

[61] J. Hirase and T. Furukawa, "Chip identification using the characteristic dispersion of transistor," *Test Symposium, 2005. Proceedings. 14th Asian*, pp. 188–193, Dec. 2005.

[62] M. Hofer and C. Boehm, "An alternative to error correction for sram-like pufs," in *CHES - Workshop on Cryptographic Hardware and Embedded Systems*, 2010, pp. 335–350.

[63] ——, "Error correction coding for physical unclonable functions," in *Austrochip, Workshop on Microelectronics*, 2010.

[64] ——, "Identifikation, authentifizierung und schluesselgenerierung mittels physical unclonable functions," in *Informationstagung Mikroelektronik ME2010*, 2010, pp. 267–272.

[65] D. E. Holcomb, W. P. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in *Proceedings of the Conference on RFID Security*, July 2007. [Online]. Available: http://www.cs.umass.edu/~kevinfu/papers/holcomb-FERNS-RFIDSec07.pdf

[66] ——, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, September 2009. [Online]. Available: http://www.cs.umass.edu/~kevinfu/papers/holcomb-FERNS-IEEE-Computers.pdf

[67] C. Horng, "Method of authenticating an object or entity using a random binary id code subject to bit drift," US Patent 6 802 447, 2004.

[68] C. Hu, S. C. Tam, F.-C. Hsu, P.-K. Ko, T.-Y. Chan, and K. Terrill, "Hot-electron-induced mosfet degradation – model, monitor, and improvement," *Solid-State Circuits, IEEE Journal of*, vol. 20, no. 1, pp. 295 – 305, feb 1985.

[69] Infineon, *Chip Card & Security ICs - my-d vicinity, SRF 55V10S*, secure mode operation ed., Infineon, July 2007.

[70] Irdeto, "the sky's the limit," 2010.

[71] A. Islam and M. Alam, "Mobility enhancement due to charge trapping x00026; defect generation: Physics of self-compensated bti," in *Reliability Physics Symposium (IRPS), 2010 IEEE International*, may 2010, pp. 65 –72.

[72] H. L. . P. S. Jain, A., "Biometric identification," *Communications of the ACM*, vol. 43(2), pp. 91–98, 2000. [Online]. Available: http://helios.et.put.poznan.pl/~dgajew/download/PUT/SEMESTR-10/IO/FACE-RECOGNITION/BiometricsACM.pdf

[73] V. R. Joan Daemen, *The design of Rijndael: AES–the advanced encryption standard*, 1st ed., ser. Information Security and Cryptography. Springer Berlin Heidelberg, 2 2002, iSBN-13: 978-3540425809.

[74] M. Kassem, M. Mansour, A. Chehab, and A. Kayssi, "A sub-threshold sram based puf," in *Energy Aware Computing (ICEAC), 2010 International Conference on*, dec. 2010, pp. 1 –4.

[75] D. Kholodnyak, P. Turalchuk, A. Mikhailov, S. Dudnikov, and I. Vendik, "3d antenna for uhf rfid tags with eliminated read-orientation sensitivity," in *Microwave Conference, 2006. 36th European*, sept. 2006, pp. 583 –586.

[76] H. Koerner, "Method for identifying electronic circuits and identification device," US Patent 7 893 699, 2011.

[77] Y. Koutsoyannopoulos and Y. Papananos, "Systematic analysis and modeling of integrated inductors and transformers in rf ic design," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 47, no. 8, pp. 699 –713, aug 2000.

[78] R. Kumar, V. C. Patil, and S. Kundu, "Design of unique and reliable physically unclonable functions based on current starved inverter chain," in *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, july 2011, pp. 224 –229.

[79] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly puf protecting ip on every fpga," *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pp. 67–70, June 2008.

[80] Y. Lao and K. K. Parhi, "Reconfigurable architectures for silicon physical unclonable functions," in *Electro/Information Technology (EIT), 2011 IEEE International Conference on*, may 2011, pp. 1 –7.

[81] J. Lee, D. Lim, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, june 2004, pp. 176 – 179.

[82] M. Lehtonen, T. Staake, and F. Michahelles, "From identification to authentication – a review of rfid product authentication techniques," in *Networked RFID Systems and Lightweight Cryptography*, D. C. Ranasinghe and P. H. Cole, Eds. Springer Berlin Heidelberg, 2008, pp. 169–187.

[83] D. Lim, J. Lee, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.

[84] L. Lin, D. Holcomb, D. K. Krishnappa, P. Shabadi, and W. Burleson, "Low-power subthreshold design of secure physical unclonable functions," in *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, aug. 2010, pp. 43 –48.

[85] S. Lin and D. Costello, *Error Control Coding, Fundamentals and Applications*, second edition ed., P. Hall, Ed. Pearson Education International, 2004.

[86] K. Lofstrom, "System for providing an integrated circuit with a unique identification," US Patent 6 161 213, 2000.

[87] ——, "Icid - a robust, low cost integrated circuit identification method, ver. 0.9," 2007. [Online]. Available: http://www.kl-ic.com/papers.html

[88] K. Lofstrom, D. Castaneda, B. Graff, and A. Cabbibo, "Icid - tracing individual die from wafer test through end-of-life," in *International Mixed Signal Test Workshop*, 2004.

[89] K. Lofstrom, W. Daasch, and D. Taylor, "Ic identification circuit using device mismatch," *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, pp. 372–373, 2000.

[90] E. Lucero, "Balanced cells with fabrication mismatches that produce a unique number generator," US Patent 7 482 657, 2009.

[91] ——, "Method of forming a unique number," US Patent 7 602 666, 2009.

[92] D. J. C. Mackay, *Information Theory, Inference and Learning Algorithms*, Mackay, D. J. C., Ed., Oct. 2003.

[93] R. Maes, P. Tuyls, and I. Verbauwhede, "Intrinsic pufs from flip-flops on reconfigurable devices," in *3rd Benelux Workshop on Information and System Security (WISSec 2008)*, Eindhoven,NL, 2008, p. 17.

[94] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security*, ser. Information Security and Cryptography, D. Basin, U. Maurer, A.-R. Sadeghi, and D. Naccache, Eds. Springer Berlin Heidelberg, 2010, pp. 3–37. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14452-3-1

[95] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of ro-puf," in *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, 2010, pp. 94 –99.

[96] A. Maiti and P. Schaumont, "Improving the quality of a physical unclonable function using configurable ring oscillators," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, 312009-sept.2 2009, pp. 703 –707.

[97] ——, "Improved ring oscillator puf: An fpga-friendly secure primitive," *Journal of Cryptology*, pp. 1–23, 2010.

[98] M. Majzoobi, F. Koushanfar, and S. Devadas, "Fpga-based true random number generation using circuit metastability with adaptive feedback control," in *CHES*, 2011, pp. 17–32.

[99] M. Marunaka, "Method for identifying semiconductor integrated circuit device, method for manufacturing integrated circuit device, semiconductor integrated circuits device and semiconductor chip," US Patent 6 941 536, 2001.

[100] S. Meguerdichian and M. Potkonjak, "Device aging-based physically unclonable functions," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, june 2011, pp. 288 –289.

[101] W. Mehuron, *DATA ENCRYPTION STANDARD (DES)*, U.S. DEPARTMENT OF COMMERCE Std. FIPS PUB 46-3, 10 1999. [Online]. Available: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[102] C. Mezzomo, A. Bajolet, A. Cathignol, R. Di Frenza, and G. Ghibaudo, "Characterization and modeling of transistor variability in advanced cmos technologies," *Electron Devices, IEEE Transactions on*, vol. 58, no. 8, pp. 2235 –2248, aug. 2011.

[103] T. Moon, *Error Correction Codeing: Mathmatical Methods and Algorithms*. John Wiley & Sons, Inc., 2005.

[104] A. Moradi, A. Barenghi, T. Kasper, and C. Paar, "On the vulnerability of fpga bitstream encryption against power analysis attacks," *Memory*, vol. Report 201, pp. 111–123, 2011. [Online]. Available: http://eprint.iacr.org/2011/390

[105] A. Moradi, M. Kasper, and C. Paar, "On the portability of side-channel attacks," *Design*, vol. Report 201, pp. 5–7, 2011. [Online]. Available: http://eprint.iacr.org/2011/391

[106] J. Moreira, *Essentials of Error-Control Coding*. Wiley, 2006.

[107] S. Morozov, A. Maiti, and P. Schaumont, "A comparative analysis of delay based puf implementations on fpga," Cryptology ePrint Archive, Report 2009/629, 2009, http://eprint.iacr.org/.

[108] NSA. (2009, Jan) The case for elliptic curve cryptography. National Security Agency - Central Service. [Online]. Available: http://www.nsa.gov/business/programs/elliptic-curve.shtml

[109] OECD, "The economic impact of counterfeiting and piracy," 2007. [Online]. Available: www.oecd.org/dataoecd/13/12/38707619.pdf

[110] ——, "Magnitude of counterfeiting and piracy of tangible products: An update," 2009.

[111] T. Okayasu, S. Sugawa, and A. Teramoto, "Electronic device identifying method," US Patent 7 812 595, 2010.

[112] E. Ozturk, G. Hammouri, and B. Sunar, "Physical unclonable function with tristate buffers," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, May 2008, pp. 3194 –3197.

[113] R. Pappu, R. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *SCIENCE*, vol. 297, no. 5589, pp. 2026–2030, SEP 20 2002.

[114] M. PELGROM, A. DUINMAIJER, and A. WELBERS, "Matching properties of mos-transistors," *IEEE JOURNAL OF SOLID-STATE CIRCUITS*, vol. 24, no. 5, pp. 1433–1440, OCT 1989.

[115] M. Pelgrom, H. Tuinhout, and M. Vertregt, "Transistor matching in analog cmos applications," *Electron Devices Meeting, 1998. IEDM '98 Technical Digest., International*, pp. 915–918, Dec 1998.

[116] R. Posch, "Protecting devices by active coating," *Journal of Universal Computer Science*, vol. 4, no. 7, pp. 652–668, 1998.

[117] D. Puntin, S. Stanzione, and G. Iannaccone, "Cmos unclonable system for secure authentication based on device variability," in *Solid-State Circuits Conference, 2008. ESSCIRC 2008. 34th European*, 2008, pp. 130 –133.

[118] G. Qu and C.-E. Yin, "Temperature-aware cooperative ring oscillator puf," in *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, 2009, pp. 36 –42.

[119] D. C. Ranasinghe, D. W. Engels, and P. H. Cole, "Security and privacy: Modest proposals for low-cost rfid," 2004.

[120] G. Ribes, J. Mitard, M. Denais, S. Bruyere, F. Monsieur, C. Parthasarathy, E. Vincent, and G. Ghibaudo, "Review on high-k dielectrics reliability issues," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 1, pp. 5 – 19, march 2005.

[121] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, February 1978. [Online]. Available: http://doi.acm.org/10.1145/359340.359342

[122] I. V. Roel Maes, Pim Tuyls, "Statistical analysis of silicon puf responses for device identification," *Katholieke Universiteit Leuven: ESAT-COSIC*, 2008.

[123] K. Rosenfeld, E. Gavas, and R. Karri, "Sensor physical unclonable functions," in *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, 2010, pp. 112 –117.

[124] D. Roy, J. Klootwijk, N. Verhaegh, H. Roosen, and R. Wolters, "Comb capacitor structures for on-chip physical uncloneable function," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 22, no. 1, pp. 96–102, Feb. 2009.

[125] U. Schaper and J. Einfeld, "Matching model for planar bulk transistors with halo implantation," *Electron Device Letters, IEEE*, vol. 32, no. 7, pp. 859 –861, july 2011.

[126] Securikett, "Securikett online image katalog," http://www.securikett.com, IZ-NÖ SÜD, Strasse 10, Objekt 48, A-2355 Wiener Neudorf, 2011.

[127] G. Selimis, M. Konijnenburg, M. Ashouei, J. Huisken, H. de Groot, V. van der Leest, G.-J. Schrijen, M. van Hulst, and P. Tuyls, "Evaluation of 90nm 6t-sram as physical unclonable function for secure key generation in wireless sensor nodes," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, may 2011, pp. 567 –570.

[128] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, pp. 3–55, January 1948. [Online]. Available: http://doi.acm.org/10.1145/584091.584093

[129] C. Shannon, "Communication theory of secrecy systems," pp. 656–, 1949.

[130] E. Simpson and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," in *Cryptographic Hardware and Embedded Systems-CHES 2006. 8th International Workshop. Proceedings (Lecture Notes in Computer Science Vol.4249)*, L. Goubin and M. Matsui, Eds. Berlin, Germany: Springer, 2006 2006, Conference Paper, pp. 311–23, Cryptographic Hardware and Embedded Systems-CHES 2006. 8th International Workshop. Proceedings, 10-13 October 2006, Yokohama, Japan.

[131] W. Simpson, *PPP Challenge Handshake Authentication Protocol (CHAP)*, Computer Systems Consulting Services Std., August 1996.

[132] S. Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. London: Fourth Estate, 1999.

[133] B. Skoric, S. Maubach, T. Kevenaar, and P. Tuyls, "Information-theoretic analysis of capacitive physical unclonable functions," *JOURNAL OF APPLIED PHYSICS*, vol. 100, no. 2, JUL 15 2006.

[134] B. Skoric, P. Tuyls, and W. Ophey, "Robust key extraction from physical uncloneable functions," vol. 3531, pp. 407–422, 2005, 3rd International Conference on Applied Cryptography and Network Security, New York, NY, JUN 07-10, 2005.

[135] A. Sreedhar and S. Kundu, "Physically unclonable functions for embeded security based on lithographic variation," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, march 2011, pp. 1 –6.

[136] W. Stallings, *Network Security Essentials: Applications and Standards*, 1st ed. Prentice Hall PTR Upper Saddle River, NJ, USA, 1999.

[137] S. Stanzione, D. Puntin, and G. Iannaccone, "Cmos silicon physical unclonable functions based on intrinsic process variability," *Solid-State Circuits, IEEE Journal of*, vol. PP, no. 99, p. 1, 2011.

[138] Y. Su, J. Holleman, and B. Otis, "A digital 1.6 pj/bit chip identification circuit using process variations," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, pp. 69–77, Jan. 2008.

[139] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," pp. 9–14, 2007.

[140] J. Sune, I. Placencia, N. Barniol, E. Farres, and X. Aymerich, "Degradation and breakdown of gate oxides in vlsi devices," *physica status solidi (a)*, vol. 111, no. 2, pp. 675–685, 1989. [Online]. Available: http://dx.doi.org/10.1002/pssa.2211110235

[141] D. Suzuki and K. Shimizu, "The glitch puf: A new delay-puf architecture exploiting glitch shapes," in *Cryptographic Hardware and Embedded Systems, CHES 2010*, ser. Lecture Notes in Computer Science, S. Mangard and F.-X. Standaert, Eds. Springer Berlin / Heidelberg, 2010, vol. 6225, pp. 366–382, 10.1007/978-3-642-15031-9-25. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15031-9-25

[142] N. Takao, "Microcontroller- engineering review," 2007.

[143] S. Trimberger, "Copy protection without non-volatile memory," US Patent 7 941 673, 2011.

[144] Y. Tsividis, *The MOS Transistor*. New York: Oxford University Press, 1999.

[145] Y. Tsividis and C. McAndrew, *Operation and Modeling of the MOS Transistor*, 3rd ed. Oxford University Press, 2011.

[146] H. Tuinhout, A. Montree, J. Schmitz, and P. Stolk, "Effects of gate depletion and boron penetration on matching of deep submicron cmos transistors," in *Electron Devices Meeting, 1997. IEDM '97. Technical Digest., International*, dec 1997, pp. 631 –634.

[147] P. Tuyls, T. Denteneer, J. Linnartz, and E. Verbitskiy, "Method and system for authentication of a physical object," US Patent 8 032 760, 2011.

[148] P. Tuyls and G. Schrijen, "Method of reducing the occurrence of burn-in due to negative bias temperature instability," US Patent 12/921,901, 2009.

[149] P. Tuyls, B. Skoric, S. Stallinga, A. Akkermans, and W. Ophey, "Information-theoretic security analysis of physical uncloneable functions," vol. 3570, pp. 141–155, 2005, 9th International Conference on Financial Cryptography, Roseau, DOMINICA, FEB 28-MAR 03, 2005.

[150] P. Tuyls, "Grey-box cryptography: Physical unclonable functions," vol. 4357, pp. 3–5, 2006, 3rd European Workshop on Security and Privacy in Ad Hoc and Sensor Network, Hamburg, GERMANY, SEP 20-21, 2006.

[151] P. Tuyls and B. Škorić, *AmIware: Hardware Technology Drivers of Ambient Intelligence*, ser. Philips Research Book Series. Springer, 2005, ch. Secret Key Generation from Classical Physics., pp. 421–447.

[152] V. van der Leest, G.-J. Schrijen, H. Handschuh, and P. Tuyls, "Hardware intrinsic security from d flip-flops," in *Proceedings of the fifth ACM workshop on Scalable trusted computing*, ser. STC '10. New York, NY, USA: ACM, 2010, pp. 53–62. [Online]. Available: http://doi.acm.org/10.1145/1867635.1867644

[153] D. Varghese, P. Moens, and M. Alam, "on-state hot carrier degradation in drain-extended nmos transistors," *Electron Devices, IEEE Transactions on*, vol. 57, no. 10, pp. 2704 –2710, oct. 2010.

[154] D. Vogel and M. Okronglis, "Stabilization for random chip identifier circuit," US Patent 7 676 726, 2010.

[155] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, march 2005, pp. 324 – 328.

[156] X. Wang and M. Tehranipoor, "Novel physical unclonable function with process and environmental variations," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 1065 –1070.

[157] R. Want, "An introduction to rfid technology," *Pervasive Computing, IEEE*, vol. 5, no. 1, pp. 25 – 33, jan.-march 2006.

[158] S. A. Weis, "Rfid (radio frequency identification): Principles and applications," *MIT CSAIL*, 2007.

[159] L. Wuidart, M. Bardouillet, and A. Malherbe, "Extraction of a binary code based on physical parameters of an integrated circuit," US Patent 6 836 430, 2004.

[160] L. Wuidart, M. Bardouillet, and L. Plaza, "Diversification of a single integrated circuit identifier," US Patent 7 796 759, 2010.

[161] H. Yu, P. H. W. Leong, P. Zipf, H. Hinkelmann, L. Moller, and M. Glesner, "Towards a unique fpga-based identification circuit using process variations," 2009.

[162] Zeghbroeck. (2011) Principles of semiconductor devices. Boulder. [Online]. Available: http://ecee.colorado.edu/~bart/book/