ulm university universität
uulm

Enhanced Physical Unclonable Function Structures in Mixed-Signal MOS Technology

Markus Schuster
markus.schuster@alumni.uni-ulm.de

Master's Thesis

Institute of Microelectronics
University of Ulm

1st Examiner:	Prof. Dr.-Ing. Maurits Ortmanns
2nd Examiner:	Prof. Dr. rer. nat. Frank Kargl
Supervisor:	Prof. Dr.-Ing. Maurits Ortmanns

2013

Abstract

Physical Unclonable Functions (PUFs) present a promising concept for cryptographic hardware. PUFs provide a challenge-response scheme, where the response is not calculated but determined by a kind of randomness available in the device. Every PUF behaves different, even equally manufactured ones. The Arbiter PUF makes use of race conditions between equally designed delay paths. Although the basic principle is proven to work, it suffers from predictability attacks and a not negligible quantity of unreproducible response bits. This thesis deals with the simulation and enhancement of Arbiter PUFs in CMOS technology. A novel simulation approach is introduced, which allows a reliable prediction of the Arbiter PUF's reproducibility behavior without a transistor level simulation of the whole system. Only the components need to be characterized by circuit simulations using industrial manufacturer models. The resulting PUF can be modeled on a higher level and simulated very fast using statistical distributions. Furthermore the influence of some design parameters on each component's characteristic is investigated. An enhanced parameter set is determined and evaluated using the new simulation approach. Finally the results are verified by transient transistor level simulations for the reference and the enhanced Arbiter PUF. The bit error rate could be reduced by 61 % just by design parameter modifications of specific transistors.

Contents

Chapter 1

Introduction

1.1. Motivation

Modern cryptographic protocols rely on the secrecy of keys. The algorithms are public. This partitioning is state of the art and it is known as Kerckhoffs's Principle in cryptography. An advantage: Publicly known protocols can be analyzed by everyone, which makes sensible mistakes improbable. What remains is the challenge to keep keys secret. Keys are usually bit strings. Copying is possible, easy, and in general unprovable.

Physical Unclonable Functions (PUFs) introduce a functionality which may be utilized for future cryptographic protocols. A PUF is a piece of hardware which transforms an input to an output in a particular way. Usually input and output are bit strings. The mapping from the input to the output depends on some kind of randomness, but the same input should always lead to the same output. The evaluation must be easy while the prediction must be hard, even if complete knowledge of the architecture is available. Thus a PUF provides secret information in a challenge-response procedure, which is illustrated in Figure 1.1. The mapping from a challenge to a response is not done by a mathematical function, where parameters (→keys) are necessary to achieve different mappings, but by some randomness without an external key.

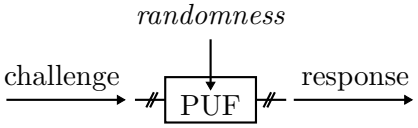


Figure 1.1.: Challenge Response Procedure with PUFs. PUFs generate responses with aid of internal randomness and the external challenge.

What distinguishes a PUF from a key is its unclonability. Applications are conceivable where PUFs could replace secret keys. Again, the protocols and even the hardware design of the PUF can be public. But this time, no key exists which could be shared. All information is random and physically bound to a device.

Building and using a PUF are challenging tasks. Up to date both sufficiently reliable constructions and suitable protocols are missing and have to be developed. This thesis deals with the construction of enhanced PUFs in CMOS technology.

1.2. Properties and Quality Measures

This section deals with properties which are characteristic or desirable for a PUF. Due to the variety of possible constructions and applications this is a complex but important task. Only a short introduction will be given here. Detailed discussions are available in literature, e.g. in [1]. The following paragraph and enumeration is taken from there.

> To simplify the property description, we start from a very basic classification for a PUF as a *physical challenge-response procedure*. Note that already this implicitly assigns two properties to PUFs, i.e. an instantiation of a PUF cannot merely be an abstract concept but it is always (embedded in) a physical entity, and a PUF is a procedure (not strictly a function) with some input-output functionality. [...] For brevity, we use the notation $\Pi : \mathcal{X} \rightarrow \mathcal{Y} : \Pi(x) = y$ to denote the challenge-response functionality of a PUF Π. [...] The informal parts of the property descriptions are clearly marked in sans serif font.

1. **Evaluable**: given Π and x, it is easy to evaluate $y = \Pi(x)$.
2. **Unique**: $y = \Pi(x)$ contains some information about the identity of the physical entity embedding Π.
3. **Reproducible**: $y = \Pi(x)$ is reproducible up to a small error.
4. **Unclonable**: given Π, it is hard to construct a procedure $\Gamma \neq \Pi$ such that $\forall x \in \mathcal{X} : \Gamma(x) \approx \Pi(x)$ up to a small error.
5. **Unpredictable**: given only a set $\mathcal{Q} = \{(x_i, y_i = \Pi(x_i))\}$, it is hard to predict $y_c \approx \Pi(x_c)$ up to a small error, for x_c a random challenge such that $(x_c, \cdot) \notin \mathcal{Q}$.
6. **One-way**: given only y and Π, it is hard to find x such that $\Pi(x) = y$.
7. **Tamper evident**: altering the physical entity embedding Π transforms $\Pi \rightarrow \Pi'$ such that with high probability $\exists x \in \mathcal{X} : \Pi(x) \neq \Pi'(x)$, not even up to a small error.

Figure 1.2 (taken from [2]) shows some PUF properties including the previously mentioned ones and their relations. Some of them are immediately clear, e.g. that the PUF

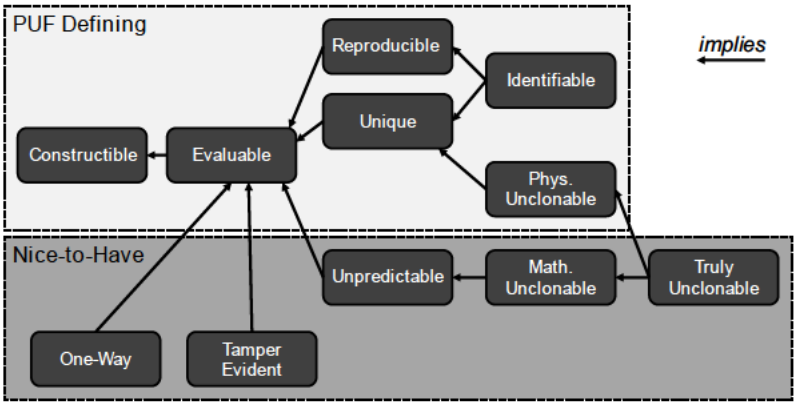


Figure 1.2.: PUF properties incl. relations and categorization. From [2].

is evaluable, others may require further explanations. The requirements may differ per application and a proof of the properties may be infeasible. This work will mainly deal with the properties reproducibility and uniqueness, which are very important properties and measures are available for quantitative evaluations.

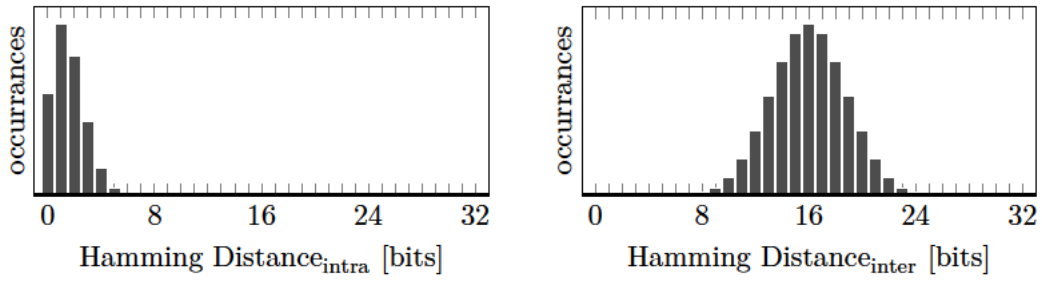


Figure 1.3.: Histograms illustrating reproducibility and uniqueness.
Left: Comparing multiple measurements of the same PUF instance.
Right: Comparing various PUF instances against each other.

The following explanations use the term Hamming Distance. Given two bit strings x_1 and x_2 of equal length, the Hamming Distance is the number of positions at which the bits in x_1 and x_2 are different. For example, $\mathrm{HD}(1010, 1010) = 0$ or $\mathrm{HD}(1010, 0011) = 2$. The Fractional Hamming Distance is the Hamming Distance divided by the bit length. It is a measure which is normalized to be between 0 for equal strings and 1 for completely different strings, independent from the string length.

Unique means, that the PUF's responses are specific to its physical entity. Speaking generally and informal, this means that different PUF instances should produce different responses for the same challenge. This can be analyzed by an inter-distance histogram, see Figure 1.3 (right). In this example the bit length is 32. The responses of various PUF instances are compared to each other and the occurrences of each possible Hamming Distance is recorded. If all response bits are independent and equally distributed 0 or 1, the histogram equals the probability mass function of the binomial distribution $\mathcal{B}(k|p = 0.5, n = 32)$, which is plotted here. A more detailed explanation is given in [3].

Reproducible means, that the same challenge on the same PUF instance should always result in the same response. Otherwise said, the response should not depend on noise or environmental conditions. This property can be illustrated by comparing multiple readouts of the same PUF and same challenge. Again, the Hamming Distance can be used to compare the responses and a histogram may look like Figure 1.3 (left). What is shown there is a binomial distribution $\mathcal{B}(k|p = 0.05, n = 32)$, which corresponds to a bit error rate of 5 %. Ideally all responses differ by 0.

Although this work has a focus on improving these two aims, other properties are not less important. For example, the common Arbiter PUF which will be introduced in section 1.5 is known to be prone to machine learning attacks [4], which violates the aim for unpredictability.

1.3. Applications

Maes et al. [1] make a distinction between three types of application: identification, key generation, and hardware-entangled cryptography. They are briefly introduced below.

It was said that PUF responses depend on the challenge, but the mapping is random. Furthermore it is hard to copy a PUF in a way it exposes the same mapping. Thus a PUF can be used to identify a circuit by comparing responses to known responses recorded in advance. Having sufficient challenge-response pairs this method can even authenticate a circuit, this is known as challenge response authentication. For both purposes a small bit error rate can be allowed.

No errors are allowed for cryptographic keys, e.g. used for encryption. A PUF can be used to determine a secret cryptographic key, which is used for cryptographic operations. The key can be generated on demand. No permanent storage is needed, from which the key could be leaked. No key programming is necessary, because the key is determined intrinsically. Error correction methods can be applied to allow certain bit error rates in PUF responses and generate correct keys anyhow.

Further applications are summarized as hardware-entangled cryptography. An ideal PUF could be modeled as random oracle. A random oracle is a theoretical replacement for a cryptographic hash function. Shortly said, a hash function maps a bit string to a bit string of constant length in a determined way. A cryptographic hash function is a hash function which provides additional properties, e.g. it must be infeasible to generate an input string resulting in a given output string (one way function). Ideal PUFs offer a similar behavior and could thus be applied in cryptographic protocols currently using keyed (parametrized) cryptographic hash functions, e.g. Message Authentication Codes (MAC).

It is also possible that PUFs evolve into a new cryptographic primitive, similar to hash functions, which allows the construction of new cryptographic protocols. However, to comply with the requirements such a primitive must provide some challenging properties, e.g. very reliable responses in terms of bit errors, environmental conditions and tamper

resilience. A lot of research will be necessary to develop sufficiently good PUFs and PUF-optimized applications. To give an outlook: Gassend et al. propose *code, that runs only on a specific processor* in 2002 [5]. Research on this is is also in progress at the Institute of Distributed Systems at Ulm University.

1.4. Basic PUF Principles

There are various ways of implementing a PUF. In literature, mainly two principles are exploited: bi-stability and timing differences. Further constructions and classifications are possible. Constructions are not necessarily electrical. An overview is available in literature, e.g. in [6] or [2], and not given here in detail.

An example exploiting bi-stability is the SRAM PUF. It was proposed by Guajardo et al. [7] and Holcomb et al. [8] in 2007. An SRAM cell is a simple transistor circuit capable of storing one bit of data. However, after power up the state is undefined. If the power up states of various bits differ, the pattern can be characteristic for a specific device and differ for different devices of the same kind.

An example exploiting time differences is the ring oscillator PUF. It is based on delay lines which are connected to act as oscillators. The frequencies of oscillators can be compared to each other. The information which of both oscillators runs faster can be evaluated to a response bit. For different configurations of the oscillators (→challenge) the relation can differ. Due to the equally designed oscillators the responses depend on the specific device and can differ for different devices of the same kind. This idea was already proposed by Gassend et al. [5] in 2002.

This thesis will deal with the Arbiter PUF, which is another kind of delay based PUF. It was introduced by Lee et al. [9] in 2004. Its idea is to introduce a race condition between two equally designed delay paths and decide for the faster one. The construction concept is explained in the next section.

1.5. Arbiter PUF Construction and Principle

Lim, Lee, et al. [4] introduced the Arbiter PUF as following:

> An arbiter-based PUF is composed of delay paths and an arbiter located at the end of the delay paths. [Figure 1.4] depicts an arbiter-based PUF circuit. In this scheme, we excite two delay paths simultaneously and make the transitions race against each other. The arbiter determines which rising edge arrives first and sets its output to 0 or 1 depending on the winner. The circuit takes 64 challenge bits (b_i) as an input to configure the delay paths and generate a 1-bit response as an output.

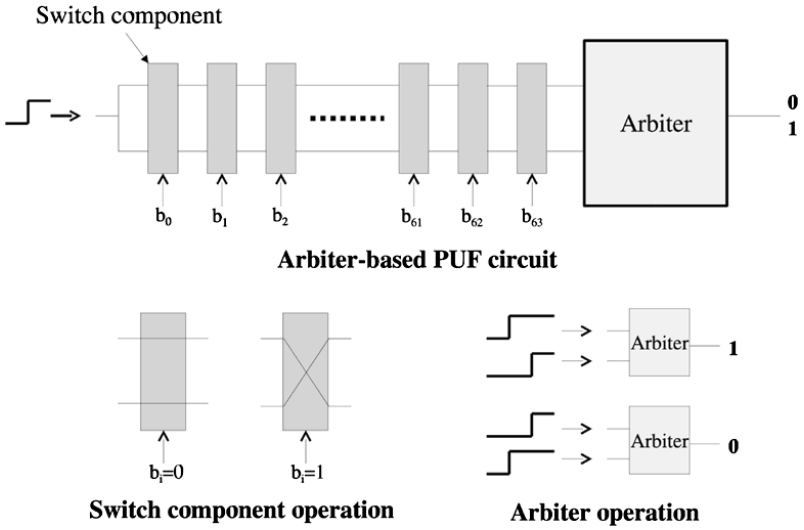


Figure 1.4.: Structure of an arbiter-based PUF (basic arbiter scheme). From [4].

The actual implementation of each component will be discussed in chapter 3, an evaluation is given in chapter 5.

1.6. Arbiter PUF State Of The Art

To the best of the author's knowledge no currently known PUF construction is able to provide all desired properties in an acceptable extent for versatile applications. However there are numerous enhancements on each PUF construction improving a certain property. The following overview is limited to Arbiter PUFs, which are examined in this work exclusively.

The Arbiter PUF proposed in [9] is prone to violating some of the desired PUF properties. The most severe problems affect reproducibility and unpredictability. The bit error rate is not neglectable, e.g. 3 % for a 64 bit Arbiter PUF in TSMC 65nm technology [2]. Machine learning algorithms can predict responses with small error rates, e.g. 5 % prediction error after only 500 training samples for a 64 bit Arbiter PUF [2]. However, these values differ a lot in literature and the absolute numbers have to be regarded with care.

Various countermeasures against predictability were proposed. Lee et al. proposed feed forward paths leading to non-linearities in the delay paths [9]. Majzoobi et al. proposed input and output networks to hide the real PUF behaviour [10]. Rührmair et al. demonstrated attacks on PUFs with and without these protections. It was shown that despite attacks are still possible, the effort in model building and training data increases significantly [11].

A simple countermeasure against noisy responses are multiple readouts and a majority decision. In case of recording a limited number of challenge-response-pairs, e.g. for identification, unreliable challenges can be determined by multiple readouts and avoided. In case a limited range of responses is sufficient, e.g. for key generation, an error correction code can be applied [5] [12]. However, more sophisticated constructions are required to improve the reliability (in terms of reproducibility) of the PUF itself.

There are design attempts claiming a higher reliability. One attempt is current starving. Kumar et al. [13] show that the delay of current starved inverters offers a broader distribution in comparison to normal inverters, namely a significantly higher standard deviation. Unfortunately reliability - as the term is used here - is not examined, but only *expected to be around 100 %* [13], which can be doubted.

To the best of the author's knowledge there is no approach which is proven to solve the problem of unreliability in a way that the bit error rate can be decreased arbitrarily low. However this might be necessary for hardware-entangled cryptography introduced in section 1.3.

1.7. MOS FET Basics

The transistor is the most important element in nanometer scale circuit technology and is introduced here in brief. The following chapters require at least basic knowledge of its behavior, usage, and design parameters. The technology used throughout this thesis is TSMC 90 nm. A typical supply voltage is 1.2 V. All potentials named in this section will not exceed the range of 0 V to 1.2 V. In particular 1.2 V is declared a high voltage.

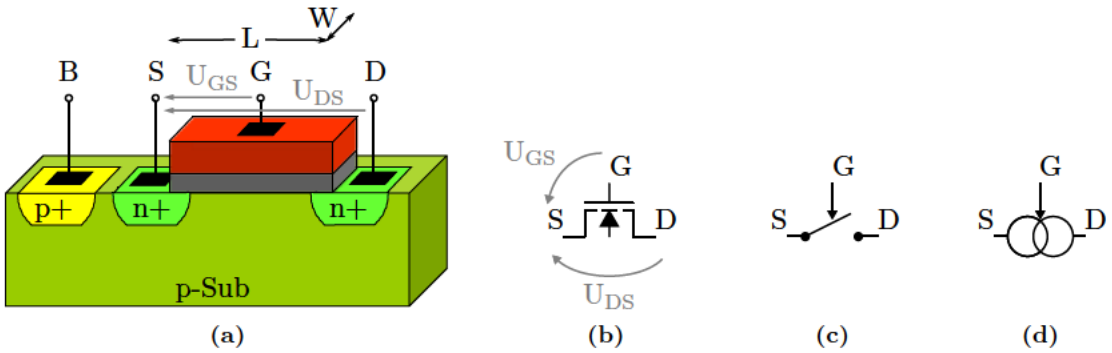
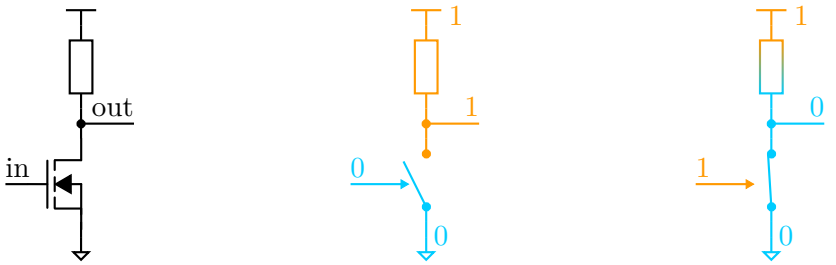


Figure 1.5.: (a) nMOS transistor's bird's view (adapted from [14]), (b) circuit symbol, and illustration of its behavior as (c) switch and (d) current source.

Figure 1.5 (a) displays the basic physical transistor structure in bird's view. It is a simplified model. The image refers to an n-type enhancement-mode metal-oxide-semiconductor field-effect transistor (in the following abbreviated: **nMOS**). Applying the same potential to all pins, no current flows. Even if a voltage is applied between source S and drain D, no significant current flows. When a voltage $U_{GS} \gg 0$ is applied to the gate G, which

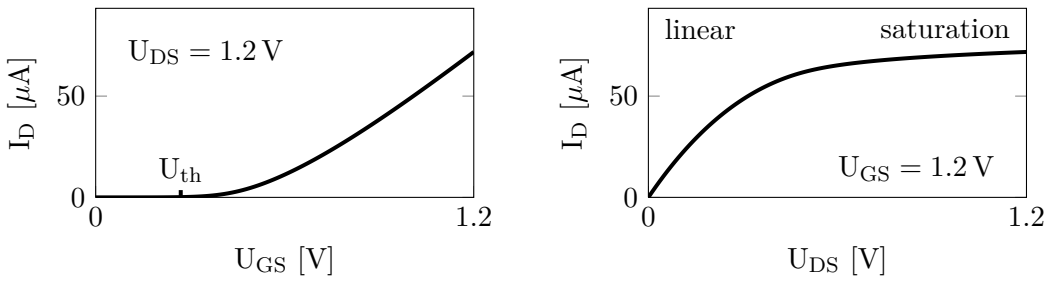
is isolated from the rest by a gate oxide (gray), a channel is invoked between source S and drain D. The channel allows current to flow if a voltage $U_{DS} > 0$ is applied. In the following the source S is fixed to the lowest potential, $U_S = 0\,V$. Figure 1.5 (b) displays the corresponding circuit symbol to the bird's view.



(a) **(b)** **(c)**

Figure 1.6.: (a) nMOS inverter circuit. (b)(c) Illustration of the digital inverter functionality.

For digital circuits the function can be simplified to a voltage controlled switch, see Figure 1.5 (c). At a low gate voltage the switch is off, for sufficiently high voltages the switch is closed. Exploiting this functionality allows building digital circuits like an inverter, see Figure 1.6 (a). The output is high if the input is low (b), the output is pulled low if the input is high (c).



(a) Input Characteristic **(b)** Output Characteristic

Figure 1.7.: Exemplary transistor input characteristic and output characteristic. The drain current as a function of the gate voltage (→input) and drain voltage (→output).

For analog circuits the equivalent model can be expanded to a voltage controlled current source, see Figure 1.5 (d). This is illustrated by the input characteristics, see Figure 1.7 (a). When the gate voltage $U_G = U_{GS}$ exceeds the threshold voltage U_{th}, current starts flowing. The higher U_G, the higher I_D. However, the current I_D does not only depend on U_{GS}, but also on U_{DS}, the voltage over the „current source", see the output characteristic in Figure 1.7 (b). Only for higher voltages, within the so called saturation region, the current I_D is almost independent from U_{DS} and the assumption of a current source is a good one. A circuit exploiting the characteristics is a simple inverting common source amplifier,

see Figure 1.8 (a). The conversion of the input voltage to a current is illustrated in Figure 1.8 (b). The resistor transforms the current to the output voltage. In Figure 1.8 (c) an exemplary waveform is given which demonstrates the inverting voltage amplification. The amplification of the transistor can be expressed with aid of the transconductance $g_m = \frac{\Delta I_D}{\Delta U_G}$. It puts the output current in relation to the input voltage.

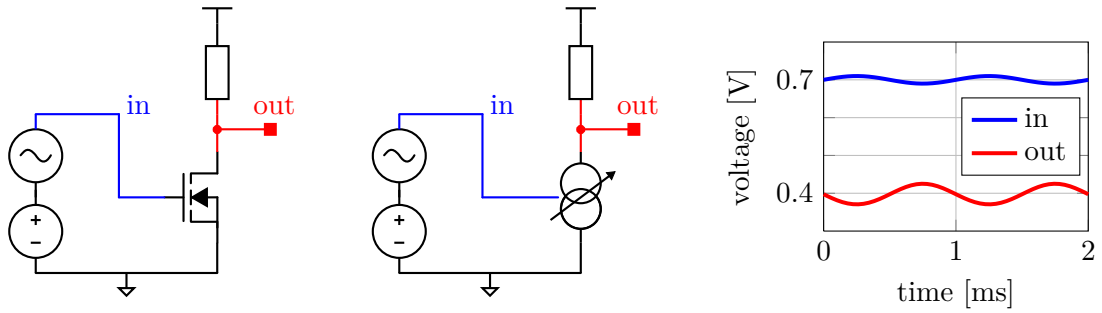


Figure 1.8.: (a) nMOS common source amplifier circuit. Illustration of the (b) current source functionality and (c) the inverting amplifier.

Each transistor offers at least two design parameters: its width W and length L, see bird's view in Figure 1.5 (a). Not only the pin voltages but also these parameters influence the characteristics, e.g. a larger width W results in higher currents. Good and not too complex approximations determining the current I_D in the linear region and saturation region (indicated in Figure 1.7 (b)) are given in Equation 1.1 and 1.2. They are taken from [14] and do not regard any second order effects.

$$I_D \approx \beta_{0n} \frac{W}{L} \left(U_{GS} - U_{th} - \frac{U_{DS}}{2} \right) U_{DS} \qquad U_{GS} > U_{th},\ U_{DS} < U_{GS} - U_{th} \tag{1.1}$$

$$I_D \approx \frac{\beta_{0n}}{2} \frac{W}{L} (U_{GS} - U_{th})^2 \qquad U_{GS} > U_{th},\ U_{DS} > U_{GS} - U_{th} \tag{1.2}$$

The constant β_{0n} is proportional to the charge carrier mobility. In both regions the current is proportional to W: $I_D \sim \frac{W}{L}$. The independence of I_D from U_{DS} in saturation region is expressed by the fact that U_{DS} does not appear in Equation 1.2.

The size of a transistor also changes its gate capacitance, approximately $C_G \sim W \cdot L$. This is relevant because a transistor gate acts as load on a potential previous stage.

The behavior of a **pMOS** is basically similar to the nMOS, but signs (and constants) are changing, see Figure 1.9. A pMOS requires a negative gate voltage U_{GS} and a negative drain voltage U_{DS} to operate. Only these two transistor types are used in this work.

(a) nMOS **(b)** pMOS

Figure 1.9.: Circuit symbol and voltages of (a) nMOS transistor and (b) pMOS transistor.

1.8. Sources of Variance in Integrated Circuits

Manufacturing integrated circuits (ICs) in nanometer scale is a process at technology limits. Structures are only as small as the technology to build them is sufficiently controllable. There are various factors introducing inaccuracies, leading to the fact that every device (transistor or a complete circuit) is a little different from the specification. This is usually unwanted, but indispensable for PUFs. Note that this thesis solely uses standard CMOS technology; materials and processes are not touched.

Variance can be explained physically, like uneven thickness of the oxide or uneven doping in the channel. Variance can also be described by consequences, like varying threshold voltage U_{th} or varying constants like β_{0n} which includes charge carrier mobility. None of these is further explained here. But a categorization is made and necessary for the next chapters. One has to distinguish between manufacturing variances affecting all devices in the proximity equally (global mismatch, named **process** variation in the following), and inter device variation even in close proximity (local mismatch, simply named **mismatch** in the following). The latter is particularly important for our purpose.

Even after manufacturing, the behavior of an IC may differ. It can depend on environmental conditions like temperature or supply voltage. Also chip aging was observed [15], making the IC a time variant system. Furthermore, noise is present in each circuit and prevents exploiting too tiny mismatches. All these are unwanted sources of variance and must be kept in mind when building and applying a PUF. There is only a single environmental influence which is allowed to change the PUF behavior: tampering attempts.

1.9. Outline

This work deals with enhancements of Arbiter PUFs in order to improve their reliability in terms of reproducibility. Therefore the concept of the Arbiter PUF and all its components are investigated and a profound understanding of its functionality is obtained. All examinations are based on applicable simulations. The thesis is segmented as follows.

A suitable simulation approach is presented in chapter 2. It will be outlined how established simulation tools can be operated to analyze Arbiter PUFs, why this can be effortful, and how time can be saved. This knowledge is absolutely necessary for all following examinations.

The components used for Arbiter PUF proposals in literature are analyzed in chapter 3. Implementations known from literature are characterized. It is shown how delay elements and switch elements can be characterized by the distribution of their propagation delay. It is also shown how the arbiter works and which signal and design parameters influence its reliability.

The components are improved in chapter 4, by simple measures. The previous chapter delivered valuable knowledge and a reference case to quantify improvements. More deviation is introduced in delay and switch elements, the arbiter is made less noise-sensitive.

The components are combined to a complete Arbiter PUF in chapter 5 and measures for the quality are evaluated. The newly designed components are also combined to a PUF and the results are compared. Finally chapter 6 summarizes the achievements and provides ideas for future work.

Chapter 2

Simulation Approach

Manufacturing every integrated circuit analyzed in this work is unfeasible due to its cost, alone in terms of time. Simulations are necessary to predict the behavior of a circuit without manufacturing. This chapter introduces the simulation approach used throughout this work.

2.1. Transient Simulation on Transistor Level

Simulating a PUF is not trivial. An analogue circuit simulator is required to perform transient simulations on a transistor level design. Standard CMOS technology is used, for which simulation models and simulators are available. The simulator requires good transistor models which depict the manufactured devices well, especially the mismatch and process variation, and even if devices are used in an uncommon way. The arbiter PUF relies on (small) timing effects and thus requires a simulation of the temporal behavior. This is addressed with a transient simulation.

Our simulations make use to the Cadence Virtuoso SPECTRE Circuit Simulator and industrial manufacturer models of TSMC's 90nm technology.

2.2. Mismatch Simulation

Mismatch is extremely important for PUFs. It is the reason why PUFs work at all. Thus the transistor models need to provide mismatch information. Unlike some other applications, not only the worst case (corner) is relevant, but a realistic distribution is required.

Monte Carlo (MC) Simulations address this issue. The circuit is not simulated once, but several times. Each time realistic parameters are used for every transistor. The parameters are randomly chosen from a distribution deposited in the models. Thus every transistor, even equally designed ones in the same circuit and same simulation run, has its own behavior. The behavior has to be realistic and as random as after manufacturing. Using a sufficiently high number of MC simulations this allows more general predictions than a single simulation with the nominal parameters. For PUFs a nominal run is senseless anyway, because responses would rely on the simulator accuracy rather than PUF behavior.

Various sources of variation were introduced in section 1.8. The simulation environment differentiates between process variation and device mismatch, which can be enabled separately or together. For realistic simulation results of a whole system, it is recommendable to turn both on. However, most of the simulations in this work will not represent a whole system but one of its components. Numerous instances of the simulated component are later combined to a system. In this case process variation should be excluded, because process variation represents global mismatch on the system and may not be regarded for each component separately.

2.3. Noise Simulation

Noise is very important to evaluate the PUFs reliability. It is always present in reality and has to be included in the simulations, too. This again adds requirements on the used simulation environment.

The chosen simulation environment offers the inclusion of transient noise. However, simulating noise is costly. The computation time for a simulation depends strongly on the noise bandwidth, but a higher noise bandwidth promises more realistic results. This issue will be addressed later in subsection 3.4.3.

2.4. Script Control of the Simulator

Uniqueness and Reproducibility are two important PUF properties which were introduced in section 1.2. Analysis of uniqueness requires multiple MC runs. Analysis of reproducibility requires multiple noise runs. These evaluations must take place after every change in the design, even after a single parameter change. And the evaluation may take hours. An environment is required which allows an automation of tasks. It should be flexible enough to allow parameter sweeps and the like.

To achieve this aim the genuine simulation can be controlled by a scripting language. The simulations are prepared in the simulation environment, but executed from the script. Parameters can be changed in advance by the script, the evaluation can take place automatically. A very detailed and practice-oriented explanation is present in appendix B.

2.5. Reducing the Overall Simulation Time

The speed of simulations is a limiting factor. Multiple simulations are required for one PUF evaluation, each simulation is costly, mainly due to noise. Running n = 200 MC runs and m = 200 noise runs already results in a total of $n \cdot m = 40000$ runs per evaluation. Temperature or parameter sweeps contribute additional factors.

In the following a distinction is made between *simulated* time and *simulation* time. The evaluation of a PUF response takes some time, even on a manufactured PUF in reality. This evaluation time defines a lower limit for the *simulated* time. Simulating the same time interval in a circuit simulator takes much longer. This is named *simulation* time.

The simulation time for a certain simulated time interval is necessary and cannot be reduced easily. However, many independent simulations of the same circuit are required. Thus parallelization is possible. Nevertheless it became evident that each simulation produces a huge overhead and most of the time elapses for it. Several actions were taken to reduce the overhead and thus reduce the overall computation time per evaluation.

Time for parameter sweeps on MC simulations can be saved if the simulator input (netlist etc.) is reused. The input is equal anyway except for the sweep parameter, and changing the simulation parameter directly before re-triggering the simulator is much faster than generating its complete input again for every run.

Time for multiple noise runs can be saved if all runs are done within a single simulation. Instead of starting n simulations of length t with different noise seeds, starting one simulation of length n·t avoids the overhead of n-1 runs.

Very detailed explanations are present in appendix B. Considering all mentioned simulation enhancement techniques the computation time is reduced significantly without losing accuracy. The importance is demonstrated by an illustration of the time saving in Figure 2.1. Base is a simulation of a demonstration circuit, so the relation of the times is more relevant than the absolute values. Execution time is the time needed to perform 20 MC runs with 20 noise runs each. The *reference run* performs all 400 simulations sequentially and every run produces overhead. The simulation environment can perform multiple simulations at a time. Therefore only settings need to be adapted, so this is referred to as *quick* enhancements. Further improvements were introduced in this section which mainly reduce the overhead. Combining all measures is summarized as *sophisticated* enhancements. For details see appendix B.

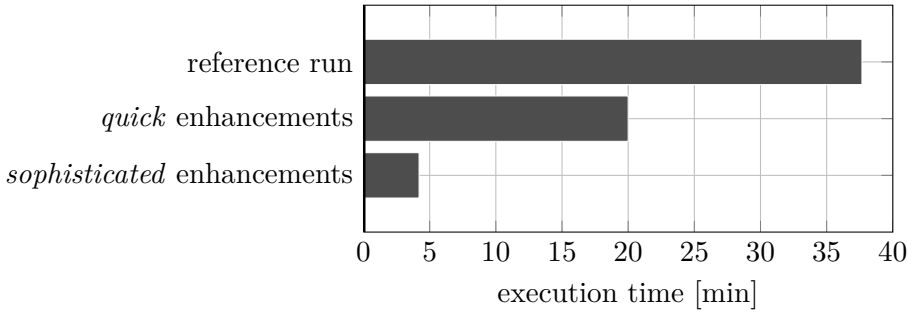


Figure 2.1.: Time savings using enhanced simulation techniques.

Chapter 3

Elementary Circuits

In this chapter some basic circuits that can be used in a PUF, in particular in an Arbiter PUF, are presented. The Arbiter PUF consists of delay elements, multiplexers, and an arbiter. There are various options for each of these components to be implemented. The ones that will be examined here are:

- inverters as delay elements,
- transmission gate multiplexers for choosing a signal path,
- a latch for deciding on the faster of the two paths.

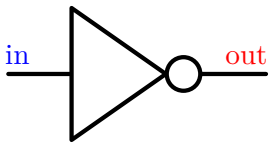
3.1. Inverter

The inverter is a very simple element. Nonetheless it is of great importance for many PUF architectures like the arbiter PUF or the ring oscillator PUF. This is the reason why it is important to understand its function.

Although it is a simple element, it already introduces a few design parameters that influence its behavior, namely the sizes of the two involved transistors. We will deal with that later, for the moment we assume minimum size transistors.

3.1.1. Circuit (Logical)

The basic function of an inverter is to negate its input in terms of a HIGH (near VDD) or LOW (near ground) voltage signal. In digital circuits these signal levels correspond to



in	0	1
out	1	0

Figure 3.1.: Inverter symbol and truth table.

a logical 1 or 0. Due to the limited options the truth table indicating the static behavior is very simple, see Figure 3.1.

3.1.2. Circuit (Transistor Level)

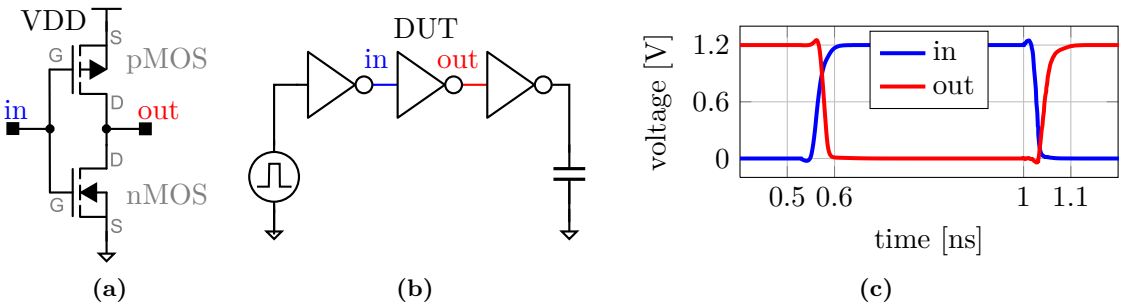


Figure 3.2.: (a) CMOS inverter circuit, (b) test circuit, and (c) transient signal.

The circuit of the CMOS inverter consists of a pMOS and an nMOS transistor, see Figure 3.2 (a). A low signal at the input causes the nMOS to be 'off' and the pMOS to be 'on' which pulls the output high, and vice versa. Assuming a capacitive load no current is necessary to preserve a state.

For our application we are mainly interested in the transient behavior, which is of course not ideal. Traversing a rising edge at the input the initially blocking nMOS starts conducting more and more (I_D) due to the rising U_{GS} and the pMOS in contrast closes. The output node discharges. Due to the transistor characteristics to be passed and the output to be discharged, transitions take some time and edges are not infinitely steep. The path in the test circuit, Figure 3.2 (b), starts with an inverter to get a realistic input for the device under test (DUT) and adds another inverter at the output to simulate a realistic load. Figure 3.2 (c) displays the result of a transient simulation of the test circuit. Note that the regarded in- and output are relative to the single inverter in the middle. Thus the value of the capacitor does not really matter in this setup. The following subsections will establish some measures to characterize the transient behavior.

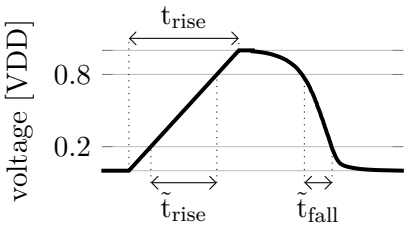


Figure 3.3.: Definition of inverter's fall time and rise time.

3.1.3. Inverter's Typical Slew Rate

As previously mentioned the inverter changes its output voltage gradually. This subsection first introduces a measure for the speed: $\tilde{t}_{fall}$ is defined as the time the signal needs to fall from 80 % to 20 % of VDD. $\tilde{t}_{rise}$ is defined respectively, see Figure 3.3. Assuming a linear progression this corresponds to the rise time t_{rise} or the reciprocal slew rate which is usually defined as $\max(|\frac{dU}{dt}|)$. As we will mostly deal with delays in our context, the time measure is preferred in this paper.

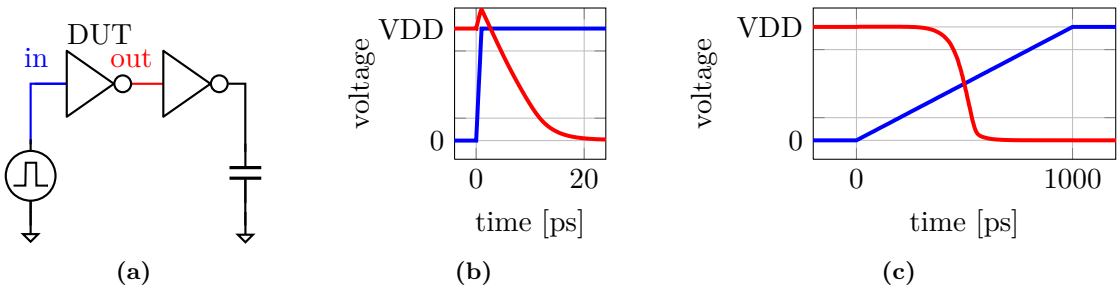


Figure 3.4.: Examination circuit and exemplary waveforms.

Figure 3.4 (a) shows a test circuit to analyse $\tilde{t}_{fall}$ of an inverter. The input slew rate is variable, the output node is self loaded which means another inverter is placed there. The signal curves show examples for a fast and a slow rising input edge. Note the different scaling of the time axis. These examples outline clearly that the inverter has a specific transient behavior rather than a logical one. The output voltage at a specific time cannot be calculated by the input voltage at that time. For the fast rising edge, Figure 3.4 (b), the output starts falling below VDD not before the input fully reached VDD. The output slew rate is slower than the input, $\tilde{t}_{fall} > \tilde{t}_{rise}$. For a slow rising edge, Figure 3.4 (c), the output toggles at about VDD/2 at the input. In this case the output slew rate is faster than the input, $\tilde{t}_{fall} < \tilde{t}_{rise}$. The transient behavior also makes clear why it was necessary to introduce the artificial measures $\tilde{t}_{fall}$ and $\tilde{t}_{rise}$.

Plotting $\tilde{t}_{fall,out}$ against $\tilde{t}_{rise,in}$ this behavior also becomes evident, see Figure 3.5 (a). For short rise times the output slew rate is almost independent of the input slew rate. For longer rise times it increases, but slowly. This means it makes sense to define a typical

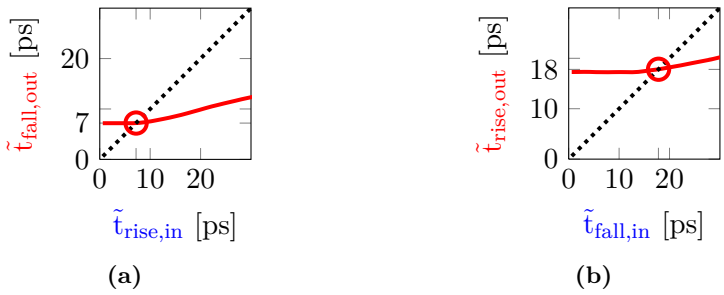


Figure 3.5.: Behavior of inverter's fall time and rise time.

slew rate in an inverter chain (red circle). Within an inverter chain the slew rate where $\tilde{t}_{rise,in} = \tilde{t}_{fall,out}$ will prevail. In this case $\tilde{t}_{fall,typical}$ is 7.2 ps.

The same measure can be applied for a falling input edge resulting in a rising output edge. This is shown in Figure 3.5 (b). It is remarkable that $\tilde{t}_{rise,typical}$, which is 18 ps, is a factor of 2.5 higher than $\tilde{t}_{fall,typical}$. The typical rising edge is slower than the falling edge. This results from a higher resistance of a same sized pMOS transistor due to the lower charge carrier mobility. This is a reason why normal inverters are built with a larger pMOS transistor in comparison to the nMOS.

The gained insight predestines the inverter to refresh edges to a defined slew rate if needed for a subsequent stage. Note that the absolute slew rate is still dependent on the transistor parameters and the load. This will be examined in chapter 4.

3.1.4. Inverter's Delay

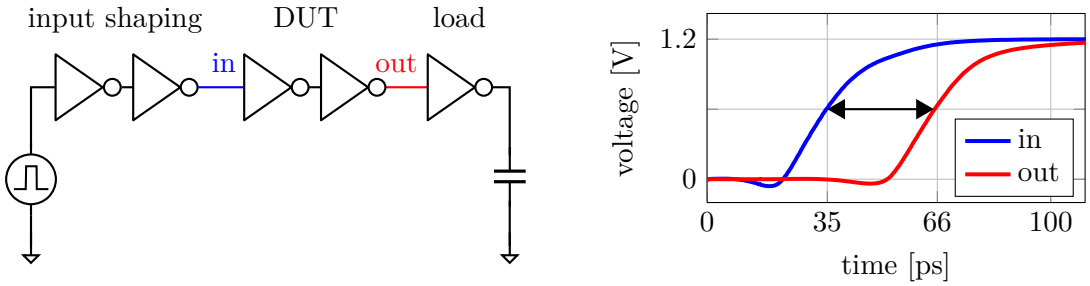


Figure 3.6.: Examination circuit and illustration of the typical delay of two inverters in series.

This subsections deals with the delay, which is the probably most important property of an inverter in the PUF context. A side effect of the previous subsection is the insight that defining a delay of a single inverter might be difficult. The test circuit in Figure 3.6 addresses this problem by examining an inverter chain. The first inverters are used to generate a realistic input signal, the last inverter serves as a realistic load. By evaluating

the delay between two inverters instead of one the input and output signal curve are very similar, but delayed. The delay is defined as the time between the input crossing VDD/2 and the output crossing VDD/2. A typical delay value for minimum size transistors in our technology is 32 ps.

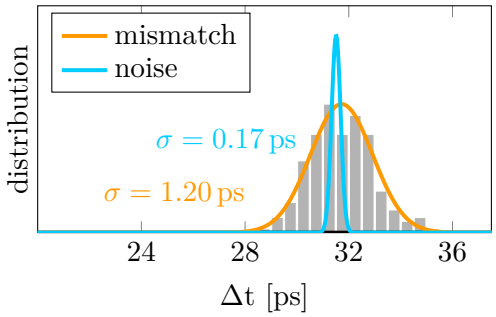
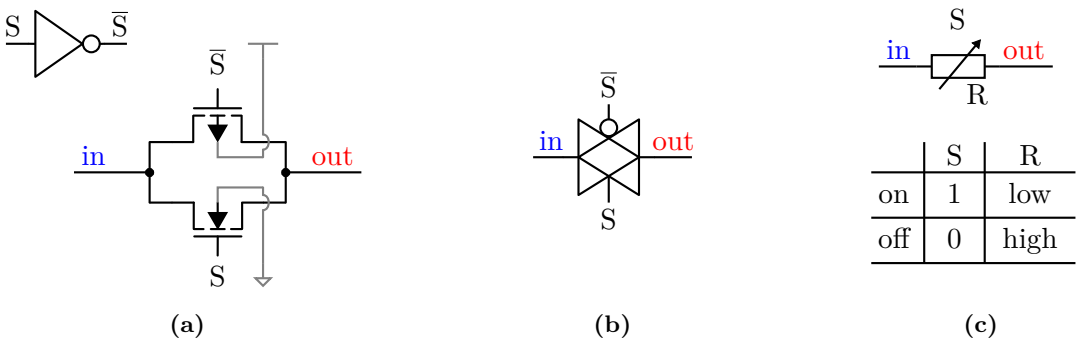


Figure 3.7.: Distribution of the inverter delays.

What is even more important than the delay is the deviation of delays. Equally designed transistors will behave differently due to mismatch caused by manufacturing, even when placed near each other in the layout. This issue is addressed by a monte carlo simulation. Figure 3.7 displays the resulting distribution of delays in orange. It shows a standard deviation of 1.2 ps, which is not neglectable in relation to the mean.

Another fact that results in varying delays is noise, which causes even a specific manufactured inverter to exhibit slightly different delays every time. This is addressed by simulating transient noise. The resulting distribution is also shown in Figure 3.7 in blue. For our applications it will be important that the deviation introduced by noise is much less than the deviation introduced by manufacturing. Note that the values depend on the transistor parameters, which are minimum size here. Their influence will be examined in chapter 4.

3.2. Transmission Gate



	S	R
on	1	low
off	0	high

Figure 3.8.: Transmission Gate circuit, symbol, and resistance table.

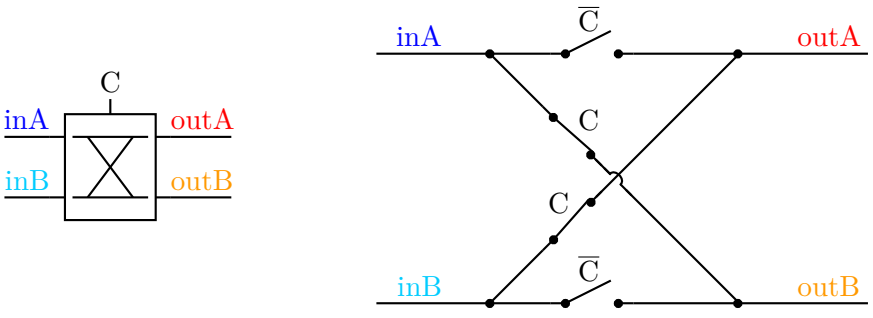
A transmission gate is another useful basic element. It acts like a switch and can be turned on and off by an electric signal.

Figure 3.8 (a) displays the circuit. It consists of an nMOS and a pMOS transistor in parallel. For operation two complementary signals S and $\overline{S}$ are needed. Assuming S is logic high and $\overline{S}$ is logic low the transmission gate is conducting. The nMOS passes logic lows well and the pMOS logic highs well. Rather than a digital gate the transmission gate also allows to propagate analog signals. Although its resistance varies with the input voltage, its behavior resembles more a resistor rather than a switch, illustrated in Figure 3.8 (c). Its resistance can be either low or very high, but never zero. The symbol, Figure 3.8 (b), resembles a buffer working in both directions. A detailed introduction and applications can be found at [16, p. 375ff].

3.3. Transmission Gate Switch

An arbiter PUF relies on elements that modify the two signal paths. Each challenge bit (C) stands for a path decision. For this purpose switch elements are needed.

3.3.1. Circuit (Logical)



(a) **(b)**

Figure 3.9.: Switch element symbol and function.

A switch element symbol is displayed in Figure 3.9 (a). InA is either propagated to outA or outB; inB to the other output. This behavior is illustrated in Figure 3.9 (b).

3.3.2. Circuit (Implementation)

The early PUF proposed by Lee et al. in 2004 [9] uses transmission gate multiplexers to achieve this. This is illustrated in Figure 3.10 for one output, the same applies to the other

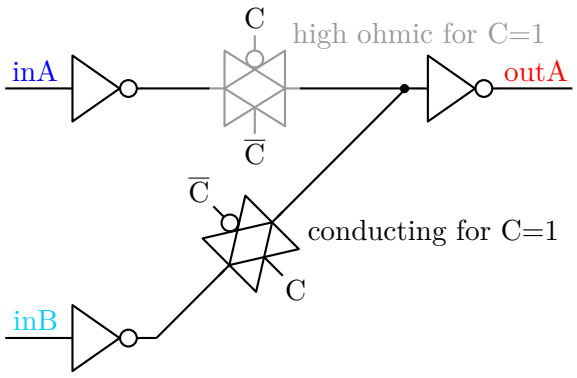


Figure 3.10.: Transmission Gate Multiplexer circuit.

one (left out for clarity). Inverters at the input drive the transmission gates; inverters at the output load the transmission gate and drive the output node.

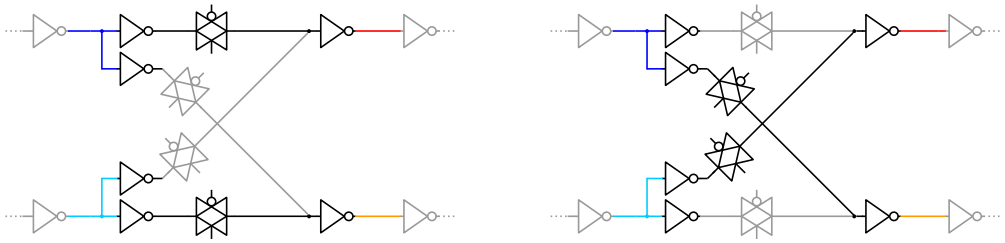


Figure 3.11.: Switch element path illustrated for C=0 and C=1.

Figure 3.11 shows the two paths and illustrates which circuit elements are affected for both options. It can be assumed that both paths are driven by an inverter of the previous stage (gray) and the output has to drive an inverter of the following stage (gray). Inside, each two corresponding transmission gates are switched on or off in the demonstrated way (black or gray). Note that changing the path is equivalent to interchanging the outputs as long as mismatch is not regarded.

3.3.3. Switch's Delay

Figure 3.12 displays a test circuit to analyze the delay of the switch circuit. Therefore examining one path is sufficient due to the symmetric structure. At the DUT input two inverters have to be driven by a single inverter. The DUT output has to drive a single inverter. Inside the DUT, one path is conducting and the other one is high ohmic. Nevertheless the second path must not be neglected. In a complete switch element the shown circuit exists twice, once per path. Thus a mismatch simulation on the DUT makes sense.

A mismatch simulation based on the test circuit in Figure 3.12 will allow statements on the difference of the delays of two paths that are active simultaneously. It is only an

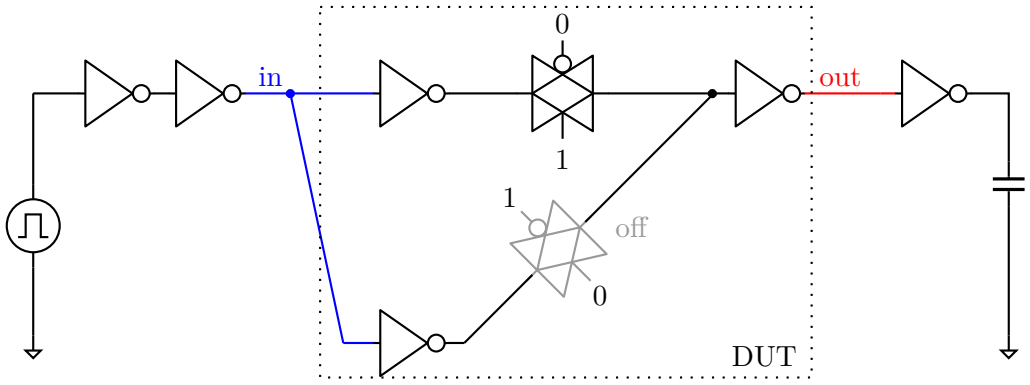


Figure 3.12.: Switch element delay test circuit.

approximation for the comparison of two paths that feed the same output. In this case the second inverter is shared by both paths and only the first inverters and the transmission gates mismatch. Regarding the only two valid switching options illustrated in Figure 3.11 it becomes clear that this case can be neglected in our context, because these paths are never compared directly to each other during operation.

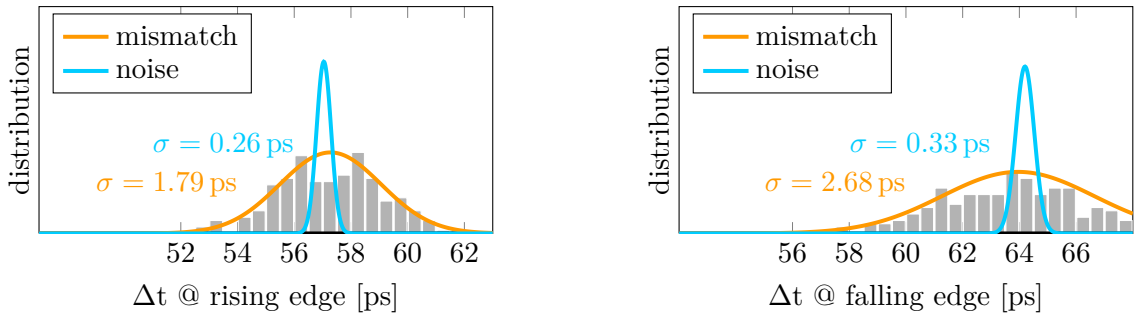


Figure 3.13.: Distribution of the delays of the test circuit.

Figure 3.13 depicts the simulation results of the presented switch element test circuit. The propagation of a rising and a falling edge are examined separately because the transmission gates inside the DUT may influence the results differently. As usual we use minimum size transistors.

In general the test circuit and the normal inverter chain, which was examined in section 3.1, display similar behavior. Both provide a delay in the range of tens of picoseconds, a mismatch standard deviation in the range of some picoseconds, and a noise standard deviation in the sub picoseconds range. This demonstrates the importance of the switch circuit for the PUF. The switch circuit cannot be neglected and must not be taken for ideal.

In the following the switch behavior is analyzed in detail. It is remarkable that a falling edge (Figure 3.13 right plot) progresses slower than a rising edge (Figure 3.13 left plot). And it can be observed that the delay of the falling edge varies far more compared to the rising edge, with a standard deviation of 2.7 ps instead of 1.8 ps, or otherwise expressed

50 % more. It also turns out that the deviation caused by noise is a bit larger for the falling edge but considering the much higher mismatch deviation it is still in the same range than the corresponding one of the rising edge.

The transmission gate was introduced in section 3.2 as a kind of resistor. An investigation is made to check whether the behavior of the transmission gate can be approximated by a resistor to get a simplified model. A modified test circuit is given in Figure 3.14. The closed transmission gate is replaced by a 100 GΩ resistor, the conducting transmission gate is replaced by a 77 kΩ resistor. The latter value was chosen to match the mean propagation delay of the unmodified circuit, which is 57 ps for the rising edge. The rising edge was chosen to match, because a rising edge is propagated during the PUF operation.

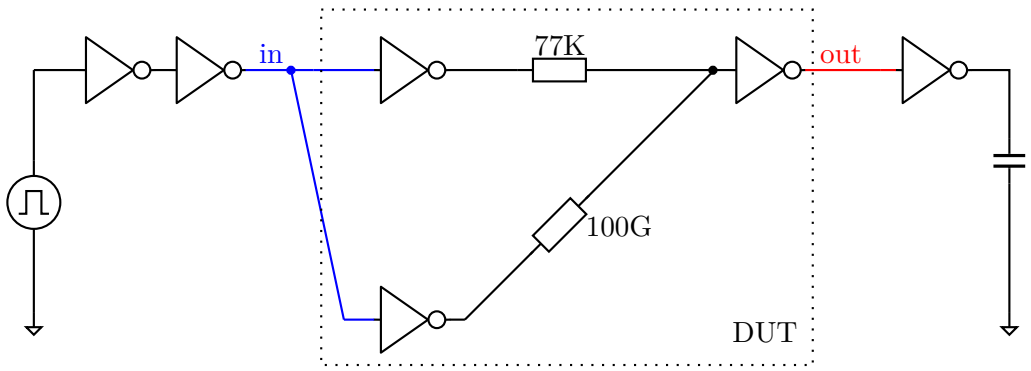


Figure 3.14.: Switch element delay test circuit with resistors instead of transmission gates.

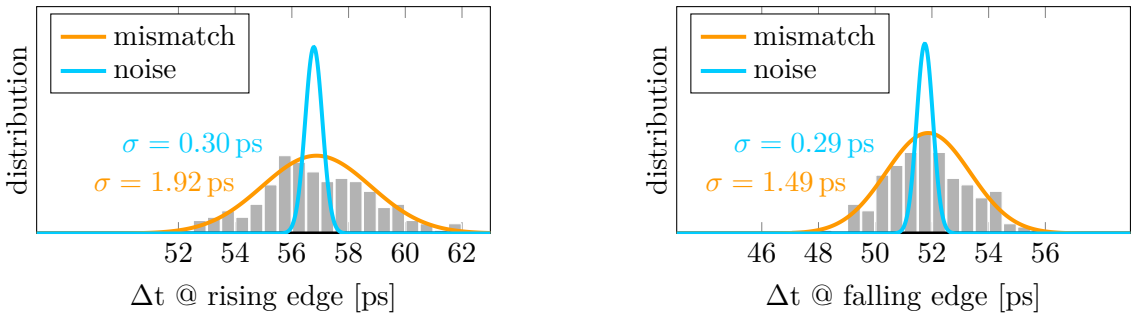


Figure 3.15.: Distribution of the delays of the test circuit. Transmission Gates are replaced by resistors.

Simulation results for the modified test circuit are displayed in Figure 3.15. They have to be compared to the results of the original test circuit in Figure 3.13. The mean delay of the rising edge is equal by design. The deviation of the delay of the rising edge due to mismatch is slightly more than for the original circuit, although the ideal resistor does not introduce additional deviating elements in contrast to the transmission gate. The behavior of the falling edges (right plots in Figure 3.15 and Figure 3.13) differ significantly for both circuits. Neither the mean delay nor the deviation of the delays are similar. This leads to the insight that a simple resistor is not sufficient to approximate the behavior of

a transmission gate within the given circuit. Further investigation requires more complex models where for example the voltage dependence of the transmission gate's resistance is regarded. This contradicts the aim for a simple model and is thus not further investigated.

3.4. SR NAND Latch

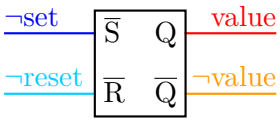
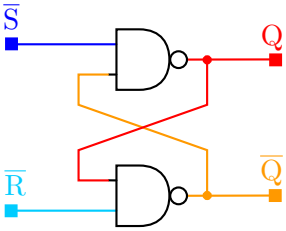


Figure 3.16.: $\overline{SR}$ latch.

A latch is a circuit that has two stable states and can thus store 1 bit of information. A latch is not clocked, in contrast to a flip-flop. Figure 3.16 shows the symbol of a set-reset latch (SR latch) with active-low inputs ($\overline{SR}$ latch).

Due to the asynchronous operation the latch can also act as an arbiter. Therefore the two inputs $\overline{S}$ and $\overline{R}$ are treated as equal competitors, the output Q indicates which input arrived first. This ability was formerly used in clock synchronization circuits (according to [16, p. 383]) and is very important for arbiter PUFs. The following subsections will deal with the function and some properties of the $\overline{SR}$ NAND latch, a $\overline{SR}$ latch built of NAND gates.

3.4.1. Circuit (Logical)



$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	
1	1	Q	$\overline{Q}$	keep state
0	1	1	0	set
1	0	0	1	reset
0	0	1|?	1|?	forbidden

Figure 3.17.: $\overline{SR}$ NAND latch and truth table.

The $\overline{SR}$ NAND latch consists of two cross-coupled NAND gates, see Figure 3.17. A NAND gate implements the logic function $Y = \overline{A \wedge B}$, where Y is the output and A and B are the inputs. *Consider the case when* $\overline{R}$ *is high and* $\overline{S}$ *is low. Forcing* $\overline{S}$ *low causes* Q *to go high. Since* $\overline{R}$ *is high and* Q *is high, the* $\overline{Q}$ *output is low. Now consider the case when both* $\overline{S}$ *and* $\overline{R}$ *are low. Under these circumstances, the latch's outputs are both high.* [16]. When operating as storage, this combination is forbidden. It violates the trivial relation $Q = \neg\overline{Q}$ and the state after releasing $\overline{S}$ and $\overline{R}$ simultaneously is not defined. When operating as arbiter, this state is the starting point for a decision due to its instability. Assuming ideal

gates, a short delay Δt between the rising signals at $\overline{S}$ and $\overline{R}$ is sufficient to decide for the corresponding stable state which can then be read out safely.

3.4.2. Circuit (Transistor Level)

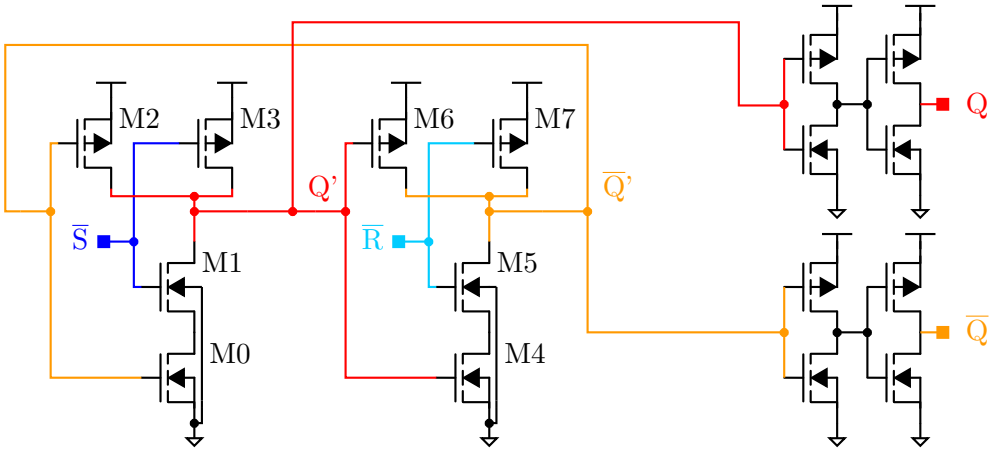


Figure 3.18.: CMOS $\overline{SR}$ NAND Latch circuit with output buffers.

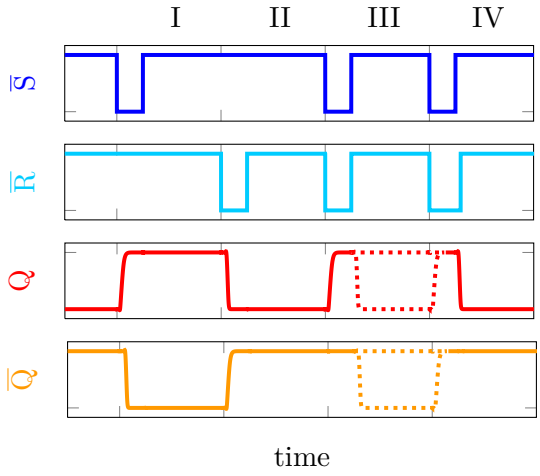


Figure 3.19.: Signal curves representing typical latch applications.

Figure 3.18 shows the transistor level circuit of the introduced latch. Its function is based on the two equal NAND gates (M0-M3 and M4-M7). Additionally buffers are added to the internal latch outputs (right). Although it could be neglected for simulation, every subsequent circuit using the value of Q' or $\overline{Q}$' loads the latch and may influence the latch's behavior. The given buffers add small loads, too, but they load Q' and $\overline{Q}$' equally. Thus no input is preferred. Unless noted otherwise all transistors are minimum size as usual.

Figure 3.19 shows some signal curves to demonstrate the behavior of a latch as arbiter. Each section has a duration of 2 ns.

I Pulling $\overline{S}$ low stores the value 1 in Q and 0 in $\overline{Q}$. The visible delay results from the settling time of the latch and the delay of the output buffer. Releasing $\overline{S}$ keeps the values.

II Pulling $\overline{R}$ low stores the value 0 in Q and 1 in $\overline{Q}$. Releasing $\overline{R}$ keeps the values.

III Pulling both $\overline{S}$ and $\overline{R}$ low and releasing them exactly at the same time results in an undefined behavior (as long as mismatch is not regarded). Q can settle for 1 or 0 randomly (dotted lines).

IV When releasing $\overline{S}$ and $\overline{R}$ successively the behavior is again defined (at least as long as mismatch and noise are not regarded): The input released last will dominate and set or reset the value Q.

In Section IV the latch acts as arbiter deciding for the order of the input rising edges. This behavior will be explained now in greater detail using the circuit from Figure 3.18. Starting point is the unstable state where both $\overline{S}$ and $\overline{R}$ are low. M3 and M4 are on ($|U_{GS}| \gg |U_{th}|$), while M1 and M5 are off ($|U_{GS}| \approx 0$). The gates' outputs Q' and $\overline{Q}$' are high. Now assume $\overline{S}$ is rising first. At the left gate M0 is on and M2 is off because $\overline{Q}$'=1. Taking M0 as shorted and neglecting M2 leaves an inverter consisting of M1 and M3. Rising its input lowers its output Q'. This affects the right gate where Q' is an input. Now M6 is on and M4 closes, which assures that the output $\overline{Q}$' remains high. This state is now stable even when $\overline{R}$ is released, too.

To achieve reliable decisions for short time differences between the rising edge of $\overline{S}$ and $\overline{R}$ it is important that the internal state flips before the second input also arises. Investigations how this behavior can be improved are made in chapter 4.

3.4.3. Input Sensitivity (Influence of Noise)

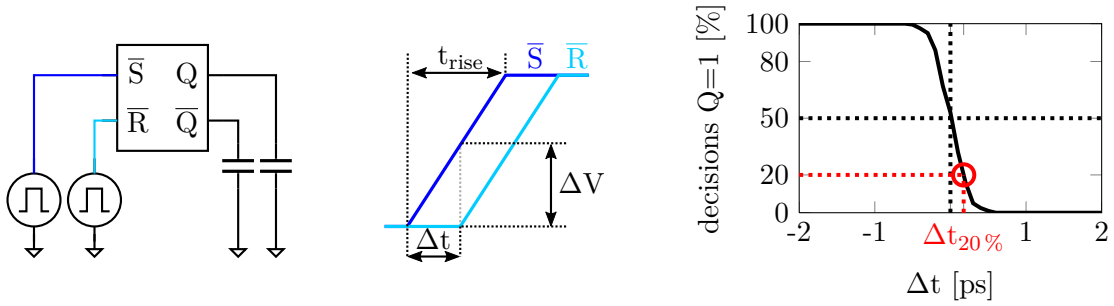


Figure 3.20.: (a) Latch test circuit, (b) test input signals, and (c) decision behavior.

A manufactured latch is not ideal and one reason for this is noise. This section will deal with the influence of noise on the arbiter decisions. All predictions in this section are

based on the Transient Noise Simulation of the SPECTRE Simulator using a Maximum Noise Frequency of 100 GHz.

Figure 3.20 (a) shows the test circuit. The input of the latch can be defined arbitrarily. Usually the inputs will resemble inverter edges. For the test the edges are approximated as shown in Figure 3.20 (b). The signals rise linearly from 0 to VDD within a rise time t_{rise}. The interval Δt between the two inputs is expected to be the most important parameter. ΔV is introduced as the voltage difference between the two signals.

Figure 3.20 (c) displays the decision ability of the latch for a given rise time $t_{rise} = 10\,ps$, which is a reasonable. The ideal latch decision function is a kind of sign function. Every $\Delta t < 0$ evaluates to 1, every $\Delta t > 0$ evaluates to 0, $\Delta t = 0$ is not decidable and should result in a decision rate of 50 %. Unsurprisingly the probability for a correct response decreases in the range around $\Delta t = 0$. But the image also allows some quantitative proposition. In this case (remember we made assumptions on the input signals and transistor sizes) that means: Avoiding the interval $|\Delta t| < 1\,ps$ makes decisions very reliable against the influence of noise. The plot content can be summarized to a single number which allows statements on the quality of the decisions. Therefore $\Delta t_{20\,\%}$ is introduced. It is defined as the Δt where 20 % of the runs decide towards Q=1.

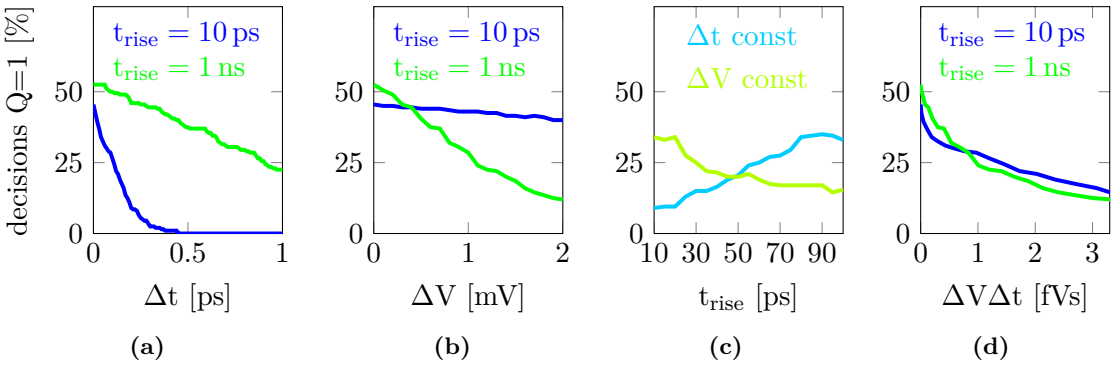


Figure 3.21.: Influence of test signal parameters on latch decisions.

The following investigation demonstrates that Δt is not the only parameter the arbiter decision is sensitive to. Figure 3.21 (a,b,c) plot the dependence on the three parameters introduced in Figure 3.20 (b). The ideal value for decisions Q=1 is 0, the worst expectable case is 50 %, which corresponds to a completely random response for Q=1 or Q=0. In brief: the lower the better. 200 decisions per data point were evaluated, which is enough for a qualitative statement even if the plots may appear noisy.

The blue curve in the Δt plot Figure 3.21 (a) is exactly what was already shown in Figure 3.20 (c), but only for $\Delta t > 0$. Due to the symmetric structure of the arbiter it is sufficient to regard one half. The green curve of the same plot reveals the same under the assumption of a longer rise time t_{rise}. It is clearly visible that the decisions become less reliable. For an arbiter this means the inputs would have to differ a lot more in Δt to achieve the same level of reliability. This is also what the t_{rise} plot Figure 3.20 (c) exposes. The blue curve demonstrates that for an increasing t_{rise} and constant Δt the rate of false

responses increases. This is an important insight: The rise time of the input signals has a big influence on the arbiter behavior.

This insight leads to the question what the input is sensitive to instead and if there are better parameters than Δt. One proposition is ΔV. As can be seen in Figure 3.20 (b) on page 28, ΔV is another kind of distance between the two input signals. The second plot Figure 3.21 (b) examines this proposition. Again the result depends much on t_{rise}. But this time the other way, as the green curve in the third plot Figure 3.21 (c) outlines. For a constant ΔV the false response rate decreases for increasing t_{rise}. This behavior can also be explained regarding the circuit. The inputs always differ by the same voltage, but the rise time corresponds to the time interval the arbiter is exposed to this difference. A longer exposure time increases the chance for the signal to influence the decision in relation to noise.

Finally a measure was found to indicate the response rate without direct dependence on the rise time. Figure 3.21 (d) plots the response rate against $\Delta t \cdot \Delta V$. It is plausible that the two curves indicating different rise times (factor 100) show the same behavior. The recorded data can easily be converted to this format using the relation 3.1

$$\frac{\Delta V}{\Delta t} = \frac{VDD}{t_{rise}} \tag{3.1}$$

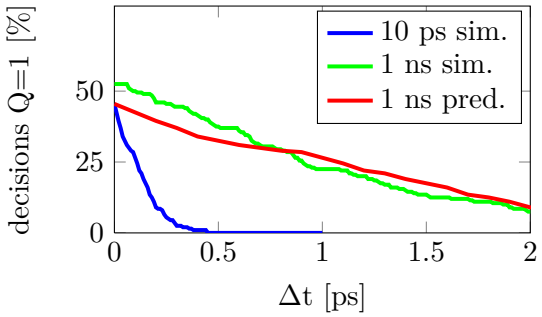


Figure 3.22.: Latch behavior prediction for a different input slew rate.

This insight is useful to predict the latch decision behavior in dependence of Δt for a given rise time, even if the original curve was recorded at another rise time. This saves simulation time because realistic noise simulations are very costly and it was shown before that the rise time influences the behavior significantly. Figure 3.22 extends Figure 3.21 (a) by the prediction (red) of the behavior for $t_{rise} = 1\,ns$ based on the values simulated for $t_{rise} = 10\,ps$ (blue). The prediction is very similar to the simulated values (green). This underlines that the proposed relation is suitable. The calculation derived from equation 3.1 is developed below.

The following calculation converts a latch decision curve recorded at a given rise time to another rise time. Therefore the time value Δt of each value pair can be converted as following. Knowing a recorded (simulated) value pair (Δt_{rec} and its decision rate) it was shown that the product of Δt and ΔV remains constant (equation 3.2) for varying t_{rise},

and ΔV can easily be calculated using expression 3.1. The final calculation is developed in equations 3.3 below.

$$X = \Delta t \cdot \Delta V = \Delta t_{rec} \cdot \Delta V_{rec} \tag{3.2}$$

$$\begin{aligned} \Delta t &= \frac{X}{\Delta V} \\ \Delta t &= \frac{X}{1} \frac{t_{rise}}{VDD \cdot \Delta t} \\ \Delta t \cdot \Delta t &= \frac{X \cdot t_{rise}}{VDD} \\ \Delta t &= \pm\sqrt{\frac{X \cdot t_{rise}}{VDD}} \end{aligned} \tag{3.3}$$

The new value pair consists of the calculated Δt using the final formula of 3.3 and the unchanged decision rate. There t_{rise} is the rise time which the curve has to be adapted to. The sign of Δt is the same as the original Δt_{rec}.

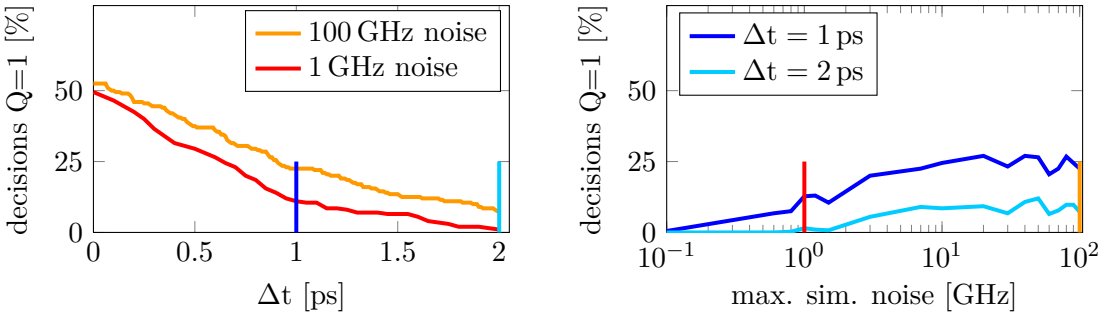
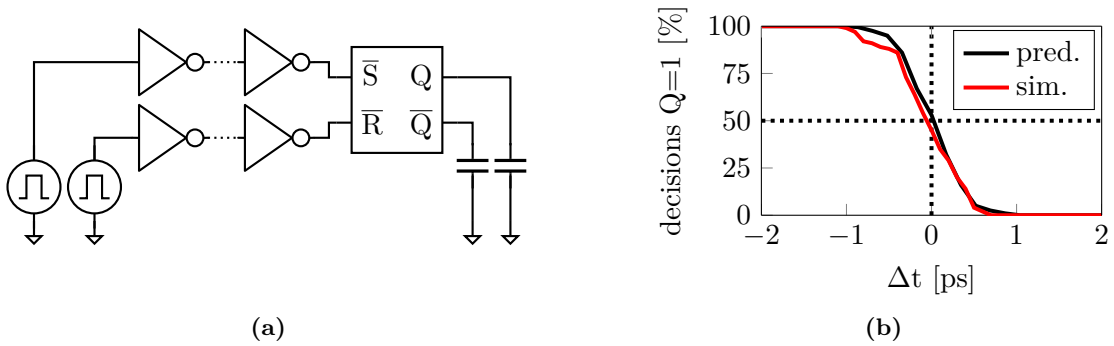


Figure 3.23.: Influence of noise simulation parameters.

Some notes on the noise simulation parameters are attached in the following. The simulation time grows a lot for high maximum noise frequencies to be simulated. Figure 3.23 (a) shows the effect if noise is simulated at reduced maximum noise frequency to save simulation time. At no delay ($\Delta t = 0$) there is almost no difference, both decision rates are around 50 %, which is the ideal value. But at $\Delta t = 1$ ps the decision rate has dropped by half, at $\Delta t = 2$ ps the error seems to have vanished while it is still significant in the more realistic simulation. Figure 3.23 (b) illustrates the error rate in dependence of the maximum noise frequency for the two mentioned Δts. It demonstrates that in general using a lower maximum noise frequency results in a illusively lower error rate. But it also demonstrates that the error rate does not change significantly in the range of 10 to 100 GHz. Unfortunately these values may not be valid in general and have to be proven individually.

Up to this point the input signals were assumed to be linearly rising edges. Figure 3.24 (a) introduces a more realistic scenario. The inputs are real inverter edges. As it was demonstrated in section 3.1 inverter edges are a bit different, they look smoother and the slew

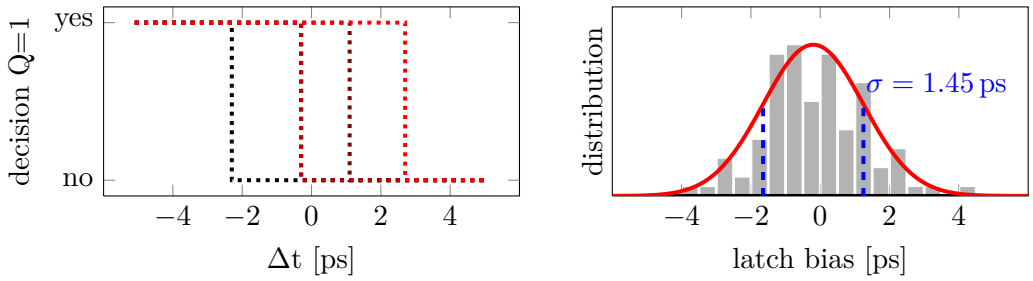


(a) (b)

Figure 3.24.: (a) Realistic test circuit and (b) comparison to prediction.

rate is intrinsically set. It was outlined that $\tilde{t}_{\text{rise,typical}}$ (raise from 20 % to 80 % of VDD) is 18 ps in a minimum size inverter chain, which equivalents a rise time (from 0 to VDD) of about 30 ps. The latch as a load can also be seen as an inverter in this case because every input consists of an nMOS and a pMOS gate without any extra load. Figure 3.24 (b) compares the predicted decision behavior assuming artificial rising edges at the typical rise time (pred., black) with the decision behavior using realistic edges (sim., red). This is the evidence that the artificial edges, which were introduced to simplify investigations, are a very good model for realistic edges.

3.4.4. Input Bias (Influence of Mismatch)



(a) (b)

Figure 3.25.: Decision behavior and distribution of the Bias.

The last subsection dealt with the influence of noise. Another artefact that makes decisions unideal results from mismatch. Mismatch causes each transistor to behave slightly different than its specification. Figure 3.25 (a) shows the decision curve for a few different monte carlo runs. The simulation is done without noise. That implies that there is a hard switching point where the arbiter decides for 1 below or for 0 above. It becomes evident that there is a bias in the range of certain picoseconds for these examples and they differ around 0. Figure 3.25 (b) shows a histogram and a fitted Gaussian distribution for 100 monte carlo runs.

It turns out that the bias can be quite significant. For the test circuit a deviation σ of 1.45 ps was calculated. A bias at the arbiter can result in a bias in the PUF response, but it remains unchanged for an arbiter instance at constant environmental conditions. It depends on the application whether the bias has a disturbing effect.

Chapter 4

Enhanced Arbiter PUF Components

This chapter deals with enhancements of the standard Arbiter PUF. Therefore the outlined characteristic properties of its components are taken into account.

Every transistor offers design parameters which can be modified at design time. Namely the width W and the length L, which define the size of the transistor, are of interest. More precisely, its deviations given by the manufacturer models and the influence of transient noise are deciding. This leads to the obvious parameter tuning investigation that will be performed in this chapter.

4.1. Inverter

The inverter and the inverter chain were introduced as very simple components in section 3.1. Nonetheless every inverter offers a few parameters which can be modified individually:

- L_n: length of the nMOS,
- W_n: width of the nMOS,
- L_p: length of the pMOS,
- W_p: width of the pMOS.

The supply voltage VDD is also variable.

If not noticed otherwise the device under test is an inverter chain consisting of 2 inverters while the parameters refer to all inverters of the complete inverter chain equally, includ-

ing previous and following stages. Monte Carlo Variations and Transient Noise are only applied to the devices under test. W without index addresses both W_n and W_p, the same applies to L. Figure 4.1 (a) displays the test circuit already known from Figure 3.6.

The main characteristic property of an inverter chain is its delay. More precisely

- μ: the mean delay
- σ_{mc}: the standard deviation of the delay due to mismatch
- σ_{noise}: the standard deviation of the delay due to noise

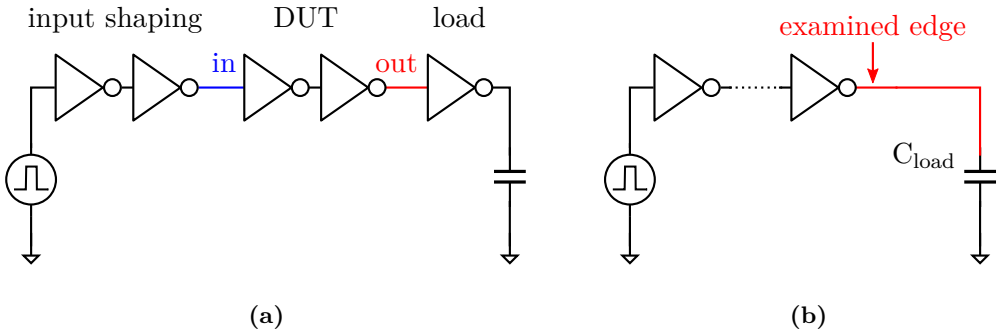


Figure 4.1.: Inverter chain's test circuits for (a) delay and (b) edge examination.

Furthermore it was outlined that the slew rate is an important characteristic property. It can be expressed by the values

- $\tilde{t}_{rise}$: the rise time from 20 % to 80 % of VDD
- $\tilde{t}_{fall}$: the fall time from 80 % to 20 % of VDD

For the determination of these values a constant load is assumed, see Figure 4.1 (b).

A matrix considering the influence of every parameter on every characteristic property was created and is present in the appendix A.1. In the following some outstanding relations are pointed out.

4.1.1. Influence of Width W on Delay

Figure 4.2 displays the influence of the transistor width. μ (left plot) remains more or less constant. This results from the fact that the inverters are loaded by equally sized inverters. This means the load grows equally to the driver strength of the transistors. Much more interesting is the behavior of the deviation, which is illustrated by the standard deviation σ (right plot). The deviation caused by mismatch drops to 48 % for 1 μm devices in

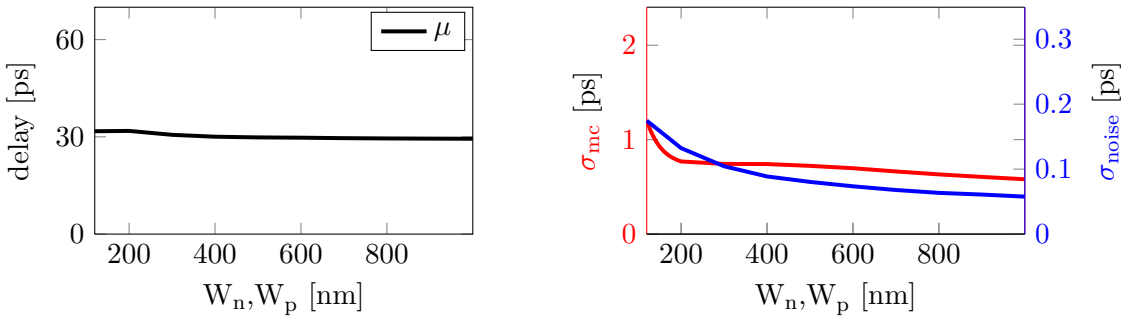


Figure 4.2.: Influence of inverter size on delay and its deviation.

comparison to the minimum size device. At the first sight this is an unwanted effect. But the deviation caused by noise drops to 33 %. This means the deviation due to noise drops more than the deviation caused by mismatch. Exploiting this effect promises more reliable responses by increasing the transistor size.

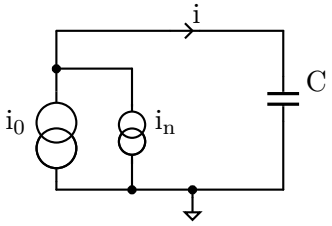


Figure 4.3.: Illustration circuit for inverter operation at a falling edge.

The deviation of mismatch behaves as expected. Matching becomes better for larger devices. The behavior of the noise deviation requires further explanation. Therefore a very simple model is used to model the charging procedure within an inverter chain. A current source loads a capacitor, see Figure 4.3. This illustrates an inverter stage (current source) driving another inverter stage (capacitor). The current source is modeled as an ideal current source and a noise current source in parallel. The capacitor voltage is the output signal and is determined by $U(t) = \frac{1}{C}\int_0^t i(\bar{t})d\bar{t}$, assuming $C_{t=0} = 0$. Splitting the equation to regard the two current sources results in Equation 4.1:

$$U(t) = \frac{1}{C} \cdot i_0 \cdot t + \underbrace{\frac{1}{C}\int_0^t i_n(\bar{t})d\bar{t}}_{\text{noise}} \tag{4.1}$$

To determine the mean delay the noise term is ignored and the equation is reformed to $t = \frac{C}{i_0} \cdot U$. For increased transistor sizes it can be assumed that $C \sim W$ (gate capacitance goes linearly with W) and $i_0 \sim W$ (drain current goes linearly with W, see Equation 1.1 and 1.2 on page 9). This results in $t \sim \frac{W}{W} \sim 1$, which means the average time to reach a certain voltage is independent from W.

To determine the deviation the noise term in Equation 4.1 has to be taken into account, too. The noise current i_n is assumed to be Gaussian distributed. Then the integral expresses a

Wiener process. It is typical for a Wiener process that its variance goes linearly with time. In our case this can be ignored because the time was already proven to remain constant in the last paragraph. Also C still behaves like $C \sim W$. But i_n behaves different from i_0: $i_n \sim g_m \sim \sqrt{i_0} \sim \sqrt{W}$. The overall impact of W is given in Equation 4.2:

$$\sigma_{\text{noise,voltage}} \sim \frac{1}{C} \cdot i_n \cdot \sqrt{t} \sim \frac{1}{W} \cdot \sqrt{W} \cdot 1 \sim \frac{1}{\sqrt{W}} \tag{4.2}$$

This deviation refers to the voltage over the capacitor. It still has to be transformed to a deviation in time. In this case this is trivial, because is was shown that the slope of the output is independent from W. Thus it can be expressed with Equation 4.3:

$$\sigma_{\text{noise}} \sim \sigma_{\text{noise,voltage}} \sim \frac{1}{\sqrt{W}} \tag{4.3}$$

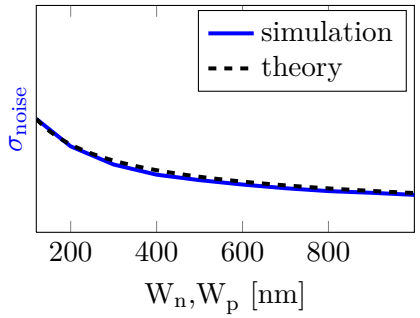


Figure 4.4.: Evaluation of the theory explaining the noise behavior.

The theory is evaluated in Figure 4.4. It can be observed that this result fits the simulation results well, despite the simple model.

4.1.2. Influence of Width W_n on Delay

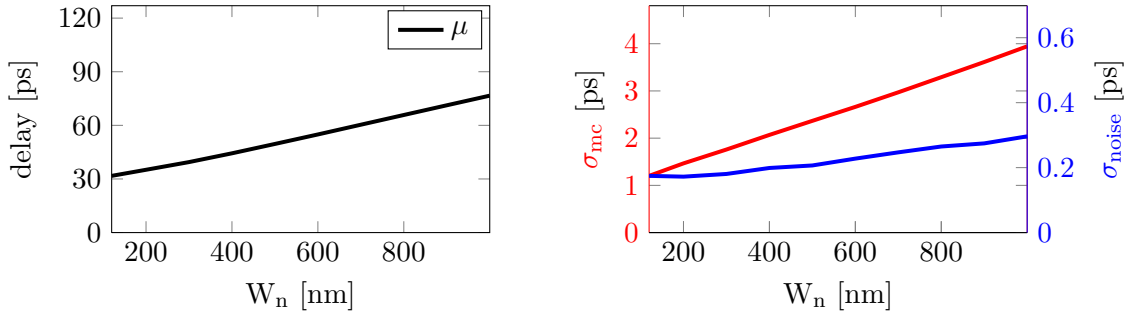


Figure 4.5.: Influence of inverters's nMOS width on delay and its deviation.

Figure 4.5 displays the influence of increasing W_n while leaving W_p at minimum size. In this case the delay increases (left plot). Regarding the values at 120 nm and 1 μm it

increases by a factor of 2.4. The influence of mismatch and noise also increases. But for the same parameter change σ_{mc} grows by a factor of 3.3 while σ_{noise} only grows by 1.7. This means the influence of mismatch grows faster than the influence of noise, which makes it desirable to use a larger $\mathrm{W_n}$ if the increased delay is acceptable. Note that this is contrary to a usual digital inverter design where the pMOS is designed larger.

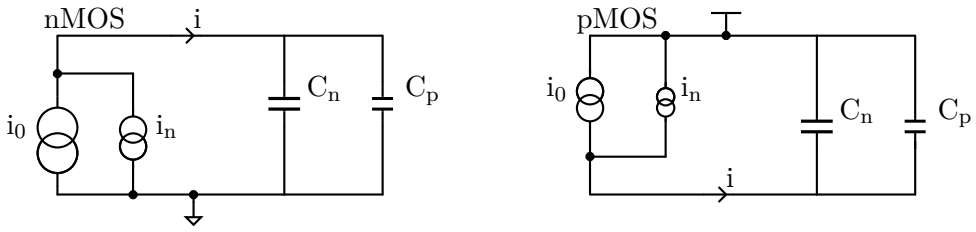


Figure 4.6.: Illustration circuit for asymmetric inverter stage.

To explain these observations the examination can be split. Rising and falling edge are regarded separately. The model is extended to depict the fact that only one transistor is increased. Figure 4.6 illustrates this. At first the increase of the delay is described. For the falling edge the nMOS determines the current, see the left schematic. In this case $\mathrm{i_0} = \mathrm{i_{nMOS}} \sim \mathrm{W}$ and $\mathrm{C} = \mathrm{C_n} + \mathrm{C_p} \approx \mathrm{C_n} \sim \mathrm{W}$. This is almost the same case as before. Current and capacitance grow equally, the time for the falling edge remains approximately constant, $\mathrm{t_{falling}} \sim 1$. For the rising edge the pMOS determines the current, see the right schematic. The current remains constant while the load itself grows. It is immediately clear that this results in a slower progress. More precisely, the time t to reach a certain voltage goes linearly with C and thus approximately linear with $\mathrm{W_n}$: $\mathrm{t_{rising}} \sim \mathrm{W_n}$. Thus also the overall delay of the inverter chain increases, although only one edge contributes to that.

To explain the noise deviation both edges are regarded separately again. It was outlined in the last paragraph that the falling edge behaves like before, where all transistor sizes were modified. $\sigma_{\mathrm{noise,falling}} \sim 1$. The noise deviation at the rising edge will be examined now. This case is illustrated on Figure 4.6 (right). An explanation similar to Equation 4.2, which was derived before, is displayed in Equation 4.4:

$$\sigma_{\mathrm{noise,rising,voltage}} \sim \frac{1}{\mathrm{C}} \cdot \mathrm{i_n} \cdot \sqrt{\mathrm{t}} \sim \frac{1}{\mathrm{W_n}} \cdot 1 \cdot \sqrt{\mathrm{W_n}} \sim \frac{1}{\sqrt{\mathrm{W_n}}} \tag{4.4}$$

Despite different factors, the result is equivalent to Equation 4.2. Again the result has to be transformed to a deviation in time. This time, the slope varies. It was shown that $\mathrm{t_{rising}} \sim \mathrm{W_n}$. Regarding this, the deviation in time is expressed in Equation 4.5:

$$\sigma_{\mathrm{noise,rising}} \sim \mathrm{t_{rising}} \cdot \sigma_{\mathrm{noise,rising,voltage}} \sim \mathrm{W_n} \cdot \frac{1}{\sqrt{\mathrm{W_n}}} \sim \sqrt{\mathrm{W_n}} \tag{4.5}$$

Finally the falling and rising edge have to be combined. As known, variances of independent random variables can be added. The resulting relation is given in Equation 4.6:

$$\sigma_{\mathrm{noise}} = \sqrt{\sigma^2_{\mathrm{noise,falling}} + \sigma^2_{\mathrm{noise,rising}}} \sim \sqrt{\mathrm{W_n} + \frac{1}{\mathrm{W_n}}} \tag{4.6}$$

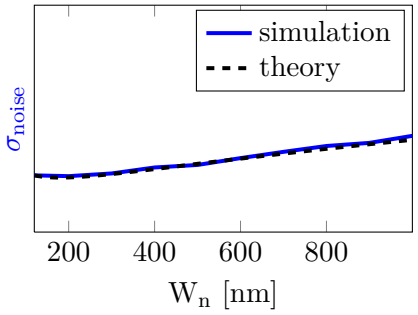


Figure 4.7.: Evaluation of the theory explaining the noise behavior.

The predicted behaviour is compared to the simulation in Figure 4.7. Therefore W_n from Equation 4.6 is regarded as relative width normalized to $W_n = 180\,\text{nm} \mathrel{\hat{=}} 1$.

What is still missing is an explanation why the standard deviation due to mismatch increases in the way it does. Therefore rising and falling edge are regarded separately again, see Figure 4.8. It was explained in the beginning of the section that the mean of the falling edge will not change significantly and only the rising edge contributes to its increase. This is proven on the left plot. Nevertheless the standard deviation of the delay of the falling edge increases.

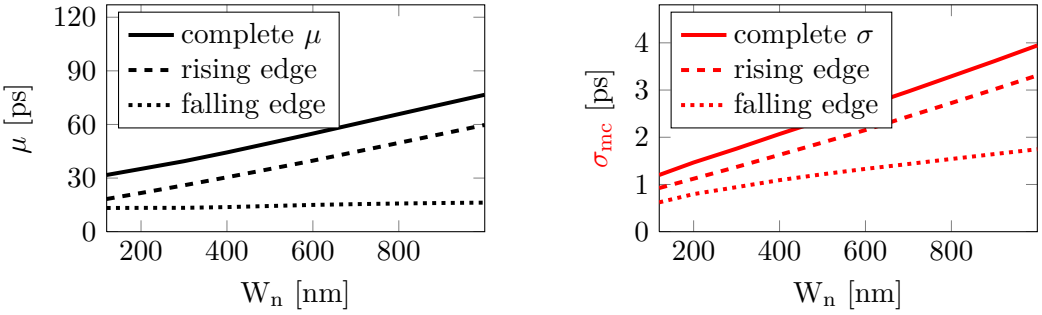


Figure 4.8.: Influence of inverters's nMOS width on delay and its deviation. Plotted for rising and falling edge separately.

To explain the behavior of the deviation the simple circuit from Figure 4.3 on page 37 is regarded again. There the equation 4.1 for the capacitor voltage was given and solved for t, see equation 4.7:

$$t = \frac{C}{i_0} \cdot U \tag{4.7}$$

Both C and i_0 are random variables of a certain distribution. As usual they are assumed Gaussian distributed. For increasing W the distributions will change, in particular the mean of i_0 increases with increasing W of the driver transistor and the mean of C increases with increasing W of the load transistor. While linear operations like addition are easy to perform on such random variables, building a quotient is not. In this case, mean

and variance cannot be regarded separately any more. Thus both distributions and their behavior for increasing W have to be known completely. Even then the calculations are challenging. As a simplification C is regarded as a constant in the following and only i_0 is assumed a Gaussian distributed random variable. As it appears in the denominator, investigating the influence on t analytically is still not trivial. Even the kind of distribution changes. An illustration is given in Figure 4.9. The figure displays the density function of a Gaussian distributed random variable X and $\frac{1}{X}$. It is obvious that the density function of $\frac{1}{X}$ is not symmetric anymore and assuming a Gaussian Distribution for the result is only an approximation.

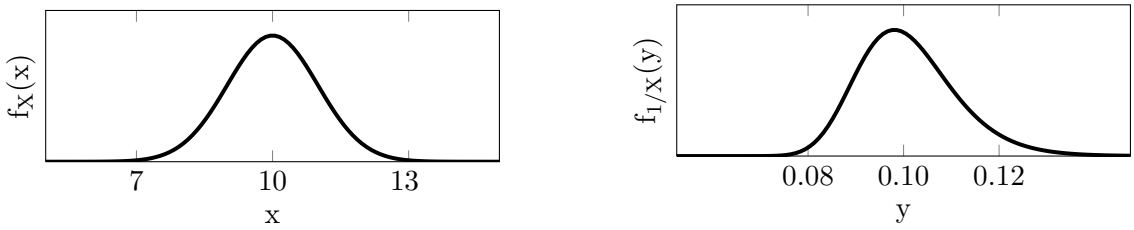


Figure 4.9.: Illustration of a non-linear operation on the density function of a Gaussian distributed random variable.

It was shown that further investigation requires a statistical characterization of i_0 and C and non-linear stochastics to obtain a suitable theoretically well-founded explanation. This effort is not made here. The simulation results provide sufficient information anyhow for the following investigations in this thesis.

4.1.3. Influence of Width W and C_{load} on Slew Rate

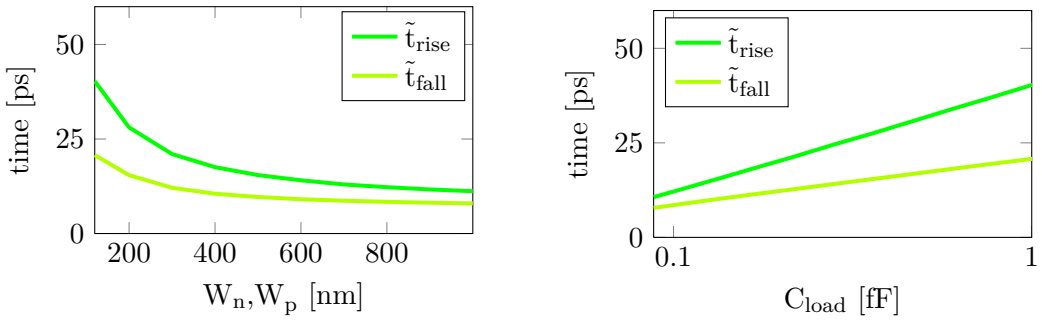


Figure 4.10.: Influence of inverter size and load on slew rate.

Figure 4.10 shows the main contributors to fast edges. As expected, the transition times decrease with increasing transistor sizes at constant load (left plot), and the transition times increase with increasing capacitive load at constant transistor sizes (right plot). Note that the transition time decreases very slowly for larger transistors while the load affects the transition time almost linearly.

4.1.4. Delay Elements in Series

Up to that point a delay element consists of two inverters. The statistical delay behavior for this case was explored. Using the gained delay model allows an easy way to derive the behavior of an inverter chain consisting of 2n ($n \in \mathbb{N}$) inverters. Assuming independent random variables the overall delay and the standard deviation can be determined by the following equations:

$$\mu_{2n} = n \cdot \mu \tag{4.8}$$

$$\sigma_{2n} = \sqrt{n} \cdot \sigma \tag{4.9}$$

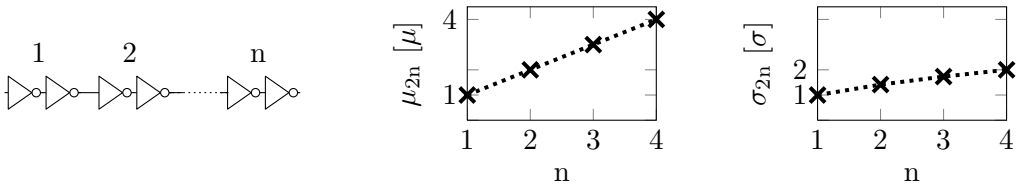


Figure 4.11.: Influence of multiple inverter delay elements in row.

This behavior is illustrated in Figure 4.11. It shows that spending more inverters in a delay chain increases the standard deviation, which is a wanted effect (right plot). But the overall standard deviation grows much slower than the mean (left plot). That means increasing the standard deviation by this measure is very costly in terms of speed and size.

4.2. Switch

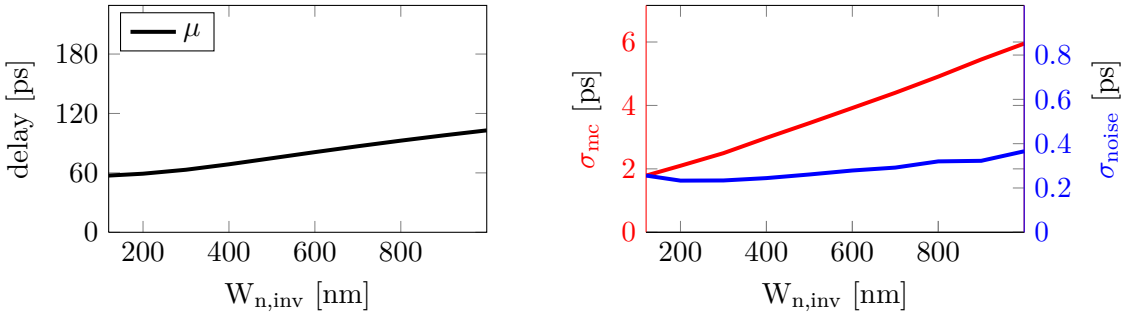


Figure 4.12.: Influence of inverter size on delay of the switch element and its deviation.

The switch circuit is analyzed analogously to the delay circuit. All parameter sweeps are available in appendix A.2. Figure 4.12 displays the effect of $W_{n,inv}$ on the delay and its standard deviation. $W_{n,inv}$ is the width of the nMOS transistors of the inverters involved in the switch circuit. Transmission gates and pMOS sizes remain minimum. The circuit is displayed in Figure 3.12, page 24.

The delay of the switch element behaves similar to the delay of the delay element. The mean delay increases, but the standard deviation due to mismatch increases far more while the standard deviation due to noise hardly changes.

4.3. Latch

In subsection 3.4.3 the measure $\Delta t_{20\,\%}$ was introduced which expresses the quality of noise resistance for a latch. It was illustrated in Figure 3.20 on page 28. The less this measure the better is the latch in terms of noise sensitivity.

Another important quality measure is the bias of the latch. It was described in subsection 3.4.4. It results from manufacturing variations and is expressed as the Δt where the response flips. The measure that will be monitored is the standard deviation σ_{mc} of this bias for several Monte Carlo runs. The ideal value is 0 and stands for an unbiased latch.

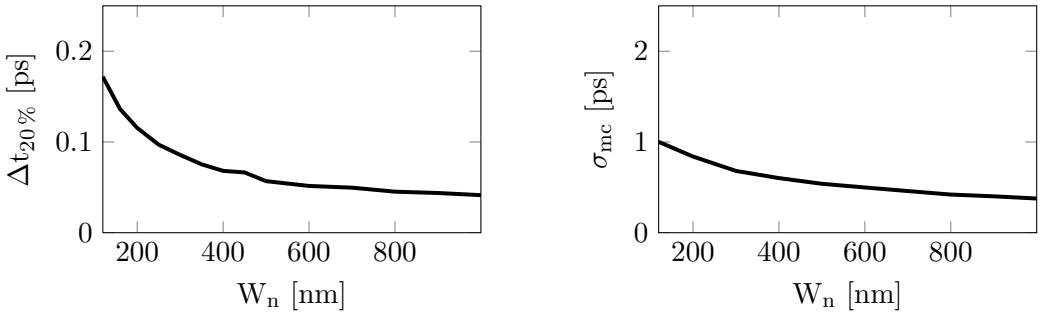


Figure 4.13.: Influence of latch's nMOS width on noise sensitivity and bias.

Both measures were monitored over several parameter sweeps. All plots can be found in the appendix A.3. The most outstanding plot is shown in Figure 4.13. It demonstrates that increasing the width of all nMOS transistors of the latch decreases both measures significantly. This is a simple but effective method to improve the latch behavior.

During the parameter sweep all other parameters were not modified. All other transistors are minimum size. The inputs are ideal edges (see Figure 3.20) with a rise time set to 10 ps. As it was shown before the input rise time is very important for the latch response. Thus the plotted values should not be regarded as absolute values but as a trend. Note that increasing W_n also increases the input capacitance which might decrease the input rise time in a real circuit. This effect might worsen the latch behavior and was not regarded in this scenario, where ideal inputs were used.

The positive effect of an increasing W_n can be explained with regard to the circuit and its function which was introduced in section 3.4. The nMOS transistors have to unload an internal node and their strength in terms of current increases with the width. Thus the internal node can be discharged faster which means the stable state is reached earlier and thus a smaller Δt is sufficient for a reliable decision.

Chapter 5

Arbiter PUF Simulation and Modelling

This chapter deals with a complete arbiter PUF which consists of the components introduced in chapter 3. The general idea was already explained in section 1.4. The PUF will be simulated on transistor level and measures for uniqueness and reproducibility will be evaluated.

Furthermore a new model-based simulation approach is suggested to reduce computation time significantly. Enhancements based on chapter 4 are applied and evaluated using the new method. The results are verified by a transistor level simulation of the newly proposed circuit.

5.1. Composition and Operation

Figure 5.1 displays an arbiter PUF with an 8 bit challenge and an 8 bit response. This construction will be examined in this chapter. Each of the 8 rows consists of

- a source providing a rising edge,
- 8 unit cells, each consisting of a switch element and buffers,
- an arbiter deciding for the earlier edge.

The arbiter was examined in section 3.4. For the unit cells the switch element introduced in section 3.3 is followed by two inverters introduced in section 3.1. Each unit cell offers 2 ways of forwarding the input signals, which equivalents a 1 bit challenge. To allow 8 challenge bits 8 unit cells are interconnected in series. Each row generates a 1 bit response. To generate 8 response bits 8 rows are used in parallel. An Arbiter PUF can easily be expanded to longer challenges or responses in the same way.

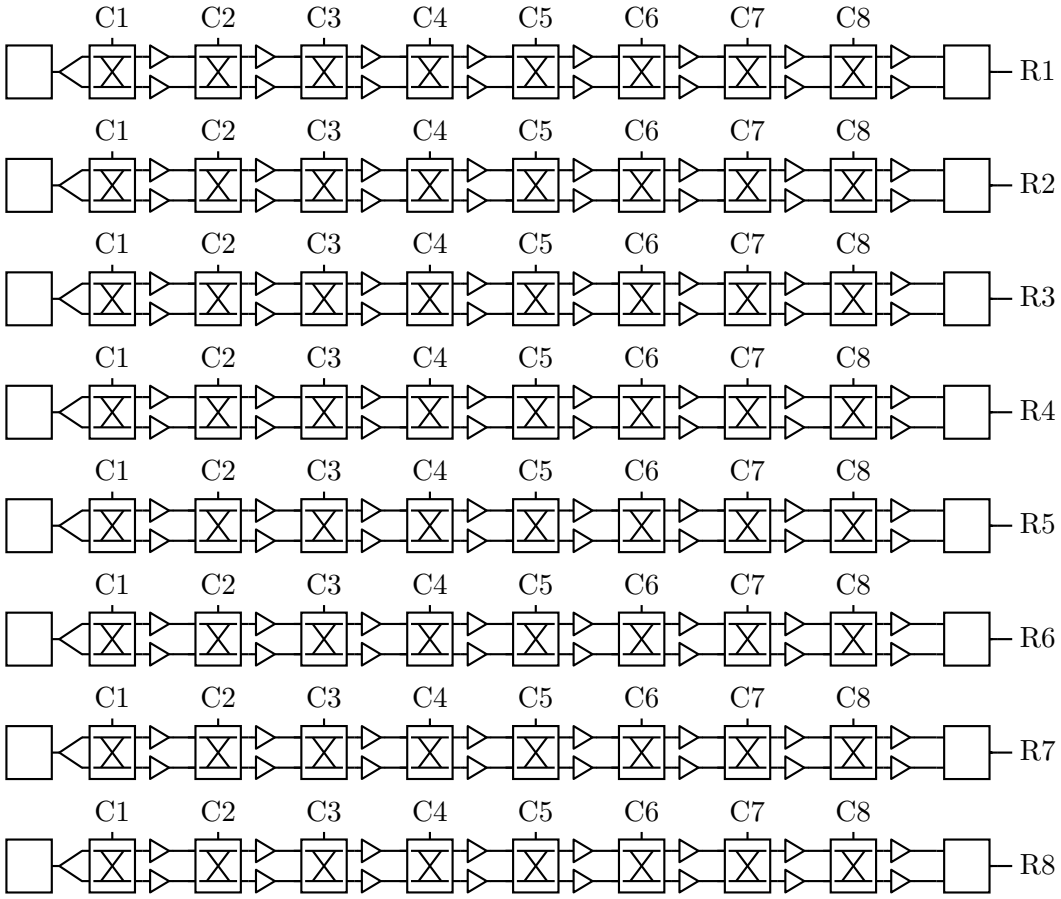


Figure 5.1.: Arbiter PUF with 8 bit challenge and 8 bit response.

The Arbiter PUF is a delay-based PUF. Each response bit is generated by the comparison of delays. The operation procedure is described in the following. First the challenge bits C1...8 are applied and left constant during the evaluation. Thus two signal paths per row are established, leading from the source to the arbiter. The configuration of the paths depends on the challenge bits. The signal from the source is pulled low and the low signal is propagated to the arbiter. Both signals low prepares the arbiter for the decision. Then the source changes to a high signal. A rising edge is propagated through the configured path to the arbiter. The arbiter decides which of the two paths propagated the rising edge faster, the result is stored and can be read out afterwards. As long as the source is high, the arbiter result is stable.

5.2. Simulation (Transistor Level)

The quality of this simple Arbiter PUF will now be evaluated. The minimum size components from chapter 3 are combined to the PUF in Figure 5.1. The determined quality will afterwards be the reference for enhancements.

The simulation effort on transistor level is enormous. As explained in chapter 2 every simulation needs to be performed several times. Furthermore, not only the number of devices increases in contrast to a single component, also a larger time interval (→simulated time) must be simulated due to longer signal paths. The following numbers result from the simulation of 50 MC runs, each with 50 noise runs, and each repeated for 25 random challenges.

5.3. Quality

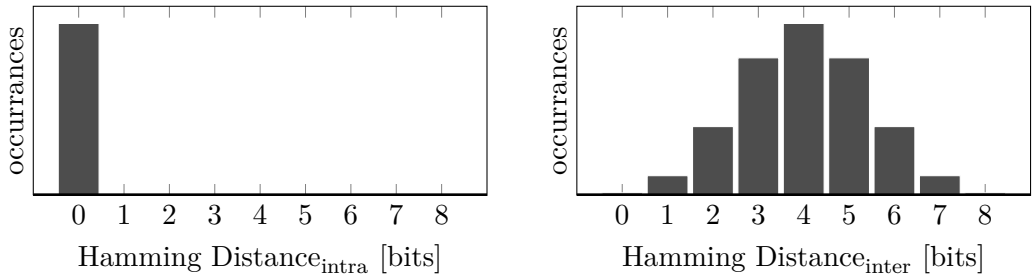


Figure 5.2.: Ideal Histograms. Left: All responses of the same PUF on the same challenge are the same; the Hamming Distance is always 0. Right: Responses of different PUFs differ by a mean of 50 % or 4 bit; the Histogram of the Hamming Distances is binomial distributed.

Figure 5.2 displays the histograms which represent the quality measures reproducibility and uniqueness. The plots display the ideal case of the measures introduced in section 1.2. The aim is to approach these ideal distributions as far as possible.

The calculation of the histograms from recorded measurements is explained in [2, p. 20ff.] and summarized in words in the following. For the intra-histogram, responses of the same PUF on the same challenge are compared (→multiple readouts). Every possible pair of responses is compared by calculating the Hamming Distance. Finally a histogram of all comparison results, even for different PUFs and challenges, is plotted. The inter-histogram is calculated accordingly. Responses of different PUFs are compared rather than different readouts and the procedure is done for every challenge and readout.

Figure 5.3 displays the intra and inter histograms of the simulated reference PUF (transistor level simulation). Fractional $\mu_{\text{inter}} \approx 50\,\%$ means uniqueness is perfect. This is expected due to the symmetric structure; no potential unbalances except the latch bias are regarded. Reproducibility leaves room for improvements. A fractional $\mu_{\text{intra}} = 6.57\,\%$

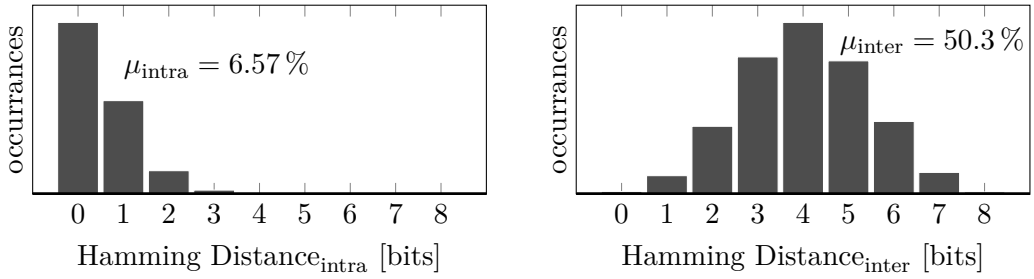


Figure 5.3.: Histograms illustrating reproducibility and uniqueness of the minimum size standard Arbiter PUF.

equivalents a bit error rate of 6.57 %, which is much higher than the ideal value of 0 and far too high for most purposes.

5.4. Simulation (Component Level)

This section will introduce a novel simulation approach. In chapter 3 all elements of a standard Arbiter PUF were examined in detail. The gained knowledge is used to build stochastic models for every basic element. The main characteristic variable will be the propagation delay, or more precisely its deviation. Finally these models will be combined to simulate the whole PUF.

The propagation delay is the only value that is needed to characterize a delaying element's behavior for a specific path, assuming constant environmental conditions and neglecting noise. The simulation has shown that the distribution of delays of the delaying elements caused by mismatch can be modeled with a Gaussian distribution. Two values are sufficient to characterize the random variable. These are the mean and the standard deviation. Therefore we introduce a random variable $\mathrm{T}_{\mu,\sigma}$. Every delaying path will be associated with an instance of $\mathrm{T}_{\mu,\sigma}$.

For simplification the following descriptions refer to a single row, the same procedure can be applied for all rows separately. The absolute delays of the two paths per row are named $\mathrm{t_a}$ and $\mathrm{t_b}$. For the algorithmic processing the values are stored in a vector $(\mathrm{t_a}, \mathrm{t_b}, 1)^{\mathrm{T}}$. The initial value is $(0, 0, 1)$, it is updated at every delaying element and finally evaluated at the arbiter.

5.4.1. Modeling Delay Elements

The simplest element is the delay element. For minimum size devices the values $\mu_{\text{delay}} = 32\,\text{ps}$ and $\sigma_{\text{delay,mc}} = 1.2\,\text{ps}$ were determined. A delay block consists of two parallel delay chains which are stochastically independent. They refer to two different mismatch instances.

The delay element is characterized by the propagation delays $t_a^+ = T(A)$ and $t_b^+ = T(B)$, where A and B are the two paths. The values t_a^+ and t_b^+ remain constant for an instance. This is similar to a monte carlo instance at transistor level simulation.

The update of the absolute propagation delay is mathematically described in Equation 5.1. n stands for the current delaying element, n-1 for the result of the previous element or the initial value for the first one.

$$\begin{pmatrix} t_a \\ t_b \\ 1 \end{pmatrix}_n = \begin{pmatrix} 1 & 0 & t_a^+ \\ 0 & 1 & t_b^+ \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} t_a \\ t_b \\ 1 \end{pmatrix}_{n-1} \tag{5.1}$$

5.4.2. Modeling Switch Elements

It was shown in section 3.3 that the switch elements can be modeled similar to delay elements. Realistic values for the delay distribution were determined to $\mu_{\text{delay}} = 57\,\text{ps}$ and $\sigma_{\text{delay,mc}} = 1.8\,\text{ps}$. In contrast to the simple delay element a switch offers four different paths. They are not necessarily all stochastically independent, but the ones that are active simultaneously are. The behavior depends on the switch value C. Mathematically this is described in Equation 5.2 and 5.3.

$$\begin{pmatrix} t_a \\ t_b \\ 1 \end{pmatrix}_n = \begin{pmatrix} 1 & 0 & t_{aa}^+ \\ 0 & 1 & t_{bb}^+ \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} t_a \\ t_b \\ 1 \end{pmatrix}_{n-1} \quad \text{if C=0} \tag{5.2}$$

$$\begin{pmatrix} t_a \\ t_b \\ 1 \end{pmatrix}_n = \begin{pmatrix} 0 & 1 & t_{ba}^+ \\ 1 & 0 & t_{ab}^+ \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} t_a \\ t_b \\ 1 \end{pmatrix}_{n-1} \quad \text{if C=1} \tag{5.3}$$

There $t_{aa}^+ = T(\text{inA} \rightarrow \text{outA})$ stands for the delay of the path from input A to output A. For an illustration see Figure 3.9 on page 22. t_{bb}^+, t_{ba}^+ and t_{ab}^+ are defined respectively. Note that the expressions also display the fact that either the paths inA $\rightarrow$ outA and inB $\rightarrow$ outB *or* the paths inB $\rightarrow$ outA and inA $\rightarrow$ outB are active simultaneously, depending on the switch input C.

5.4.3. Combining Delay and Switch Element Models

The following procedure models the propagation of the signal through the paths. Initially the delay of both paths is set to 0, the initial delay vector is defined in Equation 5.4:

$$\begin{pmatrix} t_a \\ t_b \\ 1 \end{pmatrix}_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.4}$$

Then the propagation matrix of every switch and delay element is successively applied on the propagation vector in the order illustrated in Figure 5.1. Each of these steps was explained before. Finally the difference of both delays is calculated in Equation 5.5:

$$\Delta t_{\text{mismatch}} = t_b - t_a \tag{5.5}$$

5.4.4. Modeling Noise in Delay and Switch Elements

Not only mismatch influences the propagation delays, also noise can do. Noise will also be transformed to an uncertainty in time. It is assumed Gaussian distributed and the standard deviation was also determined in the previous chapters in addition to the mismatch deviation. For the delay element σ_{noise} is 0.18 ps. For the switching element's rising edge σ_{noise} was determined to 0.26 ps.

For noise, all elements are handled stochastically independent. Furthermore the influence of noise is treated independently from the challenge bits C. This allows to summarize the random variables for both paths and combine it to a new random variable whose standard deviation can easily be calculated, see Equation 5.6 for n stages:

$$\sigma_{\text{noise, path}} = \sqrt{n \cdot \sigma^2_{\text{noise, delay}} + n \cdot \sigma^2_{\text{noise, switch}}} \tag{5.6}$$

Finally the two paths are combined. Again due to the independence this can be transformed to a new random variable using Equation 5.7:

$$\sigma_{\Delta t,\ \text{noise}} = \sqrt{2 \cdot \sigma^2_{\text{noise, path}}} \tag{5.7}$$

The additional $\Delta t_{\text{noise}} = T_{\mu=0,\sigma_{\Delta t,\ \text{noise}}}(t)$ is added to $\Delta t_{\text{mismatch}}$ resulting in the final delay difference value Δt, which is the arbiter input.

5.4.5. Modeling Arbiter

As shown before the absolute values are ignored, only the time difference between the two paths is the deciding arbiter input. As shown in subsection 3.4.3, an arbiter is not only prone to mismatch and noise, but its decision behavior also depends on the rise time of the inputs.

The characteristic value caused by mismatch is the bias t_{bias}. It was shown in subsection 3.4.4 that its value can be modeled by a random variable with an underlying Gaussian distribution with $\mu = 0$ and $\sigma = 1.45$ ps. The bias is constant per arbiter instance.

It was shown that the noise behavior can also be modeled with probabilities. Therefore the probability of the decision Q=1 was determined in dependence of $\Delta t - t_{\text{bias}}$, see Figure 5.4 (black). It can be observed that this curve is similar to a cumulative distribution function

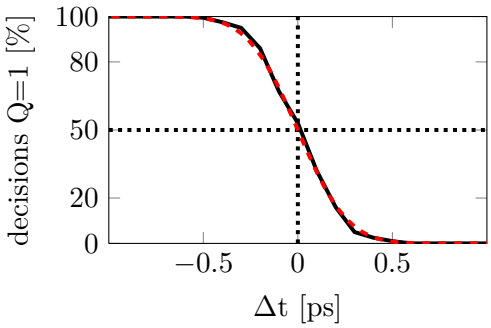


Figure 5.4.: Latch noise decision behavior. Black: Simulation. Red dotted: Approximation by CDF of Gaussion Distribution.

(CDF) of a Gaussian distribution (red dotted) which is parametrized by $\mu = 0$ and a certain σ. The right standard deviation σ can be determined numerically.

To determine the arbiter response, at first the bias is subtracted from the input $\Delta t_{mismatch}$. Then the probability for a decision Q=1 is calculated using the relation from Figure 5.4. Finally the response is chosen by a random variable with the determined probability for Q=1.

5.4.6. Model-based Arbiter PUF Simulation

An Arbiter PUF is built by putting all these stochastic models together. The model can arbitrarily be extended in terms of challenge bits, response bits, MC runs, noise runs, and the number of evaluated challenges. Less computation time is required in contrast to the complete transistor level simulation. The following subsection refers to the values of 50 MC runs, 50 noise runs, and 25 challenges, as for the transient transistor level simulation before.

5.4.7. Component Level Simulation Results

Comparing the results from component level simulation, Figure 5.5, to the reference results from transistor level simulation, Figure 5.3 on page 48, it can be observed that both simulations match very well. This means the component level simulation is a suitable approximation or even replacement of the transistor level simulation.

Table 5.1 displays the values which characterize the distributions of both simulation approaches. The most important values are μ_{intra} and μ_{inter}. It becomes obvious that both kinds of simulation display similar results with an acceptable accuracy.

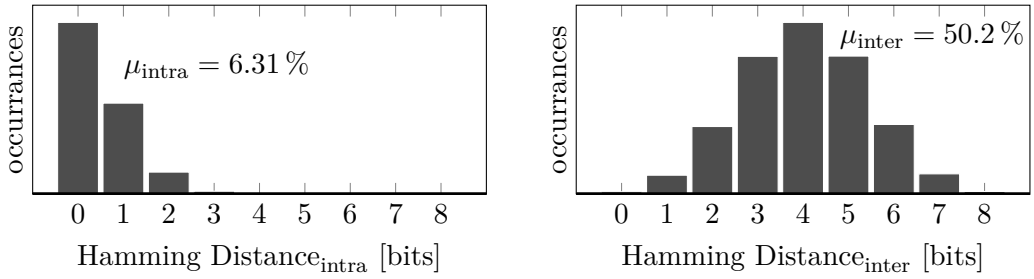


Figure 5.5.: Histograms illustrating reproducibility and uniqueness of the minimum size standard Arbiter PUF. Results from the simulation on Component Level.

simulation	μ_{intra} [%]	μ_{intra} [bit]	σ_{intra} [bit]	μ_{inter} [%]	μ_{inter} [bit]	σ_{inter} [bit]
ideal	0	0	0	50	4	1.41
component level	**6.31**	0.50	0.69	**50.2**	4.02	1.42
transistor level	**6.57**	0.53	0.71	**50.3**	4.02	1.42

Table 5.1.: Evaluation results of the minimum size Arbiter PUF. Both simulation techniques display similar results.

5.4.8. Advantages of Component Level Simulation

It was outlined before that the component level simulation generates similar results to a complete transistor level simulation. However, it requires additional effort in examining separate components and model building. This is additional effort as well as a potential error source. But there are several advantages which will be outlined in this subsection.

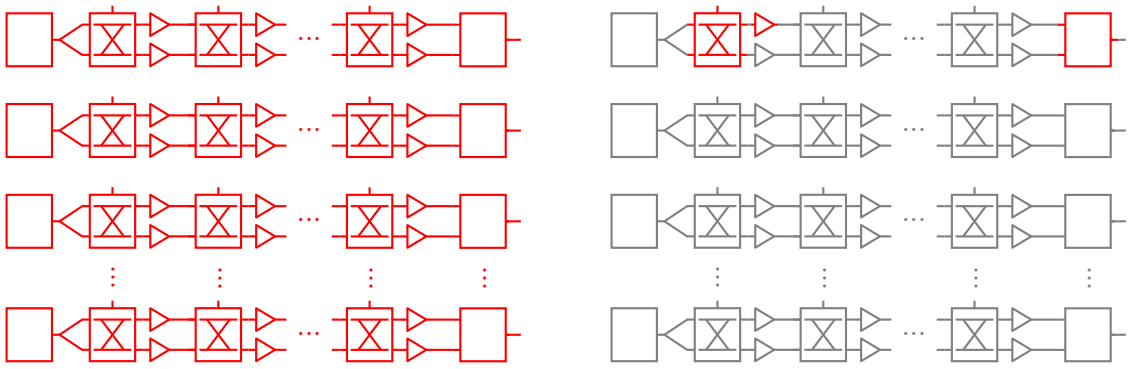


Figure 5.6.: Illustration of devices which need to be simulated on transistor level (red). The component level approach (right) offers a significantly reduced complexity of transistor level simulations compared to the standard approach (left).

Figure 5.6 displays two times the same arbiter PUF. Each time the devices which have to be simulated on transistor level are marked red. On the left is the standard approach. The whole circuit is simulated on transistor level, which is slow. Evaluating 50 MC runs

times 50 noise runs for an 8x8 bit PUF such a simulation took about 8 hours per challenge, even with advanced simulation techniques. Evaluating 25 challenges takes a week. The component approach is faster. Each component must be analyzed separately, which takes some time in advance, but the actual evaluation of the complete PUF using the models is very fast. Evaluating 25 challenges only takes about 1 minute. The values are illustrated in Figure 5.7. Note the logarithmic scaling: $10^{-2}\,\mathrm{h} \approx 30\,\mathrm{s}$, $10^{2}\,\mathrm{h} \approx 4\,\mathrm{d}$. The component approach (red) is divided into model-building and model-based simulation. The model-building for the component level simulation is faster than a single challenge evaluation on transistor level. The model-based simulation is almost negligible in terms of time effort.

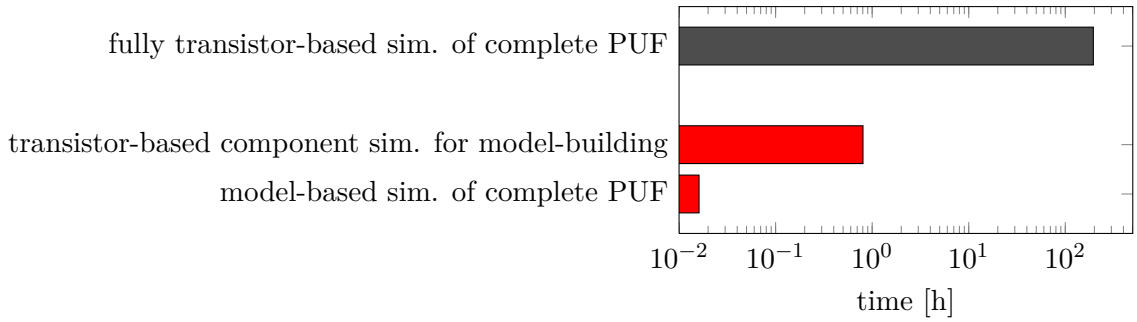


Figure 5.7.: Comparison of simulation times. 25 challenges are evaluated. The component level simulation is faster than a complete transistor level simulation by orders of magnitude.

The simulation on component level is not only faster. It also scales better. Adding additional challenge or response bits does not require additional transistor level simulations. Even evaluating more MC runs, more noise runs, or more challenges does not require further transistor level simulations. Thus the limiting factor simulation time is omitted, because it was already outlined that the model-based simulation is very fast.

Another advantage is the simple interchangeability of components. Each component can be simulated separately. In general, manipulating a component only requires simulating the component again, not the whole system. However, if the input or output behavior of a component changes, this should be regarded for the previous or following component. To give an example: The arbiter was shown to be sensitive to the input slope. Changing the delay element which drives the arbiter input may change the arbiter behavior, too.

5.5. Proposal of an Arbiter PUF based on Enhanced Components

Improvements for every component of the Arbiter PUF were proposed in chapter 4. This is the base for the enhanced PUF deduced in the following.

The minimum size values are the base for all enhancements. It was shown that a larger W_n at the inverters has a positive effect on the delay element. Thus the new value is set

to $W_n = 1\,\mu m$. A similar effect was observed at the switch element. Also here, the new inverter width is set to $W_{n,inv} = 1\,\mu m$ (the transmission gate size is untouched). For the latch, a positive effect of W_n was found, too. Also here the width is set to $W_n = 1\,\mu m$. In sum almost every nMOS transistor was increased by about a factor of 8.

The behavior of the new delay and switch elements are already determined. Their behavior was analysed as part of parameter sweeps in chapter 4. A prediction of the complete latch behavior without further simulations is more complicated. For example, in subsection 3.4.3 the influence of the input slope was examined. To avoid prediction errors the latch was simulated again with the new parameters and with respect to the modified inverters driving the latch. The results are display in Figure 5.8. The mismatch bias increases slightly, but the noise behavior which is relevant for the reliability is improved. The previous behavior (std.) is plotted in dotted gray, the new behavior (enh.) is plotted in red. A comparison of all characteristic values is given in Table 5.2.

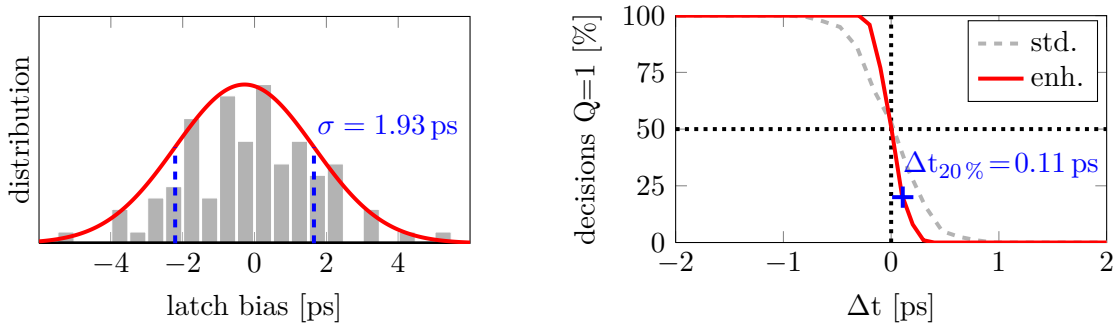


Figure 5.8.: Behavior of the enhanced latch. Left: Distribution of the mismatch bias. Right: Decision behavior due to noise.

parameter	delay element Δt [ps]			switch element Δt [ps]			arbiter [ps]	
W_n [nm]	μ	σ_{mc}	σ_{noise}	μ	σ_{mc}	σ_{noise}	σ_{bias}	$\Delta t_{20\,\%}$
120	32	1.2	0.17	58	1.8	0.26	1.5	0.28
1000	77	3.9	0.30	103	6.0	0.37	1.9	0.11

Table 5.2.: Influence of the changed design parameter on the components' characteristic.

5.6. Evaluation of the Proposed Enhanced Arbiter PUF

The previous section proposed an enhanced parametrization and outlined the improvements on the components. What is missing is an evaluation of the PUF properties regarding the new values. There is no analytic way but it can easily and quickly be done with the previously proposed component level simulation. No additional time consuming transistor level simulation is required. The result is displayed in Figure 5.9. The reference results determined before are displayed in light gray for comparison.

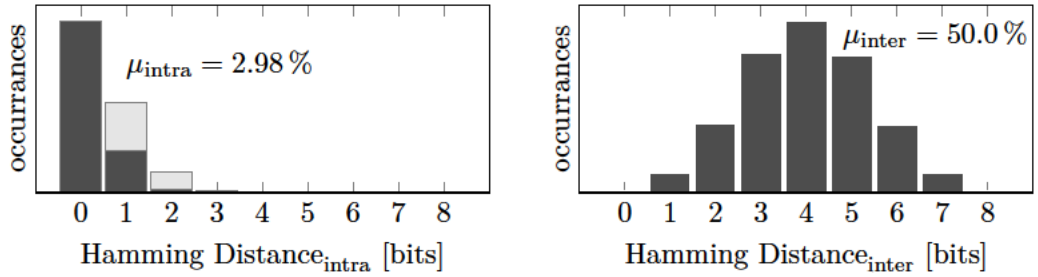


Figure 5.9.: Histograms illustrating reproducibility and uniqueness of the proposed enhanced Arbiter PUF. This evaluation was done on component level which claims for high speed simulation at a suitable accuracy.

To verify the results a last effortful transistor level simulation is performed. It took a week for 15 challenges, instead of a minute for the component level prediction. The result is displayed in Figure 5.10. Again the result from the standard approach is displayed in light gray for comparison. Comparing the histograms and in particular the factional intra distance makes obvious that the prediction is very good, especially if the reduced effort is taken into account.

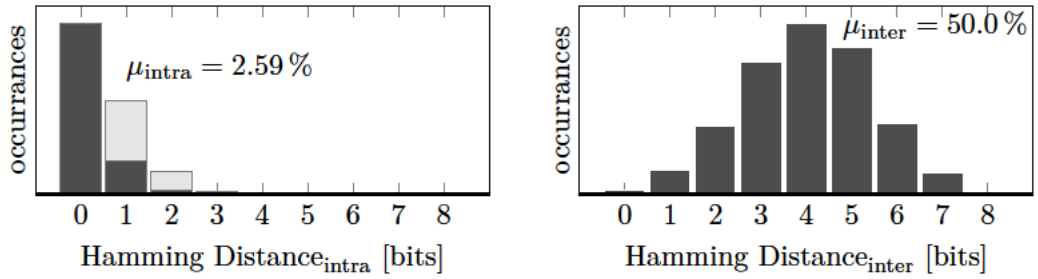


Figure 5.10.: Histograms illustrating reproducibility and uniqueness of the proposed enhanced Arbiter PUF. This evaluation was done on transistor level which claims for maximum simulation accuracy.

Finally Figure 5.11 displays the bit error rates for the standard and the proposed enhanced PUF. The bit error rate, which is equal to μ_{intra}, decreases significantly for the proposed design parameter modifications. The comparison demonstrates that the faster component level simulation generates similar results to the transistor level simulation for both cases. One has to regard that even the transistor level simulation does not provide infinite accuracy. Layout was not regarded in this work, also the suitability of the transistor models is uncertain due to the uncommon usage. Both simulation approaches can be expected to allow predictions on significant improvements, one of which was demonstrated here. However the absolute values should be verified after manufacturing.

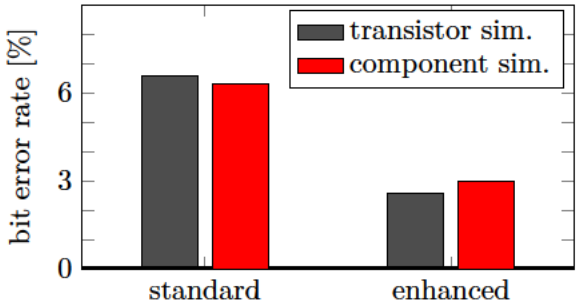


Figure 5.11.: Comparison of simulation results. The bit error rate decreases by half for the proposed enhanced PUF. Results from component level simulation are similar to transistor level simulation.

Chapter 6

Conclusion and Outlook

A new time saving simulation approach was proposed to analyze the reliability in terms of reproducibility of Arbiter PUF constructions considering mismatch and transient noise. It is based on statistical models of its components. The model parameters are gained by transient transistor level simulations. It was shown that the new component level simulation fits the transistor level simulation well. Unreproducible responses are one of the most severe problems of arbiter PUFs. Quantizing the bit error rate for a given construction is accelerated by orders of magnitude with the new approach, e.g. from 7 days to 1 hour for an 8 bit Arbiter PUF.

The components of the Arbiter PUF were investigated, suitable statistical models regarding mismatch and noise for each of the components were found, and the influence of various design parameters was evaluated. It was shown that simple parameter modifications influence the deviation of a components delay in various ways. An example is a larger width of the nMOS of an inverter. In this case the deviation of the delay caused by mismatch increases, even more than the mean delay, while the deviation of the delay due to noise remains almost constant. Similar effects were found for the switch element. Further it was shown that the SR NAND Latch as arbiter is not only sensitive to the time Δt between the input signals, but also much to their slope. The arbiter behavior could also be improved alone by increasing the right transistors.

The improved components were combined to an enhanced proposal of an Arbiter PUF in the same technology, which is TSMC 90nm. The bit error rate of the 8-bit PUF was predicted to 2.98 % by the new approach and verified to be 2.59 % by a conventional simulation. Compared to the reference Arbiter PUF which was build equally but of minimum size devices the bit error rate dropped by 61 %.

Although the bit error rate could be decreased significantly, it still cannot be neglected. The following incomplete itemization suggests some links for future work.

- The Arbiter PUF was simulated in different ways. No attempt was made to verify the results on a manufactured die.
- Influence of environmental conditions like temperature or supply voltage was not evaluated.
- Alternative component implementations were not investigated, e.g. NAND logic multiplexers instead of transmission gates as used in [17], or Gated Inverters (described as *Inverters with Tri-State Outputs* in [16, p. 351]).
- Propagating a falling edge instead of a rising edge was not investigated, although it was demonstrated that the delay of the switch element provides a larger deviation in this case.
- Designing inverters different depending on the edge they have to propagate was not investigated, all inverters were designed equally.
- Several parameters were adjusted separately to analyze their influence on a component property. A Genetic Algorithm could be applied to find even better parameter sets for a component or the complete system.
- The reason why the inverter chains delay variation due to mismatch increases so much for larger widths could not be explained completely yet.
- Edge conditioning right before the arbiter was not investigated, although it was shown that the slope significantly determines the decision reliability.
- Security issues were not regarded, e.g. it could be better to introduce variability in the switch paths instead of the delay paths.
- Currently an arbiter decision contains 1 bit of information, namely if $\Delta t > 0$ or $\Delta t < 0$. Alternative quantizations were suggested in [18] for ring oscillator PUFs. It might be possible to build intentionally biased arbiters with a decision point unequal 0. Combining different arbiters provides more information about Δt. In [18] this is used for error correction, but the usage is not limited to that.
- The readout of the latch could be done with a (clocked) readout circuit similar to SRAM cells instead of loading the internal nodes with buffers.

The idea of novel keyless cryptographic systems using Physical Unclonable Functions is promising. This thesis introduced a new viewpoint on Arbiter PUFs. The derived simulation techniques can also be applied to other delay-based PUFs like Ring Oscillator PUFs to determine important quality measures with low effort. Thus the new insights can serve as base for future investigations to make PUFs suitable for various hardware-based cryptographic applications.

Appendix A

Additional Simulation Results

This chapter adds some simulation results which were not completely described in the main text. For the sake of completeness they are added here and referenced from the text.

A.1. Inverter Parameter Influence

In section 4.1 some inverter chain properties and parameters were introduced. The matrix is displayed in Table A.1. Note that the delay properties and the slew rate are examined on different test circuits, see Figure 4.1.

A.2. Switch Parameter Influence

Modifications to the switch introduced in section 3.3 were proposed in section 4.2. The test circuit from Figure 3.12 is used to perform the changes. The matrix is available in Table A.2. The index $_{\text{tg}}$ refers to transistors of the transmission gate, the index $_{\text{inv}}$ refers to transistors of the involved inverters.

A.3. Latch Parameter Influence

For the latch the width of various transistors is modified to investigate their influence. To keep the circuit symmetric the corresponding transistors of both nand gates are always changed equally. The gray transistor names in Table A.3 refer to the names in Figure 3.18.

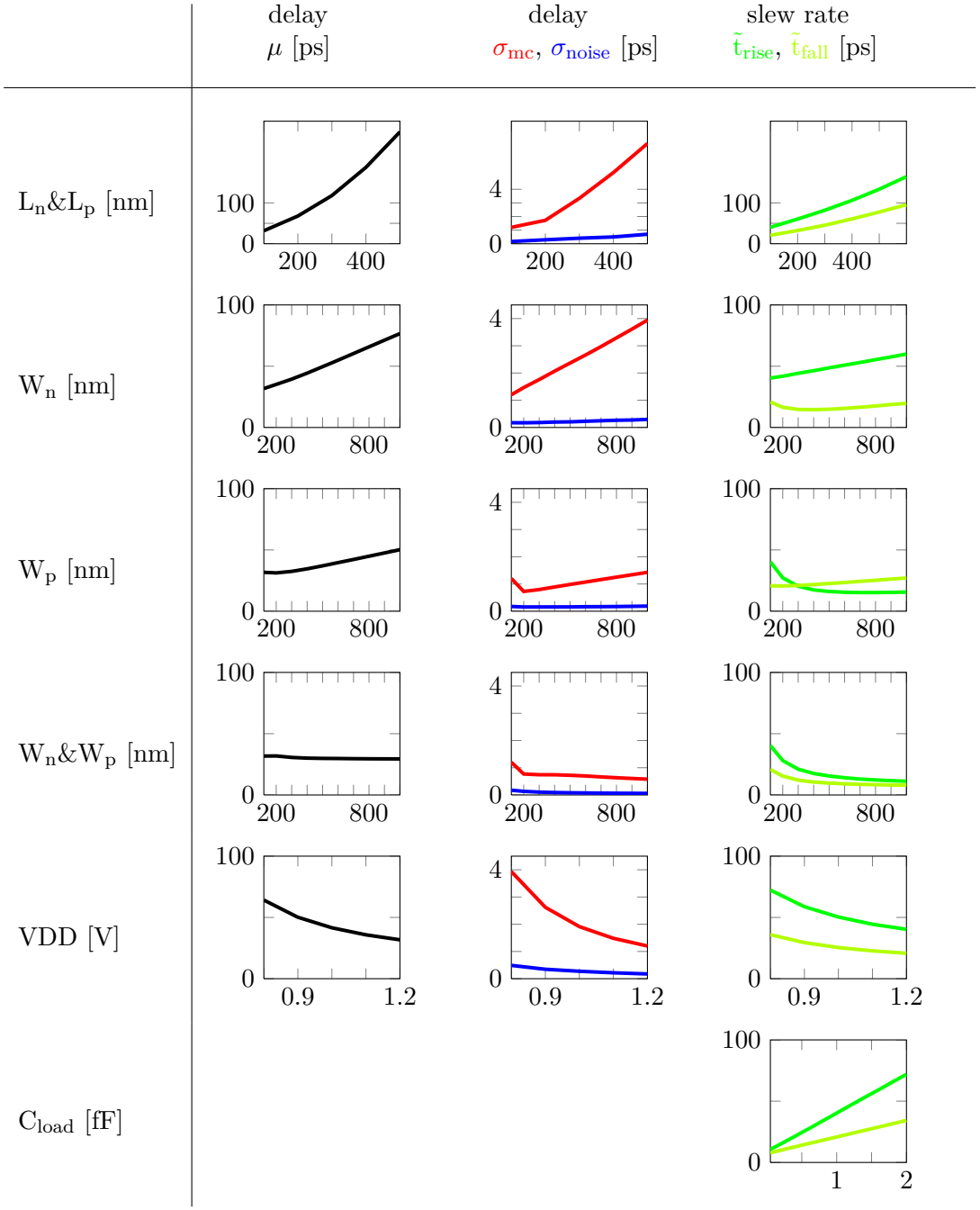


Table A.1.: Influence of design parameters on inverter behavior.

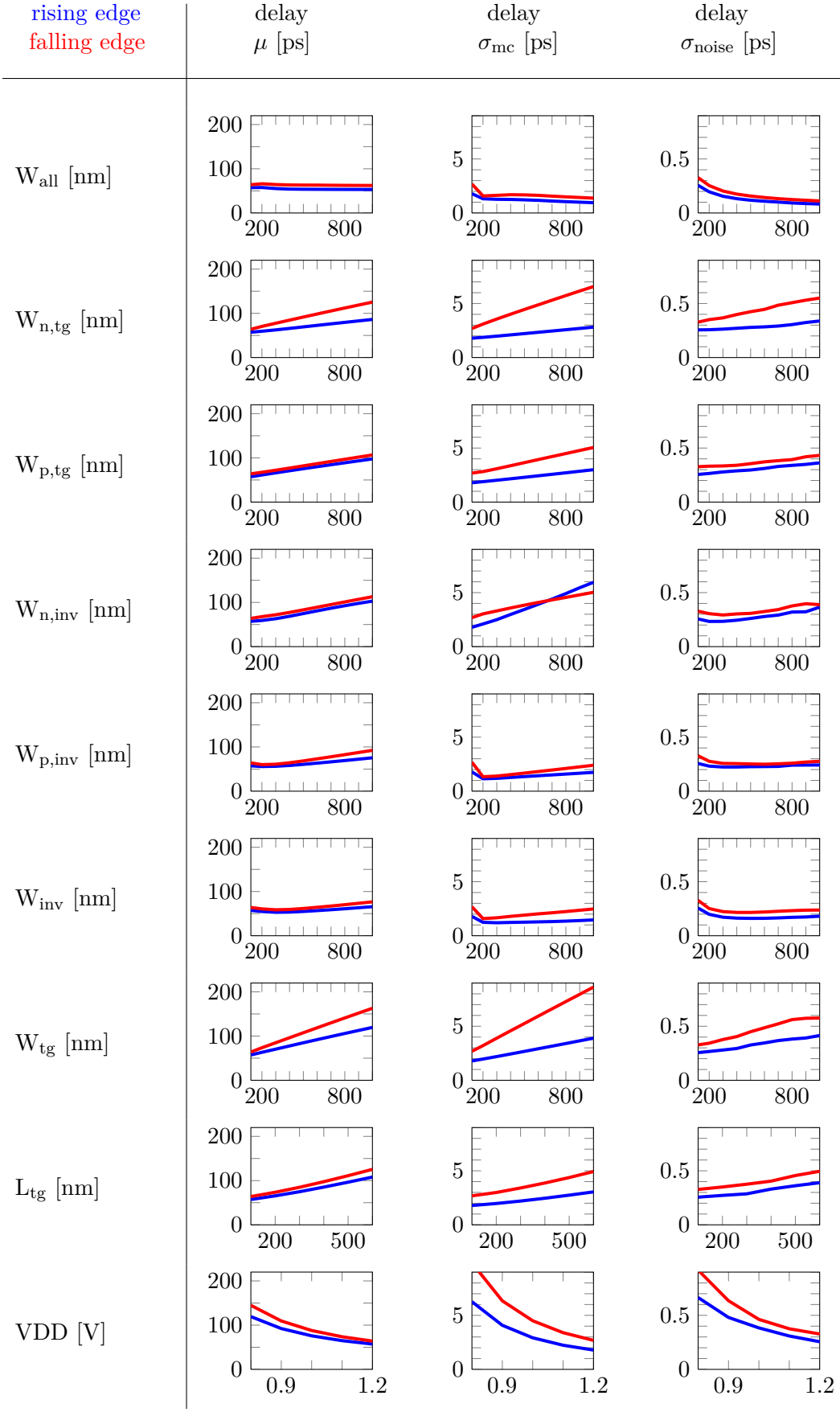


Table A.2.: Influence of design parameters on switch behavior.

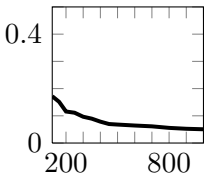
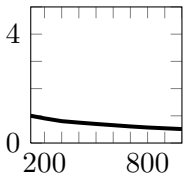
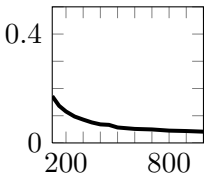
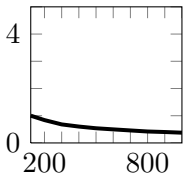
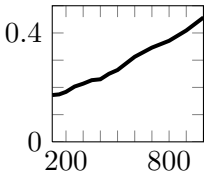
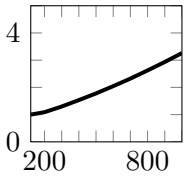
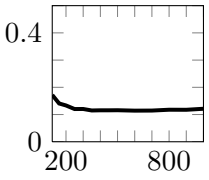
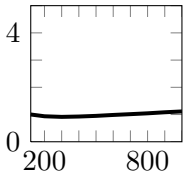
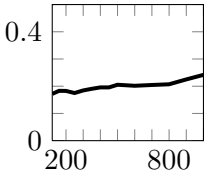
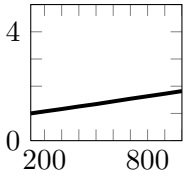
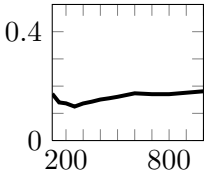
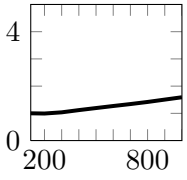
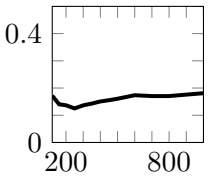
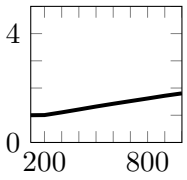
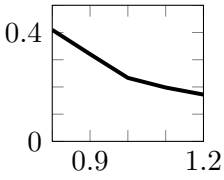
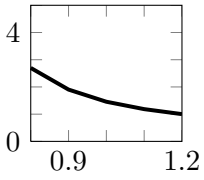
	noise influence $\Delta t_{20\,\%}$ [ps]	mismatch bias σ_{mc} [ps]
W [nm] M0...M7		
W_n [nm] M0, M1, M4, M5		
W_p [nm] M2, M3, M6, M7		
$W_{n,input}$ [nm] M1, M5		
$W_{p,input}$ [nm] M3, M6		
$W_{n,intern}$ [nm] M0, M4		
$W_{p,intern}$ [nm] M2, M6		
VDD [V] [nm]		

Table A.3.: Influence of design parameters on latch behavior.

Appendix B

Simulations

EDIT: Chapter not included in online publication.

Abbreviations

CDF	Cumulative Distribution Function
CMOS	Complementary Metal Oxide Semiconductor
DUT	Device Under Test
FET	Field-Effect Transistor
HD	Hamming Distance
IC	Integrated Circuit
MC	Monte Carlo
MOS	Metal Oxide Semiconductor
NAND	Not AND
PUF	Physical Unclonable Function
SR	Set-Reset
TG	Transmission Gate
TSMC	Taiwan Semiconductor Manufacturing Company
VDD	Supply Voltage

Erklärung

Ich versichere hiermit, dass ich die vorliegende Masterarbeit selbstständig verfasst habe. Stellen der Arbeit, die anderen Werken wörtlich oder dem Sinn nach entnommen sind, habe ich kenntlich gemacht und mit einer Quellenangabe versehen.

Ulm, den 02.12.2013

Markus Schuster

Bibliography

[1] 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*, A.-R. Sadeghi and D. Naccache, Eds. Springer, 2010, pp. 3–37.

[2] R. Maes, "Physically Unclonable Functions: Constructions, Properties and Applications," Ph.D. dissertation, 2012.

[3] C. Böhm and M. Hofer, "Testing and Specification of PUFs," in *Physical Unclonable Functions in Theory and Practice*. Springer New York, 2013, pp. 69–86.

[4] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.

[5] B. Gassend, D. Clarke, M. v. Dijk, and S. Devadas, "Silicon Physical Random Functions," in *Proceedings of the Computer and Communication Security Conference*. ACM, 2002, pp. 148–160.

[6] A.-R. Sadeghi and D. Naccache, Eds., *Towards hardware-intrinsic security: Foundations and practice*. Heidelberg and New York: Springer, 2010.

[7] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA Intrinsic PUFs and Their Use for IP Protection," in *Cryptographic Hardware and Embedded Systems (CHES)*, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds. Springer Berlin Heidelberg, 2007, vol. 4727, pp. 63–80.

[8] 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*, 2007.

[9] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Symposium on VLSI Circuits. Digest of Technical Papers.*, 2004, pp. 176–179.

[10] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for Design and Implementation of Secure Reconfigurable PUFs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 1–33, 2009.

[11] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security.* ACM, 2010, pp. 237–249.

[12] M.-D. Yu and S. Devadas, "Secure and Robust Error Correction for Physical Unclonable Functions," *Design & Test of Computers, IEEE*, vol. 27, no. 1, pp. 48–65, 2010.

[13] R. Kumar, V. C. Patil, and S. Kundu, "Design of Unique and Reliable Physically Unclonable Functions Based on Current Starved Inverter Chain," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2011, pp. 224–229.

[14] M. Ortmanns, "Analog CMOS Circuit Design: MOS Technology and Devices Review," Ulm University, 15.10.2012.

[15] A. Maiti, L. McDougall, and P. Schaumont, "The Impact of Aging on an FPGA-Based Physical Unclonable Function," in *International Conference on: Field Programmable Logic and Applications (FPL)*, 2011, pp. 151–156.

[16] R. J. Baker, *CMOS: Circuit Design, Layout, and Simulation*, 3rd ed., ser. IEEE Press Series on Microelectronic Systems. Wiley, 2010.

[17] L. Lin, D. Holcomb, D. K. Krishnappa, P. Shabadi, and W. Burleson, "Low-power sub-threshold design of secure physical unclonable functions," in *ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, 2010, pp. 43–48.

[18] A. Maiti, I. Kim, and P. Schaumont, "A Robust Physical Unclonable Function With Enhanced Challenge-Response Set," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 333–345, 2012.