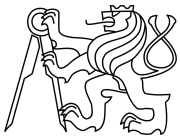
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF MASTER'S THESIS

Title:	Behaviour Analysis and Improvement of the Proposed PUF on FPGA
Student:	Bc. Filip Kodýtek
Supervisor:	prof. Ing. Róbert Lórencz, CSc.
Study Programme:	Informatics
Study Branch:	Computer Security
Department:	Department of Computer Systems
Validity:	Until the end of summer semester 2016/17

Instructions

Analyze statistical parameters of the proposed ring-oscillator-based PUF. Furthermore, test the quality of the PUF output in dependence on the temperature conditions and the change of supply voltage. Propose suitable modifications of the PUF based on the results of the experiments that will improve the quality of its output in case of varying physical conditions.

References

Will be provided by the supervisor.

L.S.

prof. Ing. Róbert Lórencz, CSc.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 2, 2016

Czech Technical University in Prague
Faculty of Information Technology
Department of computer systems

Master's thesis

Behaviour Analysis and Improvement of the Proposed PUF on FPGA

Bc. Filip Kodýtek

Supervisor: prof. Ing. Róbert Lórencz, CSc.

9th May 2016

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor prof. Ing. Róbert Lórencz, CSc. for his guidance during my studies and the time he devoted to me when I was working on this interesting topic. I would also like to thank Ing. Jiří Buček for his help with performing the experiments presented in this work and for his helpful comments and advice. Finally, I would like to express my deep gratitude to my family for their support throughout my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 9th May 2016 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Kodýtek, Filip. *Behaviour Analysis and Improvement of the Proposed PUF on FPGA*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

Abstrakt

Tato práce se zabývá fyzicky neklonovatelnými funkcemi (PUF) na FPGA. Nejprve poskytneme čtenáři literární rešerši týkající se problematiky PUF obecně a také různé konstrukce PUF se zaměřením na ty, jež jsou vhodné pro FPGA. Poté představíme PUF založený na kruhových oscilátorech, navržený v naší předešlé práci, a popíšeme jeho vlastnosti. Tento PUF je následně analyzován a testován v různých teplotních podmínkách a při různém napájecím napětí. Na základě výsledků provedených experimentů navrhneme vhodné úpravy tohoto PUFu ke zvýšení kvality jeho výstupu.

Klíčová slova Fyzicky neklonovatelná funkce, FPGA, kruhový oscilátor, bezpečnost hardwaru, identifikace zařízení, generování klíčů.

Abstract

This thesis deals with Physical Unclonable Functions (PUFs) on FPGAs. First, we provide a literature research concerning PUFs in general and their various constructions with a focus on PUFs suitable for FPGAs. Then we introduce PUF design proposed in our previous work which is based on ring oscillators and we discuss its properties. The proposed PUF is analysed and tested at varying temperature and voltage. Based on the results of the experiments, we propose suitable modifications of the PUF design in order to improve the quality of its output.

Keywords Physical unclonable function, FPGA, ring oscillator, hardware security, device identification, key generation.

Contents

List of Figures

List of Tables

Introduction

Motivation and background

Electronic devices are currently becoming an integral part of our everyday life. Such devices (for example mobile phones, smart cards, RFIDs) can be used to authenticate their owners and provide them with access to private areas or their bank accounts, to store personal data, and for many other applications. Since these devices are widespread and commonly used, they are a target for adversaries. This fact implies a problem with security. Most of the devices contain some secret information or key which is used to authenticate their owners. Therefore this secret has to be stored in a secure manner so that the potential adversary is not able to extract it from the device.

To prevent an adversary from obtaining the secret from the device, it is necessary to consider various countermeasures against possible attacks when designing the architecture of the device. However, designing such secure architecture is not a trivial task. There are numerous possible attacks on the devices that the adversary can use. From the perspective of hardware security, the possible threats are side channel attacks such as power analysis (simple power analysis, differential power analysis etc.), timing analysis and also fault injection attacks. Of course, the adversary can perform other attacks than physical attacks. One can also perform mathematical attacks (linear cryptanalysis, differential cryptanalysis etc.) on the cipher that is used, the cryptographic protocol itself, or exploit wrong implementation of the cryptographic system.

From what has been said, it is obvious, that secure storage and usage of the secret key is a complex task. However, for secure storage of keys we can use Physical Unclonable Functions, which are able to hide the secret in a secure manner. Usually the secret keys are stored in a non-volatile memory, but that is difficult to secure and therefore it is expensive. Non-volatile memory also tends to be vulnerable to invasive attacks, because the key is stored in a digital form. For a high level of security, the electronic devices have to be

protected by expensive circuits that are able to detect manipulation with the device and, moreover, they need to be continually supplied with power. An additional disadvantage can be the cost of even basic cryptographic operations for resource-constrained platforms such as RFID chips.

These issues were one of the motivations that contributed to the deeper interest and extended development in research of Physical Unclonable Functions. Physical Unclonable Functions (abbreviated as PUFs) are increasingly used in proposals of cryptographic protocols and security architectures. PUF is a function based on physical properties which are unique for each device. It exploits local mismatches and differences between physical components of a device arising during the manufacturing process to generate unpredictable outputs. Its concept is based on these random variations which cannot be controlled during the manufacturing process because they result from the effects of random and uncontrollable influences. Therefore it is impossible or extremely difficult to produce two identical devices with the same physical properties which are used in the PUF present on these devices. It is primarily these random variations arising during the manufacturing process that play the main role in how the PUFs are used and which source of randomness they benefit from.

There is a strong similarity with human biometrics, such as fingerprints, retina and others. For example, we are able to identify any person by her fingerprints. Using physical properties as a fingerprint of a device, we can similarly identify the electronic devices because the physical properties are unique for each device and also random (or unpredictable) among various devices.

There is a wide spectrum of applications where PUFs can be used. Among others, they can be used for device identification, authentication, anti-counterfeiting, binding software to hardware platforms, cryptographic key generation and they can also be integrated into cryptographic algorithms. Nowadays, security products based on PUFs are already being announced for the market, focusing on intellectual property protection, anti-counterfeiting and RFID applications (Verayo, Intrinsic-ID, QuantumTrace, Invia).

Research goals

This thesis deals with Physical Unclonable Functions on field-programmable gate arrays (FPGAs). A novel PUF design for FPGA was proposed and implemented in our previous work [13, 14, 15] where we proposed a PUF design based on ring oscillators (RO) suited for FPGAs which showed good results in terms of good statistical properties, simplicity and efficiency. The basic concept of the proposed PUF consists in different usage of ROs in order to generate PUF's output bits. The main idea in this proposal is to let two ROs forming a pair run simultaneously and count the number of their oscillations

using two counters of the same size (for example 16-bit counters). When one of the counters overflows, the ROs are stopped and the value of the counter that did not overflow is used for further processing. The result of the processing is a sequence of bits that is part of the final PUF output. The whole PUF output consists of bits obtained from multiple different RO pairs.

As mentioned before, one of the advantages of the proposed PUF design is the fact that it is easy to implement, area efficient, and additionally it does not require all ROs to be mutually symmetric, in contrast with the classical approach where all ROs are mutually symmetric and the PUF output is derived from the comparison of RO frequencies of various RO pairs. However, as it is shown in experimental results in Chapter 5, when the symmetric ROs are used in our design, it enhances stability of the proposed PUF design when the physical conditions are varying.

This thesis builds upon our previous work and one of its goals is to introduce the research area of PUFs. Various PUF constructions will be presented, mainly focusing on PUFs suitable for FPGAs. Since the proposed PUF design is based on ring oscillators, we will put emphasis on PUFs that use ring oscillators to generate PUF outputs.

The next goal is to analyse the proposed PUF design and discuss its properties. At first, we will describe the proposed PUF design from our previous work and introduce its main principle. Then we will discuss the advantages and disadvantages of this design and analyse its properties.

After the analysis, we propose further improvements of the design in order to enhance its statistical properties. We will present the results of the experiments to show the behaviour of the proposed PUF and the impact of the proposed improvements. All experiments presented in this work are targeted mainly on Digilent Basys 2 FPGA boards (containing Xilinx Spartan3E-100 CP132), and to verify that the design can be used on other types of FPGAs we performed additional measurements on Digilent Nexys 3 FPGA boards (Xilinx Spartan-6). Some of the findings were already published in [16].

Thesis outline

This work is divided into five chapters. The first two chapters are focused on literature research on the topic of Physical Unclonable Functions. Chapter 1 deals with PUFs in general. We introduce the topic of PUFs and present the properties which PUFs should meet. This chapter also presents the reader with applications where PUFs can be used.

The next Chapter 2 contains the literature research of existing PUF constructions that have already been proposed and it describes these different PUF constructions that exploit various sources of randomness and present their properties. This literature research is focused mainly on PUFs suitable

for FPGAs and since the proposed PUF design is based on ring oscillators, we put emphasis on PUF constructions that also use ring oscillators.

Chapter 3 presents the PUF design proposed in our previous work. We start with description of the main idea behind this PUF proposal and then we describe its properties and behaviour. We also analyse this PUF proposal and this analysis is then used in the next Chapter 4 where we propose some improvements in order to enhance the proposed PUF design and improve its statistical properties.

Finally, Chapter 5 is devoted to experiments and measurements and presenting their results. All of the measurements were performed on Digilent Basys 2 FPGA boards (Xilinx Spartan3E-100 CP132) and some of the measurements were also carried out on Digilent Nexys 3 FPGA boards (Xilinx Spartan-6). These measurements are associated with the analysis of the properties of the proposed PUF design and the improvements we proposed in Chapter 4. We also analyse the influence of temperature and voltage on our PUF design and evaluate the proposed countermeasures against varying physical conditions, which improve stability of the PUF outputs.

Chapter 1

Physical Unclonable Functions

Physical Unclonable Functions are now a very popular research topic especially in hardware security. Since this thesis is focused on PUFs on FPGAs, we introduce the reader the topic of PUFs. In this chapter we provide a description of PUFs in general and the properties we may require from them. Finally, we will also present possible PUF's applications.

1.1 Description of PUFs

Nowadays, we can find many scientific papers dealing with the topic of PUFs. For this reason, we can encounter multiple definitions of PUFs. In general, it can be said that PUF is a function which is realised within some physical system and expresses its inherent and instance-specific features [20], thus it is strongly similar to biometric features of humans.

The first description of a general concept of PUFs can be found in Pappu's dissertation thesis written in 2001 [26]. Pappu used a term POWF (Physical One-Way Function) and defined it as a function, which is easy to compute, but hard to invert, and the underlying physical system is difficult to clone and simulating the physical interaction is computationally demanding. The next used term denoting a new PUF construction was PRF (Physical Random Function) proposed by Gassend et al. [8]. But to avoid confusion with a term Pseudo Random Function (also abbreviated as PRF) they used the term PUF.

Term PUF is now widespread and various constructions and concepts which share a number of properties are called PUFs. Some of these constructions or concepts were proposed earlier than the term PUF was used, and therefore they were not denoted as PUFs from the beginning. In other cases, the proposal of some construction that could be qualified as PUF was made in other research areas than hardware security, where this term is unknown.

As the term PUF indicates, PUF is a function that is unclonable. The concept of PUFs is based on random variations arising during a manufacturing process, which causes each device to possess unique physical properties. These

physical properties are, for example, circuit delay or bias of memory cells to some certain value (0 or 1) after power-up. The variations of physical properties arising during manufacturing process are random, since they arise from the influence of random and uncontrollable effects.

Since PUF is a function, it should have some characteristics of functions; given an input we should obtain the corresponding output (in this text, we will often use term challenge instead of input and response instead of output). So it maps any input (challenge) to its corresponding output (response), forming challenge-response pairs (CRPs). However, it is not strictly a mathematical function, because a PUF can produce multiple different outputs for one input or even from several inputs it can produce one output. This behaviour may be caused by random variations which can be caused by various physical conditions [12]. A more fitting mathematical description of a PUF is a probabilistic function, where part of the input is an uncontrollable random variable [20].

In summary, PUF is a function that gives us, for a given challenge, a corresponding response. These responses may change in time in dependence on physical conditions; however, the responses should be similar enough so that we can recognize that the response we obtained belongs to the given challenge. At the same time, we require the PUF responses to be unique among different devices. It means that for the same challenge we obtain different response from each device. The difference between these responses should be sufficiently large, because based on these responses we can identify or authenticate the devices. The uniqueness applies also to responses from one device, but produced by a PUF for various challenges.

Both of these requirements (similarity of responses from one device and uniqueness of responses from various devices or different challenges) imply that we need the PUF responses to be both stable and unique. This is the main difference compared to TRNG (True Random Number Generator). The purpose of TRNG is to produce a sequence of bits that are random and even if we know a large sequence of bits, we should not be able to predict the following bits. In case of PUF, we need the PUF responses to behave like random sequences of bits from the perspective of population (devices) so that if we know a large number of responses from a large population of devices, we should not be able to predict a response for a given challenge from some unknown device. This condition also holds for various challenge-response pairs for one device. Therefore, responses should be random from the viewpoint of population of devices and also different challenges, but not for the same challenge applied to one device repeatedly. In summary, even though PUF and TRNG may have a common basis, because they both exploit physical properties of some device, the source of randomness for PUF and TRNG is very different. TRNGs exploit continuous real-time random behaviour of the hardware they are implemented on, while PUFs benefit from the randomness that occurs only once during the manufacturing process.

1.2 PUF's properties

This section provides an overview of the properties that are sensible for PUFs. Some of the presented properties are necessary and they define a PUF, while the others are only considered nice to have properties and they are not guaranteed for some PUF constructions [20].

Constructibility

A necessary condition for a PUF and all of its properties is that it is constructible. We can hardly discuss the remaining PUF's properties if they were not practically feasible. Constructibility requires the PUF proposal to be at least feasible within the laws of physics. However, from a more practical viewpoint, it is related to the cost of producing the PUF. There is also a big difference if we require for the produced PUF to have some particular challenge-response behaviour. In case of producing a *random* PUF without any specific requirement on its challenge-response behaviour, it should be easy to construct. Conversely, if we want to construct a specific PUF with defined challenge-response behaviour, then it can be infeasible to construct such PUF. This implies that this property is strongly related to physical unclonability.

Evaluability

A PUF is considered to be evaluable if for any random challenge it is "easy" to evaluate a corresponding response. Since PUF exhibits a challenge-response behaviour, this property is necessary for a PUF to achieve, because it would be difficult to discuss any properties of a PUF that is not evaluable. However, the "easiness" is context dependent. From a theoretical perspective this refers to polynomial time and effort. In practice it means that it is evaluable in terms of timing, area, power, energy and cost.

Reproducibility

For a given PUF and challenge on one chosen device, we should obtain the same response with high probability when the challenge is evaluated repeatedly. Reproducibility is one of the properties that puts constraints on a PUF's challenge-response behaviour. The PUF responses are influenced by varying physical conditions, therefore some errors may occur in the PUF responses when the PUF responses are obtained repeatedly for the same challenge. For this reason we consider a response with sufficiently small number of errors as the "same" response. Similarity of the PUF responses is evaluated based on the considered distance metric (usually Hamming distance of the bit strings that represent the outputs).

Uniqueness

As in the case of reproducibility, we consider one given PUF and challenge, but we observe the responses obtained from different devices and not only from one device. The responses resulting from evaluating the same challenge on different devices should be different enough (dissimilar) with high probability. Again, the similarity of the PUF responses is evaluated according to the used distance metric.

Physical unclonability

This property is crucial for PUFs. Since a PUF is based on random variations arising during manufacturing process due to the influence of random and uncontrollable influences, it is infeasible to manufacture two identical devices containing PUF that would exhibit the same challenge-response behaviour. The infeasibility is related to the physical and technical difficulties in manufacturing such pair of identical devices.

The property of physical unclonability has the security advantage that even the manufacturer, who may influence the manufacturing process, cannot break the uniqueness property, since there are uncontrollable influences which interfere with the manufacturing process. Therefore, it is not necessary to trust the manufacturer to be sure that every device containing PUF is unique with high probability, because it is implied by the physical unclonability of PUF.

When we combine this property with constructibility, we can say that it is easy to create a PUF with arbitrary and random challenge-response behaviour, but it is infeasible to create a PUF with a specific challenge-response behaviour.

Unpredictability

Numerous PUF applications rely on their challenge-response functionality which is to send some challenge and to obtain random (but corresponding to given challenge) response. In this sense, neither uniqueness nor physical unclonability sufficiently guarantee security. It is necessary to achieve unpredictability of the PUF responses to ensure the randomness of PUF responses for an adversary even if the adversary has already observed a number of challenges and their responses. Unpredictability means that the adversary should not be able to build a model based on observed challenge-response pairs that would predict responses for a new challenge.

Mathematical unclonability

In case of unpredictability, it was assumed that an adversary learned a limited number of challenge-response pairs which he uses to predict other responses.

This is usually the case when the adversary eavesdrops on communication when a challenge-response based protocol is used. However, a situation may occur, where the adversary has unlimited physical access to a device containing PUF and therefore can obtain as many challenge-response pairs as she is able to store. If the responses remain unpredictable even in such a case, one can say that the mathematical unclonability property is achieved. A necessary assumption for mathematical unclonability is the fact that the set of possible challenge-response pairs is so large (preferably exponential) that it is beyond the capacity of any adversary to store the whole challenge-response set.

Mathematical unclonability takes into account a stronger adversarial model, where the adversary has unlimited physical access to a PUF; hence it is the extension of unpredictability. It is obvious, that mathematical unclonability implies unpredictability.

True unclonability

There are already two different notions of unclonability defined in this section. They are physical and mathematical unclonability. Both of them have the same goal, which is to ensure that a PUF cannot be cloned, but different perspectives. Physical unclonability deals with the infeasibility of creating a clone of a specific device containing PUF with the same challenge-response behaviour. On the other hand, mathematical unclonability addresses only the cloning of the challenge-response behaviour of a chosen PUF, but not cloning the physical device itself. The true unclonability property is achieved when both physical and mathematical unclonability are met.

One-Wayness

One-wayness property is defined similarly to the definition of physical one-way functions proposed by Pappu [26]. A PUF exhibits one-wayness if it is evaluable and there exists no efficient inversion algorithm that finds a challenge based on a given response which produces similar response to the given one. This definition resembles to the definition of a one-way functions, but it takes into account the unreliability and uniqueness of PUFs.

Tamper-evidence

Tampering is the alteration of the physical integrity of some circuit, in this case a PUF. The intent is to modify the circuit's operation in an unauthorized and harmful manner. It is usually used to remove or bypass protection mechanisms to obtain confidential data, and is therefore a powerful attack against security implementations. Hence it is essential to detect tampering and to provide an appropriate reaction, such as clearing confidential data or blocking all functionality.

In order to detect any tampering attempt, a security system needs to have some tamper-evidence. It means that tampering will have an unavoidable and observable impact on the system. In the perspective of PUFs, tamper-evidence means that it is very hard to physically alter a PUF without any noticeable effect on its challenge-response behaviour. Ideally, the alteration would cause the PUF to become a completely different one.

1.3 PUF's applications

Due to its properties, PUF is suitable to be used for example for identification and authentication purposes. This section gives an overview of three possible applications of PUFs. First we describe device identification, then authentication and finally we will present the concept of cryptographic key generation.

1.3.1 Device identification

Device identification is the most basic application scenario of PUFs and it is an inherent feature of a PUF. It is used to identify some physical object (device). Just like we are able to identify any person by her fingerprints, we can identify any device based on its individual and unique physical characteristics which PUF uses for its functionality. Therefore it is very similar to a biometrical identification scheme.

Since there are errors present in the PUF responses and it is influenced by varying physical conditions, the PUF responses produced by a PUF on the same device will not be the same every time. However, for the identification purposes we do not have to worry about the errors in the PUF responses, provided that the PUF responses from the same device will be sufficiently similar and also different enough from the responses produced by PUF on other devices at the same time.

During the identification process, PUF generates a response, which is then compared to responses from various devices stored in a database. If the response is similar to one of the responses stored in the database and it is also different enough from the other responses, the identification process was successful. The similarity of two responses is usually determined by their Hamming distance. For a successful identification of a device based on its response, the following conditions need to be met:

1. For a given response from a specific device, a response in the database is found, such that the Hamming distance between these two responses is less than the chosen threshold.

2. The Hamming distance between a response from a given device and the responses from all other devices stored in the database is larger than the chosen threshold.

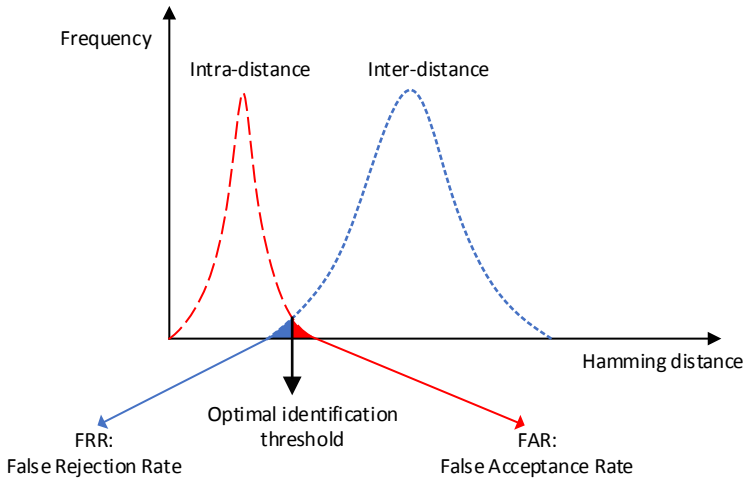


Figure 1.1: The principle of identification. The red curve represents the Hamming distance between the PUF responses from the same devices, while the blue curve shows the Hamming distance between the PUF responses from different devices.[22]

The ideal value of average Hamming distance between the responses from all devices is 50%. The principle of identification and determination of the identification threshold is shown in Fig. 1.1 [22]. This figure shows the curves of two metrics. They are Intra-device Hamming distance and Inter-device Hamming distance. Intra-device Hamming distance represents the Hamming distance of the responses generated by one device, while the Inter-device Hamming distance is the Hamming distance between the responses generated by different devices. In other words, Intra-device Hamming distance represents the bit error rate of the PUF responses and Inter-device Hamming distance shows how the PUF responses from various devices are different.

As Fig. 1.1 shows, if the curves were not overlapped, an errorless identification could be made by placing the identification threshold somewhere in the area between both curves. However, when the curves partially overlap, setting the identification threshold is a trade-off between false-acceptance rate (FAR) and false-rejection rate (FRR). The optimal choice of the identification threshold, minimizing the sum of FAR and FRR, is achieved by placing the threshold at the intersection of both histograms [22].

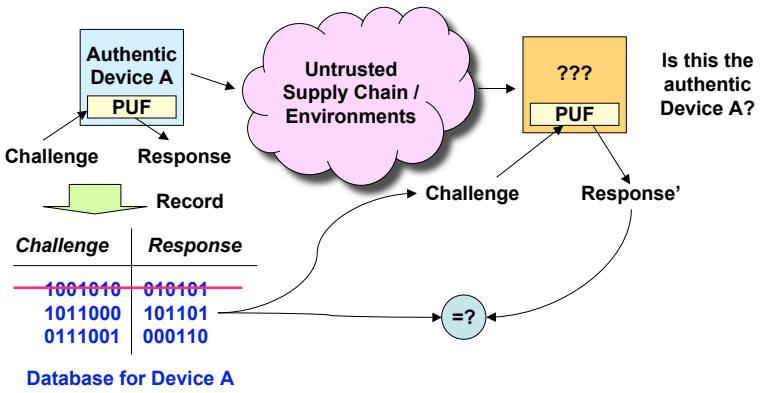


Figure 1.2: Authentication scheme using physical unclonable functions.[32]

1.3.2 Authentication

In case of authentication, any subject that wants to authenticate itself to the other party has to provide some sort of proof of its identity. For example, a subject can identify itself by some secret that only the subject knows. In addition, the subject has to demonstrate that it participated in the creation of the proof that confirms its identity.

Authentication using PUFs is realised based on challenge-response pairs. The authentication scheme benefits from the uniqueness and unpredictability of the PUF responses. One of the possible authentication schemes is shown in Fig. 1.2; it consists of two phases:

1. An ID of each subject is stored and then a sufficient number of challenge-response pairs is collected from its PUF. The challenges are generated randomly. The collected challenge-response pairs are stored in a database to the corresponding subject ID.

2. At the beginning of the authentication process, a subject identify itself by sending its ID (it does not have to be necessarily generated by its PUF). After the ID is found in the database, one of the stored challenge-response pairs is selected for the corresponding ID. The challenge is sent to the subject, which generates a response using PUF and sends back the response. If the response is similar enough (Hamming distance is less than the chosen threshold) to the response stored in the database (for the selected challenge), the subject is successfully authenticated. The challenge-response pair that was used for authentication is then deleted and never used again.

Since the challenges and responses are sent in an insecure manner, any third party can eavesdrop the communication and potentially use the captured challenge-response pairs. Therefore, there is a threat of a man-in-the-middle attack. To prevent this attack, all challenge-response pairs that were already

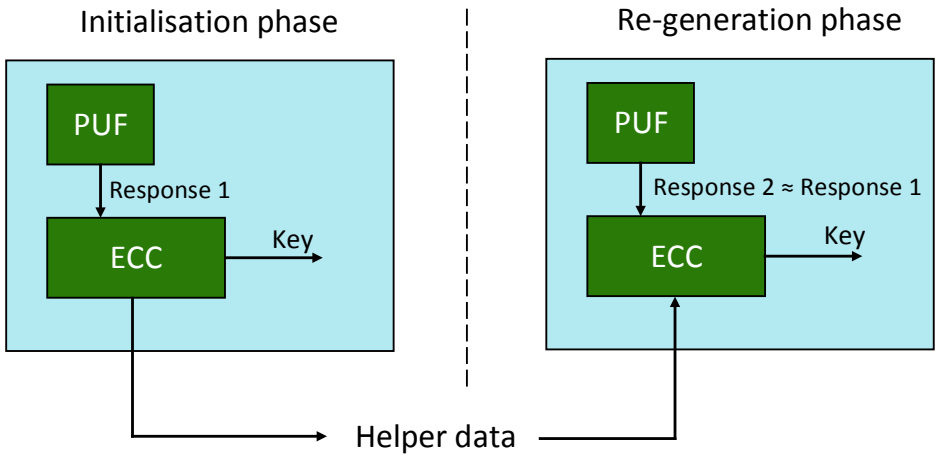


Figure 1.3: Principle of cryptographic key generation.

used in the authentication process are deleted. Moreover, it is important that the PUF is unpredictable: An adversary should not be able to predict a response of the PUF for a given challenge based on previously eavesdropped challenge-response pairs.

1.3.3 Cryptographic key generation

A number of security applications depend on cryptographic keys. These keys are usually stored in a non-volatile memory. This, however, raises a problem of how to store a cryptographic key in a secure manner so that the key is hidden from a potential adversary. Solutions to this issue are usually expensive and complex.

PUFs offer a cheap and efficient solution to the issue of secure storage of cryptographic keys. Instead of storing the secret key in memory, the keys are generated by a PUF at the moment they are needed. However, as mentioned before, the PUF responses tend to contain some errors and they are not the same when repeatedly generated due to varying physical conditions and random noise. Therefore, the PUF responses have to be stabilised before they are used as keys. This is usually achieved by the application of error correcting codes (ECC) that correct the wrong bits in the response. The generated key has to be the same on each generation, otherwise we would obtain a different result by deciphering some enciphered data even if there was only one erroneous bit in the key. The PUFs are able to generate random, unpredictable and stable keys when combined with error correcting codes.

In general, the process of key generation using PUFs is divided into two phases. During the *initialisation phase* a PUF generates the key and the error correcting code produces some *helper data* which are later used to correct the

PUF response. The helper data does not necessarily contain only information required by the error correcting code, but it can also contain some additional information needed by the PUF (for example configuration of the PUF). Since the helper data is usually public, it should not be possible to retrieve the key based on the content of helper data.

The second phase, called *re-generation phase*, re-generates the key when some application needs it. First the PUF generates a response, which is processed by the error correcting code. The error correcting code corrects the PUF response with the help of the helper data that were produced in the initialisation phase. After the correction of the PUF response, the same key as in the initialisation phase is obtained. The key generation process is shown in Fig. 1.3.

This method of key generation is only one of many possible methods. In other variants we may encounter e.g. the usage of hash functions which are applied on the PUF output after it is corrected by the error correcting code.

Chapter 2

PUF classification and construction

There are a lot of different PUF constructions. Based on their construction and operation principles, PUFs can be classified into various categories. One of possible classifications is to divide PUFs to *electronic* and *non-electronic* PUFs [22]. Non-electronic PUFs are, for example, Optical PUF and Paper PUF. It is also possible to classify PUFs based on their need to use some specialized or external equipment for their operation. Another criteria that can be used to distinguish PUFs is their source of randomness. Typical examples of sources of randomness are circuit delay or memory content after power-up.

In this chapter, we provide a brief introduction to possible classification of PUFs followed by description of various PUF constructions which is focused on *intrinsic* PUFs with two exceptions, which are Optical PUF and Coating PUF. The following Section 2.1 presents some of the possible classifications of PUFs and is followed by Section 2.2 that describes some PUF constructions; it does not present them all since there have been many PUF constructions proposed so far.

2.1 PUF classification

The PUF constructions were proposed for a large variety of technologies, materials and platforms. Therefore, we can classify PUFs based on the nature of their features (electronic components, glass, silicon integrated circuits) or even on their sources of randomness [20, 22]. Another classification can be the division of PUFs into *intrinsic* and *non-intrinsic* PUFs in dependence on the source of measurement and the origin of their random features [20, 22]. Ultimately, the PUFs can be classified based on the security properties of their challenge-response behaviour, i.e. *weak* and *strong* PUFs [10].

2.1.1 Electronic and non-electronic PUFs

The terms electronic and non-electronic corresponds to the nature of the components that are used for PUF and contribute to its randomness and uniqueness. These terms are not related to the processing methods or measurements which can be performed with the help of some electronic equipment.

The first class of PUFs are non-electronic PUFs. Their properties are based on non-electronic technologies or materials such as light scattering characteristics of an optical medium. The term non-eletronic in this case reflects the non-electronic physical basis of the PUF and not the way PUF responses are handled.

The opposite of non-electronic PUFs are electronic PUFs which exploit random variations in the electronic characteristics of electronic components or circuits. These characteristics are, for example, resistance, capacitance, delay etc. Furthermore, some of the electronic PUFs have their basic operations consisting of an analog measurement of some electric or electronic features [22] while other PUF proposals perform the measurements digitally. Therefore, the electronic PUFs may be distinguished also from this perspective.

A large subclass of electronic PUFs are silicon PUFs, which are the most popular in security solutions since they can be used in cryptographic implementations directly on an integrated circuit. Section 2.2 is focused on this type of PUFs.

2.1.2 Intrinsic and non-intrinsic PUFs

Another possible classification is based on the construction properties of PUFs. The PUFs are divided into intrinsic and non-intrinsic PUFs. According to [20, 22], two following conditions need to be met for a PUF in order to be classified as intrinsic PUF:

1. The PUF together with a measurement equipment should be embedded in the device and its evaluations are performed internally by the embedded measurement equipment.

2. Its random features are implicitly introduced during the manufacturing process.

In the following text, both of the conditions are discussed, since there are some practical and security advantages to intrinsic PUFs. Regarding the first condition, we can distinguish between two forms of a PUF evaluation, i.e. external and internal evaluation. In case of external evaluation, the measurements are performed using external instruments and the measured features have to be externally observable. Internal evaluation assumes that the necessary equipment used for measurement of random features of a device is embedded in the device together with the PUF. However, this implies a

possible disadvantage of an internal evaluation, because the embedded measurement equipment needs to be trusted since it might be impossible to verify the measurements externally.

One advantage of performing the evaluations internally is of a practical nature. Internal evaluations can be more accurate, since they avoid external influences possibly causing measurement errors. More importantly, the device containing PUF can evaluate itself without any restrictions since the necessary measurement equipment is embedded in the device.

The second advantage of internal evaluations is associated with security. When all evaluations are performed internally, the PUF response remains in the device and can be considered as internal secret that can be used for example as a key (in case of PUF used for key generation). This is useful when the PUF responses are used immediately for an embedded cryptographic applications.

The next discussed condition is related to the source of randomness which is measured during the PUF evaluation. The randomness used by PUF can be introduced implicitly to the device during the manufacturing process and form an integral and inseparable part of the PUF. The measured random features are caused by uncontrollable effects during the manufacturing process. The randomness may also be introduced by an explicit procedure during manufacturing process with the sole purpose of introducing random features that will be used by PUF. This condition implies that in case of intrinsic PUF, no extra manufacturing steps are required during the manufacturing process [22].

The advantage of implicit random variations is that there is no extra overhead and additional cost, since they are an inherent part of the manufacturing process. Introducing the randomness explicitly usually comes with extra cost. However, the main advantage of implicit random variations is in the perspective of security. The implicit randomness is caused by random variations arising during the manufacturing process and they are considered as undesirable because they may have negative impact on the manufactured device. Therefore manufacturers apply countermeasures against these process variations to reduce the effect of various random influences. However, it is technically impossible for the manufacturers to completely eliminate all random effects in the manufacturing process. This implies an interesting security advantage of PUF constructions based on implicit process variations: Even though the manufacturer has control over the manufacturing process, he is not able to control or eliminate the random features present in his manufactured devices, which are later used by PUF.

The intrinsic PUFs can be divided into two classes based on their basic operations. The two major classes of intrinsic PUFs that are also suitable for FPGAs according to their sources of randomness are delay-based and memory-based PUFs. A very common PUF design is based on SRAM (static random-access memory) and uses it as a source of randomness, since many electronic devices have embedded SRAM [11, 29]. This PUF is based on the content

of SRAM after power-up. However, some FPGAs initialise their memory after power-up, so all randomness is lost. That led to proposals of other memory-based PUFs such as Butterfly PUF [17], Latch PUF [31] and Flip-flop PUF [21].

Delay-based PUFs exploit the random variations in delays of logic gates and interconnects. One of the first delay-based PUFs is Arbiter PUF [18]. Another examples are Ring Oscillator PUF (ROPUF) [32, 9, 23] and Glitch PUF [33].

Optical PUF [26] and Coating PUF [34], introduced in the next Section 2.2, do not meet the conditions of intrinsic PUFs and they are some of the best known PUF constructions from the other class. The Optical PUF is non-intrinsic, because its evaluation is performed externally by observing the speckle pattern and also its random features are explicitly introduced by the random placement of the light scattering particles in an optical medium. The Coating PUF is classified as non-intrinsic, despite its ability to be evaluated internally, because its randomness is introduced explicitly during a manufacturing process by covering an integrated circuit by a protective coating with random dielectric particles.

2.1.3 Weak and strong PUFs

The last classification we will introduce in this thesis is based on the security properties of the challenge-response behaviour of the PUFs [10]. In this perspective, we can distinguish between weak and strong PUFs.

Weak PUFs can be considered as a digital fingerprint of some circuit. PUFs from this class usually have a very small challenge-response set or even only one challenge (or no challenge, but just some stimulation that starts the PUF evaluation to generate the fingerprint) in some extreme cases. An example of a weak PUF is a SRAM PUF, which is based on reading the memory content after device power-up. Each SRAM cell will have bias to some certain value caused by the manufacturing variability and this variability is random in the entire SRAM and also in the whole population of devices. The PUF response will be the memory content of the SRAM after power-up, but there will be no challenge, since the only challenge is powering the SRAM on. However, we can also consider some address of the SRAM as a challenge of the SRAM PUF. The fact that the challenge-response set is small implies that these challenge-response pairs must be kept secret. A typical application of weak PUFs is a cryptographic key generation.

The opposite of weak PUFs are strong PUFs. The main difference of strong and weak PUFs is the number of supported challenge-response pairs. Strong PUFs offer a large challenge-response set. The requirements for a strong PUFs are a large challenge-response set (large in this case means that ideally it should be exponential in the number of challenge bits), so that an adversary is not able to store all challenge-response pairs, and the unpre-

dictability of PUF responses even with the knowledge of a large number of challenge-response pairs. It is not feasible to build a model of the PUF based on the observed challenge-response pairs. If these requirements are not met, the PUF is classified as weak PUF.

Typically, the application of strong PUFs is authentication, where a device containing PUF is authenticated by a query with different challenge each time and comparing its response with the one stored in a database. After each usage of some challenge-response pair, this pair is deleted and never used again. However, it turned out that constructing a practical (intrinsic) strong PUF with strong security properties is a very difficult task [20].

2.2 PUF construction

The following list of PUF constructions is not complete, since there is a considerable amount of PUF constructions and we focus primarily on intrinsic PUFs.

2.2.1 Optical PUF

We can encounter one of the first PUF designs in Pappu's dissertation thesis written in 2002 [26]. At that time, the term *physical unclonable function* was not known and Optical PUF was classified as *physical one-way function.*

The main component of Optical PUF is a transparent optical medium (optical token) filled with a large amount of light scattering particles [2]. When a laser beam shines on the optical medium, a unique and random speckle pattern arises. The basic concept of Optical PUF is a very complex interaction between the laser beam and light scattering particles. The source of randomness in this PUF construction is the random placement of the light scattering particles in the optical medium during manufacturing process. The resulting speckle pattern is recorded and encoded (for example by Gabor hash) into a bit string representing a PUF response.

An exact position and angle of the laser beam is an input (challenge) to Optical PUF. Even a minute change in the relative orientation of the laser beam and the optical medium result in a completely different speckle pattern [22]. The basic principle of Optical PUF is depicted in Fig. 2.1.

2.2.2 Coating PUF

The concept of Coating PUF, introduced by Tuyls et al. in [34] consists of covering an integrated circuit with a protective coating. The coating material is filled with random dielectric particles. By random dielectric particles it is ment that they have random size, shape and location in the coating layer. Below the coating layer, a comb-shaped metal wire sensors are used to measure the local capacitance of the coating [27] as shown in Fig. 2.2. Measuring

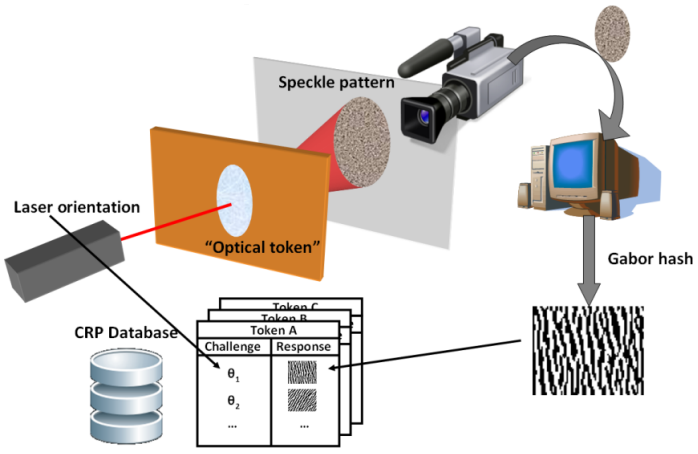


Figure 2.1: Basic operation of an Optical PUF. The setting of laser beam serves as input to Optical PUF. The laser beam shines on the optical token, which is a transparent material filled with light scattering particles. After shining on the optical token, we obtain a unique speckle pattern which is then encoded by Gabor hash into a bit string used as a response of the Optical PUF.[22]

the Coating PUF from the outside gives different capacitance results since the measurements are very sensitive to the precise locations of the dielectric particles.

Coating PUF does not rely on the random effects of manufacturing variability, but it uses the random elements explicitly introduced by a passive dielectric coating sprayed directly on the top of the sensors. Coating PUF offers strong protection against physical attacks such as tampering since the protective coating is opaque and chemically inert. By any physical intervention into the protective layer a change in the capacitance of the layer occurs, resulting in a completely different behaviour of the Coating PUF. Note that implementing a Coating PUF requires an additional manufacturing step, but it is still very cheap to produce [27].

2.2.3 SRAM PUF

SRAM (static random-access memory) is a static memory based on bistable flip-flops that are used to store data. Fig 2.3(a) shows a SRAM cell, logically constructed as two cross-coupled inverters. This circuit has two possible stable values (0 and 1) which represent the binary value stored in the cell [20].

The principle of SRAM PUF operation is based on the memory content after power-up. Since the preference of each memory cell cannot be influenced and their preference is random and independent, they are a suitable source of randomness for PUF. Large SRAM memories are capable of storing many

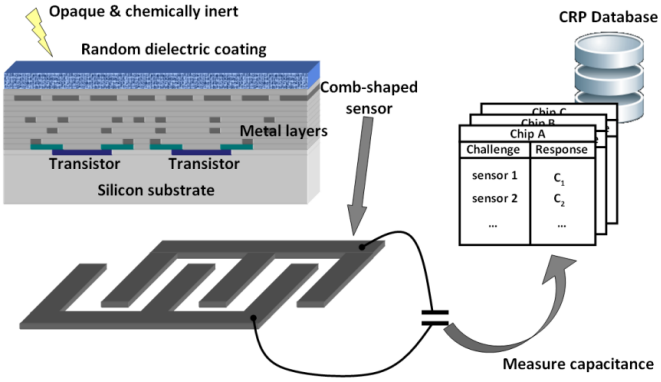
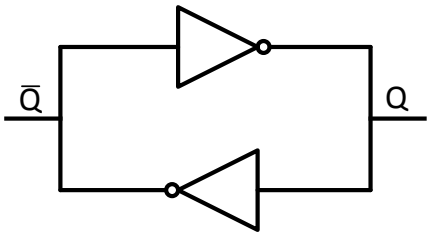
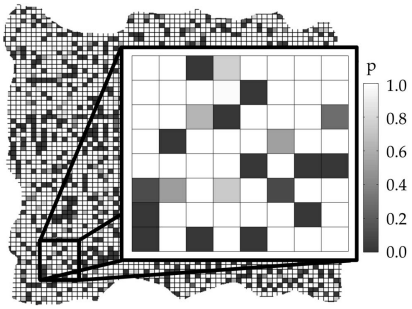


Figure 2.2: Coating PUF.[22]



(a) SRAM cell.[22]



(b) SRAM content. The shade of grey illustrates the probability of state 1.[11]

Figure 2.3: Concept of SRAM PUF.

kilobits or megabits that could be used for PUF. A memory address in SRAM PUF can be considered as the challenge for PUF.

A key property of an SRAM cell in terms of the PUF characteristics is bistability [28]. Each SRAM cell prefers different state after power-up. Some memory cells have bias to binary 1, while other cells have bias to binary 0. However, some of the memory cells do not have bias to any of the two binary values. The distribution of these three types of SRAM cells over the whole memory is random [22]. The memory cells with strong bias to one of the two binary values which in most cases stabilise in the same value are considered as stable memory cells. On the other hand, memory cells with no real preference which usually have different states over time are called unstable cells. The bias toward some value of memory cells after power-up is caused by random physical mismatches in the memory cells that originate from the manufacturing process.

The stable memory cells allow us to identify various devices when used for SRAM PUF while the unstable memory cells cause the errors in the PUF

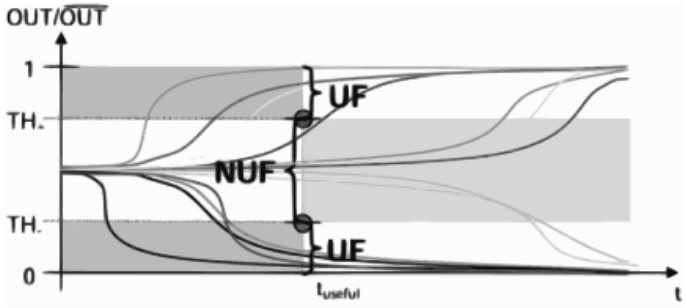


Figure 2.4: Measurement of decision time (UF: useful, NUF: not useful).[28]

output. Fig. 2.3(b) shows an example of SRAM map with probabilities of binary value 1 occurring in the corresponding cell. Ideally, we want each cell to be 100% stable. If we want the PUF output to be stable, we would need to perform measurements repeatedly in order to select only the stable SRAM cells. However, such measurements would increase the manufacturing costs and they would have to be performed at varying physical conditions such as varying temperature or voltage.

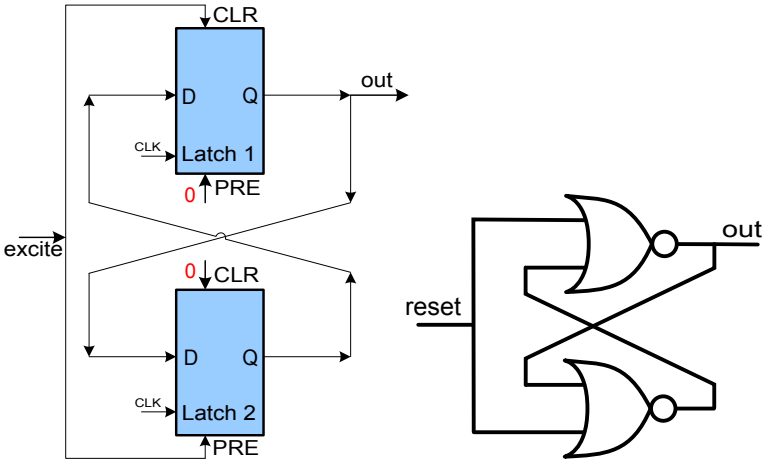
Another option of obtaining stable output is to select cells that settle in their final state faster after power-up [28]. The concept of this approach is shown in Fig. 2.4. All the cells with the decision time under t_{useful} and lie under a lower threshold or above upper threshold are considered to be useful. All other cells are not used.

2.2.4 Butterfly PUF

SRAM PUF is often impossible to implement on some FPGA platforms because the SRAM is initialised to predefined values. Another disadvantage of SRAM PUF is the fact that to generate the PUF output, the SRAM needs to be read after power-up before the memory is overwritten and the randomness is lost. These drawbacks were the main motivation for the proposal of Butterfly PUF [17], which is based on cross-coupled circuit and can be implemented on any FPGA.

The Butterfly PUF concept consists of simulating the SRAM PUF behaviour after power-up, when the SRAM cells stabilise on particular values. The basic building element of Butterfly PUF is a circuit made of two cross-coupled latches, simulating the SRAM cell. This structure can be forced into an unstable state after which the structure converges back to one of the possible stable states.

To ensure a proper behaviour of Butterfly PUF, it is necessary to achieve the best possible symmetry of the interconnects between the two latches [25] as shown in Fig. 2.5(a). The interconnects between the outputs Q and the inputs D between both latches have to be symmetric. When set to high, the signal



(a) Structure of Butterfly PUF cell.[17]

(b) Logical circuit of a Latch PUF cell.[22]

Figure 2.5: Butterfly and Latch PUF.

signal starts the Butterfly PUF cell operation. When the input signals PRE (preset) and CLR (clear) are set to 1, the value of the output Q is changed to 1 or 0. Signal CLK (clock) transfers the input signal D to the output Q. Signal CLK is always set to high in order to simulate a combinational loop. By setting the signal *excite* to 1, the structure is forced into an unstable state because of the cross-coupled latches where the output Q of each latch is transferred to the input D of the other latch. After a few clock cycles the *excite* signal is set to low and the Butterfly PUF cell will stabilise in one of the possible states. The resulting state depends on the slight delay differences of the interconnects between the two latches, which will be different among various devices and positions on the FPGA [17].

2.2.5 Latch PUF

A method of identifying integrated circuits based on latches realised by two cross-coupled NOR gates was introduced by Su et al. [31]. A simple circuit used as a latch for Latch PUF is shown in Fig. 2.5(b). This concept is very similar to SRAM PUF; however, the SRAM PUF cells are in an unstable state at the beginning before they settle on the resulting value. Latches in case of Latch PUF are in a stable state and they are brought into an unstable state using *reset* signal. The resulting value will be derived based on the random internal mismatches of the electronic components.

The advantage of Latch PUF compared to SRAM PUF is that, similarly to Butterfly PUF, it does not depend on the device power-up but we can obtain the PUF's output whenever it is needed. This implies that when the device is powered up, it is not necessary to store the PUF output – we can generate it

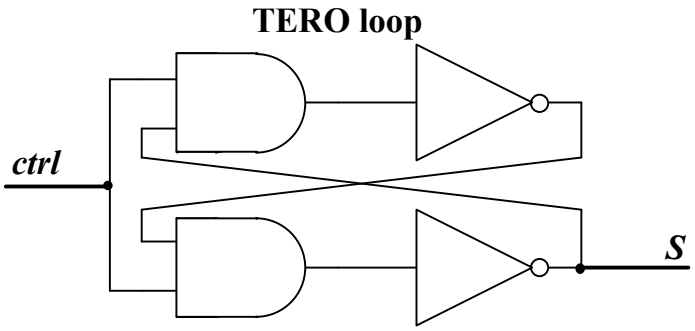


Figure 2.6: TERO loop structure.[1]

anytime.

2.2.6 Flip-flop PUF

As in the case of SRAM PUF, Flip-flop PUF depends on the device power-up, but it uses D-flip-flops instead of SRAM cells. Flip-flop PUF was proposed by Maes et al. [21] as a replacement of SRAM PUF on FPGA boards.

2.2.7 TERO PUF

A new PUF structure which exploits the oscillatory metastability of cross-coupled elements was proposed by Bossuet et al. [1] and then extended by Marchand et al. [24]. The source of the entropy in this proposal is the mean number of oscillations of the oscillatory circuit before the oscillations stop and the structure is stabilised. This PUF is based on transient effect ring oscillator (TERO) cells, therefore it is called TERO PUF. TEROs were originally proposed by Varchola and Drutarovsky [35] for TRNG designs.

The basic building element of this design is a TERO loop structure. It is composed of an SR-flip-flop implemented as two AND gates and an even number of inverters. There are usually two inverters used for the TERO loop, but the loop can be extended with more inverters in order to extend the oscillations.

The oscillatory metastability in this design is achieved by connecting the S and R inputs of the SR-flip-flop to the *ctrl* signal. An example TERO loop structure is shown in Fig. 2.6. When the signal *ctrl* is set to high, the structure is brought into an unstable state and causes transitory oscillations in the loop if certain conditions are fulfilled [35]. Theoretically, if the loop was absolutely symmetrical, the oscillations would never stop. However, the TERO loop structure oscillates for a short period of time before it stops due to the asymmetry in the time delay of both halves of the loop.

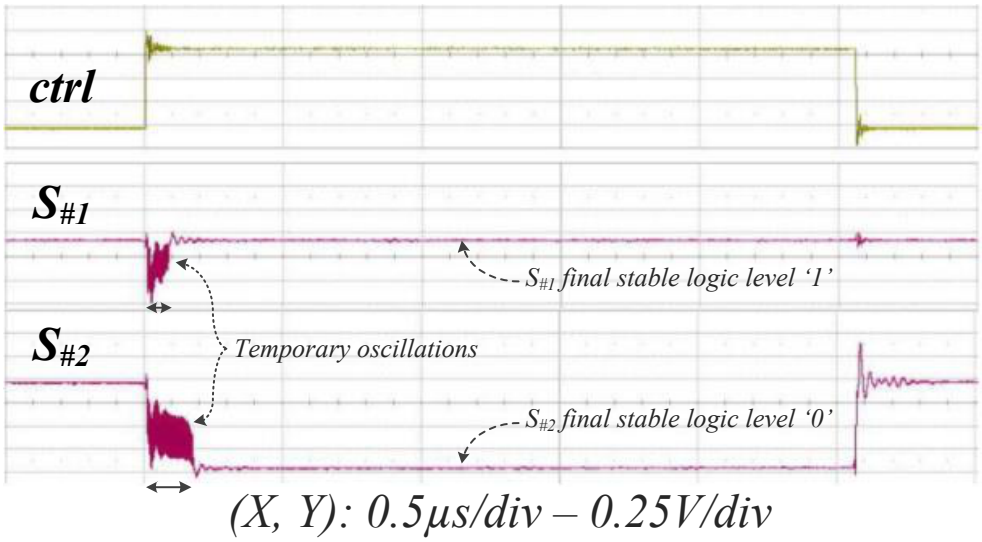


Figure 2.7: Electrical behaviour of the two TERO loops. Signal *ctrl* is an input signal to both of the TERO loops that causes temporary oscillations of the TERO loops. Signals $S_{\#1}$ and $S_{\#2}$ are outputs of the TERO loops.[1]

An example of behaviour of two TERO loop structures is shown in Fig. 2.7. It shows an input signal *ctrl* and output signals ($S_{\#1}$, $S_{\#2}$). The *ctrl* signal for both TERO loops is forced to logical value 1. The rising edge of the *ctrl* signal causes temporary oscillations in both of the TERO loops as can be seen at the $S_{\#1}$ and $S_{\#2}$ signals. In Fig. 2.7 it is clearly visible that the number of oscillations of both TERO loops is different, and when the loops are stabilised, the resulting value for both loops is also different.

In [1] Bossuet et al. measured the number of oscillations using 8-bit counters and the measurements were performed 2^{18} times for each TERO loop. Then the mean value of the number of oscillations of each TERO loop was used for further processing. Finally, the mean values were subtracted in a pair-wise manner and particular bits were selected from the resulting binary value and used for the PUF output, based on their statistical properties. The described TERO PUF architecture is shown in Fig. 2.8.

2.2.8 Arbiter PUF

Arbiter PUF uses the delay difference of logic gates and their interconnects as a source of randomness. We can find its initial proposals in the works of Lee [18] and Lim [19]. The basic concept of Arbiter PUF is a digital race on two paths on a circuit which have to be mutually symmetric. Both of the paths end in *arbiter* that decides which one of the two paths won the race, or in other words, which path had a smaller delay. Based on the result of the race, the arbiter generates one output bit for PUF. Arbiter is a logical circuit

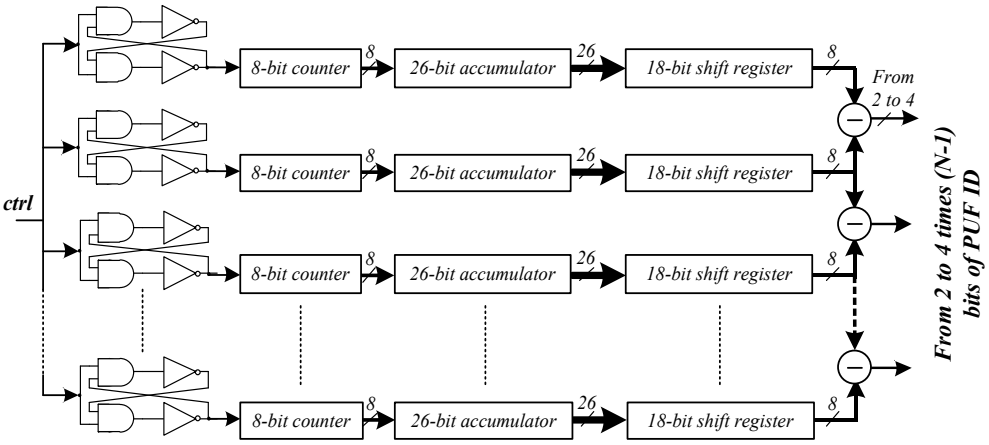


Figure 2.8: TERO PUF architecture.[1]

used to determine which path is faster. To ensure that the result is random and unpredictable, both of the paths have to be mutually symmetric, meaning that their intended delay is the same. When this condition is satisfied, the path with smaller delay will be dependent on the random delay variations in individual gates and their interconnects which arise during the manufacturing process.

Fig 2.9 shows the basic Arbiter PUF scheme according to [18]. Both of the paths are implemented as a series of switch components. The switch component interconnects its two input signals to the output ports with different configurations depending on the control bit (b_i). For $b_i = 0$ the paths go straight through, while for $b_i = 1$ the paths are crossed. The switch component logic can be implemented for example as two multiplexers. At the end of both of the paths, there is an arbiter detecting the first rising edge and which can be realised as a latch.

To obtain 2^n possible configurations of this circuit, we need n control bits configuring n switch components forming a chain of switch components. The configuration of this circuit is determined by the challenge of the PUF, which represents n control bits. The result of the operation of the whole circuit for one configuration is one output bit. To obtain more output bits, there are two possible solutions that can be combined together [3]. The first option is to implement more of these circuits which will all use the same challenge (configuration) when generating the PUF output. The second possibility is to send multiple challenges to only one circuit and chain the output bits to form the PUF response.

The Arbiter PUF output should be influenced solely by the random variations in delays of the individual paths. This can be achieved if both of the following conditions are met [20]:

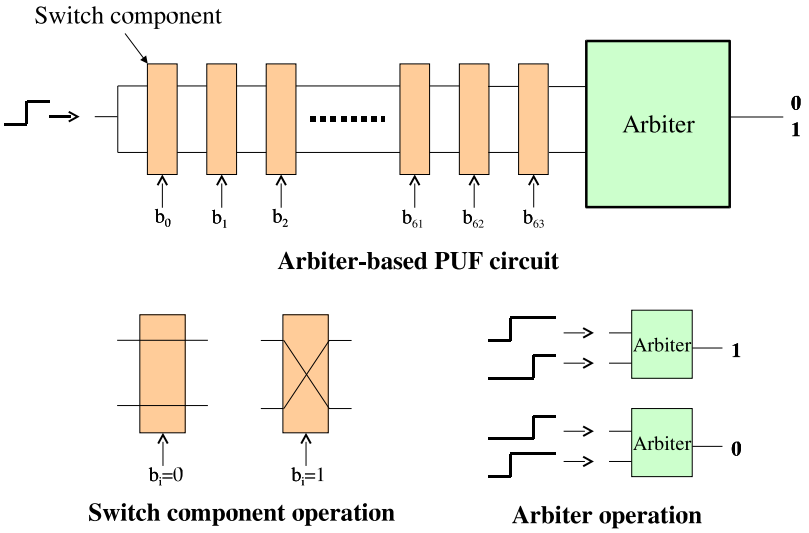


Figure 2.9: Basic Arbiter PUF scheme.[18]

1. Each pair of paths is designed to be perfectly symmetrical, thus any difference in delay is caused solely by the manufacturing process variations.

2. The arbiter circuit is absolutely fair, so it does not favour one of its inputs over the other.

If any of the conditions is not met, the Arbiter PUF is biased resulting in a lower uniqueness of its responses. Designing Arbiter PUF that satisfies both conditions is not a trivial task since it requires a low-level control over the implementation. In some technologies, such as FPGAs, it is not possible [20]. However, some unbiasing techniques can be used when bias is unavoidable.

2.2.9 Ring Oscillator PUF

A numerous PUF constructions based on ring oscillators (RO) have been proposed to this day, but it is not the goal of this paper to introduce them all. This section describes the main principle of Ring Oscillator PUFs (ROPUF) that have been proposed so far. We put emphasis on this PUF construction, because our proposed PUF design is also based on ROs.

Measuring a delay

Since ROPUF is a delay-based PUF, it exploits the random variations in delays of logic gates and their interconnects. The method that is used in ROPUFs to measure delays is to use some delay circuit and make it a self-oscillating

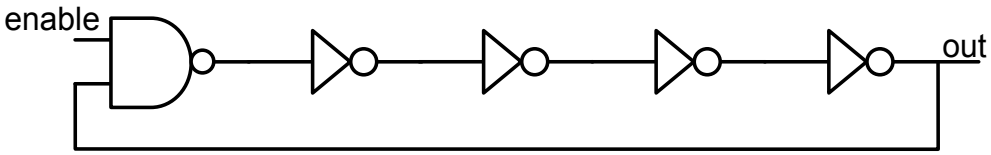


Figure 2.10: A basic ring oscillator composed of one NAND gate and four inverters forming a combinational loop.

loop. This can be achieved by inverting the output of the delay circuit and feeding it back into the delay circuit's input [8, 9]. These oscillating circuits are usually called ROs. An example of a ring oscillator is shown in Fig. 2.10.

The random variations in delays are reflected in the measured frequencies of ROs. We can simply measure ring oscillator's frequency by using counters and some reference clock where we know the frequency. This frequency can be determined by the number of ring oscillator's oscillations that are recorded by a counter at a certain time specified by the reference clock. From the resulting value in the counter we can easily calculate the ring oscillator's frequency.

Frequencies of each RO obtained this way can be used for PUF. These frequencies are used depending on the particular ROPUF proposal. In some ROPUF designs it is not necessary to know the particular frequency of each RO; we can use the resulting value in the counter.

Ring Oscillator PUF constructions

The first type of ROPUF was proposed by Gassend et al. [8, 9]. The measured frequencies of equal ROs on different devices shows sufficient variation to act as a PUF output. However, the influence of environmental conditions on the frequency of ROs is significant and some additional technique to compensate these influences is required. Gassend et al. [8, 9] proposed a technique called *compensated measuring*. The main idea behind compensated measuring is that environmental changes will affect the frequencies of ROs approximately the same way, therefore a ratio of measured frequencies of RO pairs can be considered as the eventual PUF output.

This ROPUF construction proved to be effective in compensating the environmental changes. Nevertheless, this construction has some drawbacks. Since their ROs are based on the same delay circuit as the basic Arbiter PUF, their PUF is vulnerable to modelling attacks and some countermeasures have to be made. In addition, the result of frequency ratios in case of compensated measuring are real values and cannot be used directly as a bit string, hence they have to be processed in an appropriate way to get a proper PUF output.

Another ROPUF construction proposed by Suh and Devadas [32] is shown in Fig. 2.11. Their ROPUF design consists of n symmetric ROs that are connected to two multiplexers. Each of the multiplexers selects one of the ROs according to the *input* signal and connects its output to the counter. Both

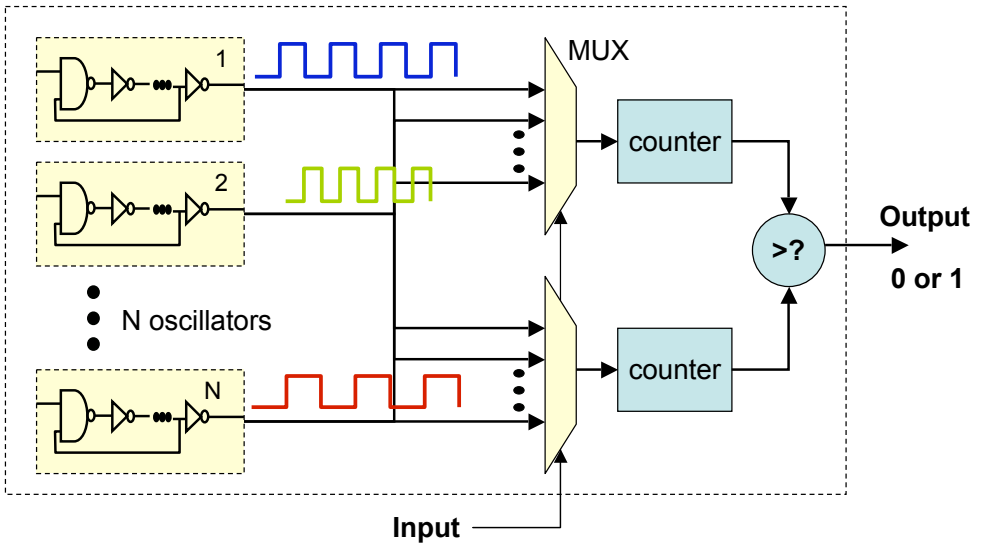


Figure 2.11: Ring Oscillator PUF design.[32]

counters count oscillations of the selected ROs for a fixed time interval. The resulting values in both counters are compared and one bit of the PUF output is produced based on the result of the comparison. Since one comparison produces only one bit of the PUF output, this whole process has to be repeated several times with different selections of ROs to produce the complete PUF output.

It is required that all of the ROs in this ROPUF construction be symmetric to each other in order to assure that the comparison results are unpredictable. Due to the symmetry of ROs the differences in their frequencies are completely dependent on random variations in delays due to manufacturing process variations. The frequency comparison can be considered another form of compensated measuring to eliminate the influences of environmental changes. Suh and Devadas [32] also proposed a technique called *1-out-of-k masking* which reduces the number of possible comparisons in order to get a more stable output. This technique is based on the selection of one RO pair with the biggest difference in their frequencies from k oscillators.

One of the drawbacks in this ROPUF design is the fact that the number of possible comparisons is limited if we want the bits in the PUF output to be independent. The maximum number of comparisons with n ROs that we can perform using this design is $\binom{n}{2} = \frac{n(n-1)}{2}$. However, the entropy of the design is less than $\binom{n}{2}$, because bits obtained in this way are correlated. For example, if RO A is faster than RO B and RO B is faster than RO C, it is clear that RO A will be faster than RO C [32].

To determine the maximum entropy of this design, in other words the maximum number of independent bits generated by pair-wise comparisons,

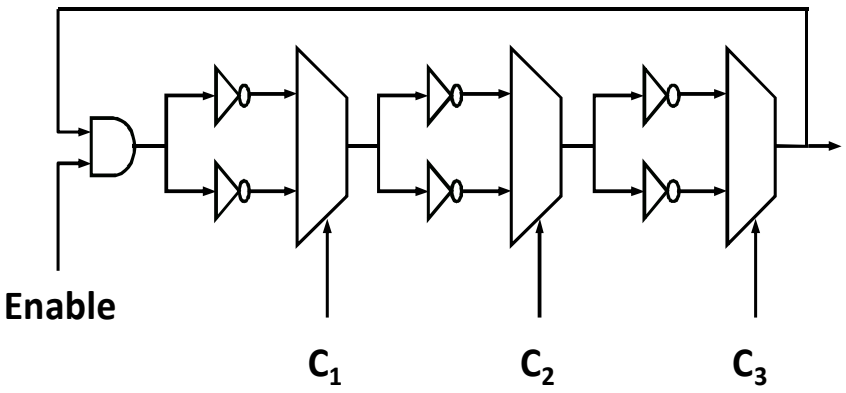


Figure 2.12: Configurable ring oscillator.[23]

we have to consider all possible orderings of n ROs. There are $n!$ possible orderings of ROs based on their frequencies and if the orderings are equally likely, the entropy of this design will be $\log_2 n!$.

An easier way to obtain independent bits in the PUF output is to use each RO only once for comparison, thus the number of output bits would be $\frac{n}{2}$. Combined with the technique 1-out-of-k masking, the number of output bits would be significantly reduced.

Another technique to increase the stability of ROPUF output was proposed by Yin and Qu [39]. The ROs are divided into mutually exclusive groups (no RO is in two groups at the same time) and the frequency comparison is performed only between the ROs in the same group, where a high stability of the results of the comparisons is guaranteed. The division of ROs into groups is performed by the proposed algorithm called LISA (Longest Increasing Subsequence-Based Grouping Algorithm). For each RO, measurements are executed at various conditions and the minimum and maximum frequencies that were measured are stored. Using the LISA algorithm, ROs are divided into groups in such manner that the differences of the minimum and maximum frequencies between any RO pair were larger than some selected threshold value. Using this technique, we can obtain a very stable output and also longer output from the same number of ROs than using 1-out-of-k masking.

Maiti and Schaumont [23] used the same design as Suh and Devadas together with the technique of 1-out-of-k masking to increase the output stability, but the design is now based on configurable ROs instead of basic ROs. They consider the most stable configuration out of k possible configurations of one RO pair, not the most stable RO pair. The advantage of configurable ROs is that they allow more efficient utilization of resources.

The configurable RO proposed in [23] is shown in Fig. 2.12. The classical RO composed of five gates (one NAND and four inverters) implemented on FPGA occupies almost the whole CLB (configurable logic block) on Xilinx

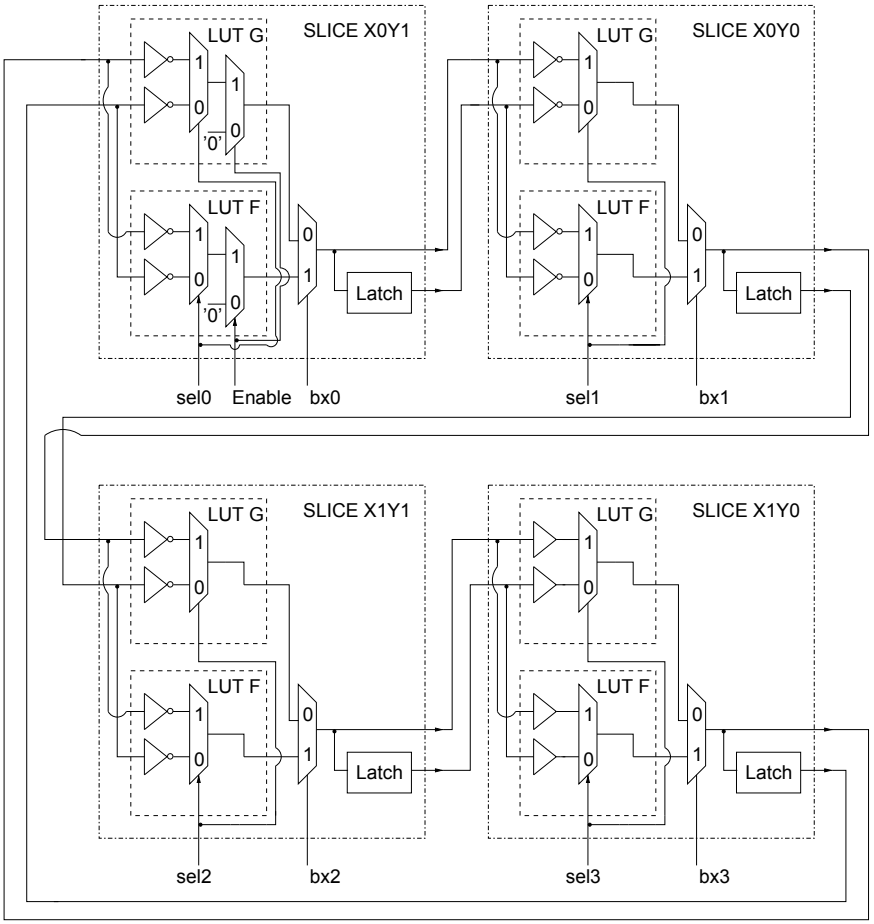
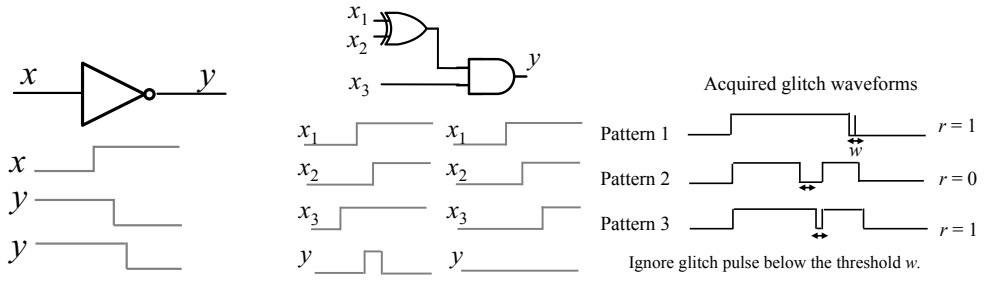


Figure 2.13: Configurable ring oscillator capable of 256 different configurations in one CLB on FPGA. In this case, CLB consists of four *slices* and each slice contains two LUTs (Look-Up Table). The signals *sel* and *bx* are control signals for multiplexers that determine the configuration of the RO.[38]

Spartan-3E. The configurable RO shown in Fig. 2.12 consists of six inverters, one AND and three multiplexers. It all fits into a single CLB occupying basically the same area as a common 5-staged RO. The configuration of the RO is set by the control bits that are used to control the multiplexers. In this case, there are three control bits, hence eight possible configurations of the RO. Since the PUF output is derived from the frequency comparisons, it is necessary that the configurable ROs are symmetrical.

Another proposal of a configurable RO on Xilinx Spartan-3E was presented in the work by Xin et al. [38]. They extended the design proposed by Maiti et al. [23]. The configurable RO still fits into a single CLB on FPGA, but now it is capable of 256 various configurations. This design uses more



(a) Behaviour of a simple circuit. (b) Behaviour of a more general circuit. (c) Conversion of the acquired waveforms.

Figure 2.14: Substance of Glitch PUF.[33]

multiplexers and also latches as shown in Fig. 2.13. The latches are used as an additional delay unit. This configurable RO is configured by eight control bits, therefore it is possible to achieve 256 possible configurations. Again, the ROs have to be mutually symmetric and also it is necessary to use the ROs with same configuration for the comparison, otherwise, the result of the comparison would not depend on the random delay variations.

2.2.10 Glitch PUF

Suzuki and Shimizu proposed a new PUF construction [33]. They considered a possible behavioural difference of the same logic circuits with different delays. Fig. 2.14(a) shows a difference in behaviour of a simple logic circuit. There is a time difference between output changes from an input change. However, this delay is variable since it depends on the random variations of gate delays acquired from manufacturing and also largely on the temperature and voltage. A behaviour of a more general circuit that performs AND and XOR operation on multiple inputs can be seen in Fig. 2.14(b). We can see that a transient state of an output signal, called a glitch, occurs; this is caused by the delay difference between input signals. The glitch shown in Fig. 2.14(b) occurs at the output of XOR due to differences between transition times of input signals x_1 and x_2 after all input signals were changed from 0 to 1. Only in the case that signal x_3 reaches the AND gate before the glitch, the glitch propagates to the AND output. In the other case, the output remains unchanged since the glitch did not propagate to the output.

According to [33], Glitch PUF consists of the three following steps. The first step is data input to a random logic. The next step is the acquisition of glitch waveforms at the output of the logic. The final step is the conversion of the acquired waveforms into PUF response bits. An example of such conversion is shown in Fig. 2.14(c), where a glitch with width less than a threshold w is ignored.

Chapter 3

Description and properties of the proposed PUF

So far, this work has focused on the state of the art. In previous chapters, we introduced PUFs to the reader and we presented some of the major PUF constructions that were proposed so far. This chapter presents our proposed PUF construction, which is based on ring oscillators and according to the classifications described in previous Chapter 2, this PUF construction may be classified as delay-based intrinsic PUF since it exploits the random variations in delays of logic gates and their interconnects that affect the frequency of ring oscillators.

The motivation that led to the proposal of our PUF design was to design a PUF that would be easy to implement, area efficient and also suitable for FPGAs. There are already numerous PUF constructions proposed for FPGAs, however, some of these are more suitable for FPGAs than others. A popular PUF construction is SRAM PUF, but a lot of modern FPGAs initialise their memory content and all of the randomness necessary for PUF is lost.

The main representative of delay-based PUFs is Arbiter PUF. Its advantage is its simplicity, low power consumption and its implementation is inexpensive and suitable for resource-constrained platforms such as RFIDs. However, Arbiter PUF is not well suited for FPGAs since it depends on the symmetry of the two paths for which the delay is measured. In case of FPGAs, it is impossible to achieve absolute symmetry of both paths; we can only try to design the two paths so that they are as symmetric as possible. Therefore, Arbiter PUFs are usually not implemented on FPGAs.

A more suitable PUF for FPGAs is Ring Oscillator PUF, which has the same source of randomness as Arbiter PUF. Instead of having two symmetric paths for which we compare their delay, the delay can be measured by ring oscillators whose frequency will be affected by the random variations in delays. A classical approach of using ROs for a PUF is to compare frequencies of selected RO pairs. However, this approach requires the ROs to be mutually

symmetric in order to generate unpredictable result of the comparison. Implementing symmetrical ROs is not as difficult task as in the case of Arbiter PUF, but it is still an additional overhead to designing the PUF.

A considerable issue of the classical approach is the appropriate selection of RO pairs so that the bits in the PUF response are unpredictable and they are not correlated. The requirement for the most efficient usage of ROs also leads to more complicated designs. Therefore, each RO is usually used for only one comparison, which simplifies the design but a lot of potential RO pairs is not used. However, there are some techniques presented in Section 2.2.9 that improve the efficiency of ROPUF on FPGAs.

In our PUF proposal, we were inspired by PUFs based on ring oscillators. Our goal was to propose a PUF that would be easy to implement and effective. In our proposal, we use a different technique to generate the PUF output than frequency comparison, which is used in the classical approach and requires the ROs to be symmetrical. The PUF output will also be obtained based on the selected RO pairs, but the problem of selecting particular RO pairs is no longer present and on top of that, more bits for the PUF output will be gained from each pair. This makes it possible to produce a longer PUF output using fewer ROs.

The following two sections detail our PUF proposal. The first Section 3.1 describes our proposed PUF design. The next Section 3.2 explains the behaviour of our proposed PUF design and discusses its properties.

3.1 The ring oscillator based PUF proposal

The basic building element of the proposed ROPUF design is a five stage RO (one NAND, four inverters). Instead of measuring frequency of each RO using a reference clock, we select one RO pair and count their oscillations simultaneously using two counters. As soon as one of these counters overflows, the measurement is stopped. The resulting value in the counter that did not overflow is used for further processing. This approach is shown in Fig. 3.1.

The proposed method implies that if we knew the exact frequencies of the ROs during measurement, we could determine the resulting counter value (in case of 16-bit counters) that is later processed as follows:

$$Counter\ value = \frac{f_2}{f_1} \times 2^{16}, \tag{3.1}$$

where f_1 is the frequency of the faster RO and f_2 is the frequency of the slower RO.

When implementing the logic for detecting overflow of one of the counters and stopping the other one, the routes between them may have different delays and in the meantime, before the counter is stopped, it can perform some additional steps. But since these two routes are the same for all RO pairs

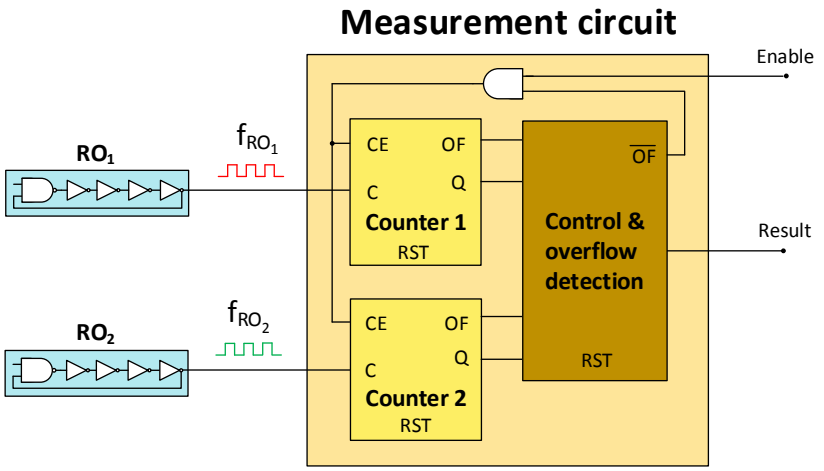


Figure 3.1: The method of measuring the number of cycles of ring oscillators in the proposed ROPUF design.

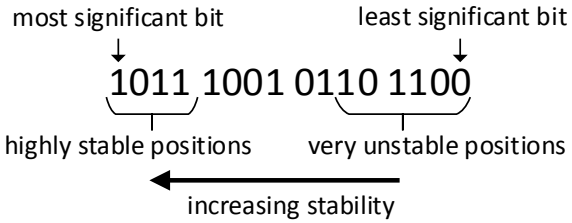


Figure 3.2: The example behaviour of positions' stability in a 16-bit value.

and for all FPGAs, it will only increase the resulting counter value by some constant offset. It will be shown later that this offset is either 0 or 1.

The counter values obtained in this way are used for PUF without any modification. Since these values are represented in binary code, we can use the appropriately selected part of each binary number for PUF output. It can be assumed that if we make multiple measurements of one RO pair, bits that are close to the least significant bit (LSB) will vary a lot, for example due to environmental changes. On the other hand, bits close to the most significant bit (MSB) will be stable and the environmental changes will have almost no influence on them. The closer we get to the MSB, the more stable these bits will be. The example of described behaviour of measured values is shown on a 16-bit value in Fig. 3.2. A concept of choosing suitable bits for PUF from the counter values was also presented in the work by Bossuet et al.[1].

3.1.1 Bit stability

The stability $s_i(RO)$ of bit at position i from a value measured using one particular RO pair is determined as follows:

$$s_i(RO) = \begin{cases} P(b_i = 1) & \text{if } P(b_i = 1) \geq 0.5 \\ 1 - P(b_i = 1) & \text{if } P(b_i = 1) < 0.5, \end{cases} \tag{3.2}$$

where $P(b_i = 1)$ stands for the probability of occurrence of 1 at position i and is defined as:

$$P(b_i = 1) = \frac{1}{k} \sum_{j=1}^{k} b_{j,i}, \tag{3.3}$$

where k is the number of executed measurements and $b_{j,i}$ indicates the i-th bit of the j-th measured value.

It is obvious, that each RO pair will have a different stability at various positions of measured values since the ROs in this design are no longer mutually symmetric. For this reason, if we want to perform a suitable selection of positions for the PUF output for all RO pairs, we have to determine the average stability of each position. Provided we have n RO pairs, the average stability s_i of position i is determined as:

$$s_i = \frac{1}{n} \sum_{j=1}^{n} s_j(RO_j), \tag{3.4}$$

where RO_j is the j-th pair of ROs.

Based on the average stability s_i of each position, we can decide which bits are suitable for PUF output. Ideally, we would like the stability s_i of selected bit positions to be equal to 1, but we might not be able to achieve such stability, either at all or only at a few bits that are closest to the MSB. Therefore, it is convenient to define a threshold value s_{th}, according to which we will select appropriate bit positions. For example, if we choose $s_{th} = 0.95$, then we select all positions from the MSB to the first position where $s_i < 0.95$.

3.1.2 Entropy

So far we have selected appropriate bit positions based on their stability s_i. However, in addition to their stability, we have to take into account their uniqueness among different FPGAs. We may assume that if we compare measured values from two equally positioned RO pairs on two FPGAs, bits close to the MSB will not differ at all while the bits approximately in the middle between the most and the least significant bits will vary. It is unnecessary to consider bit positions close to the LSB since it is expected that they will be different due to their instability.

The average entropy of bit position i within each FPGA separately can be determined as follows:

$$H_{intra}(i) = -\frac{1}{m}\sum_{j=1}^{m}\sum_{k=0}^{1} p_j(k)\log_2(p_j(k)), \tag{3.5}$$

where m is the number of FPGAs and $p_j(k)$ is the probability of message k within the j-th FPGA. There are only two possible messages, namely 0 and 1. The probability of their occurrence we compute as:

$$p_j(1) = \frac{1}{n}\sum_{k=1}^{n} \mathrm{maj}(RO_{j,k}, i), \quad p_j(0) = 1 - p_j(1), \tag{3.6}$$

where $RO_{j,k}$ represents the k-th RO pair on the j-th FPGA and n is the number of RO pairs. $\mathrm{maj}(RO, i)$ is the majority of the i-th position determined from k measurements evaluated for each pair of ROs. The result is in this case either 1 or 0 and is defined as:

$$\mathrm{maj}(RO, i) = \mathrm{round}(\frac{1}{k}\sum_{j=1}^{k} b_{j,i}), \tag{3.7}$$

where $\mathrm{round}(x)$ rounds number x to integer.

The average entropy of bit position i of each of the n RO pairs across different FPGAs can be determined using a similar formula:

$$H_{inter}(i) = -\frac{1}{n}\sum_{l=1}^{n}\sum_{k=0}^{1} p_l(k)\log_2(p_l(k)), \tag{3.8}$$

where $p_l(k)$ this time is the probability of message k of the l-th RO pair among different FPGAs. This probability is determined similarly as in the case of H_{intra}, we just calculate it for a particular RO pair among different FPGAs. $p_l(k)$ is defined as:

$$p_l(1) = \frac{1}{m}\sum_{k=1}^{m} \mathrm{maj}(RO_{k,l}, i), \quad p_l(0) = 1 - p_l(1). \tag{3.9}$$

The ideal value of H_{intra} and H_{inter} is 1 (it is the maximum entropy for 1-bit message). Such a value will guarantee that there is no correlation between bits on the same positions among different FPGAs. For example, the lower the entropy H_{inter} is, the higher is the probability of successful estimation of the bit at given position on another FPGA, provided we already know this bit from one FPGA.

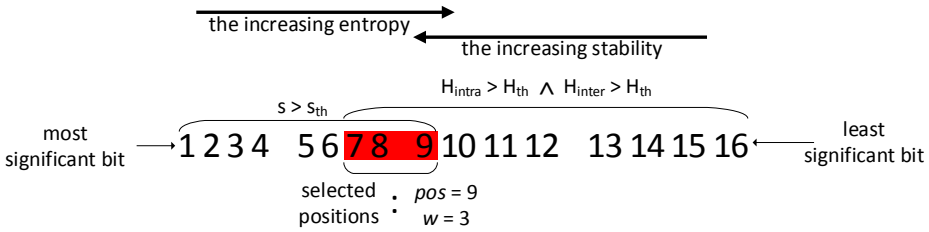


Figure 3.3: The example selection of suitable bit positions for PUF.

3.1.3 Method of selecting suitable bit positions for PUF

When selecting suitable bit positions for PUF, we have to consider both stability and entropy. The stability is increasing towards the MSB, while the entropy is decreasing in the same direction. Therefore, it is necessary to select such a trade-off between good entropy and high stability where both variables are close enough to their ideal value of 1. Based on the statistics, the selection of appropriate bits may be as follows:

- We proceed from the most to the least significant bit as long as the stability s_i is higher then the threshold value s_{th} determined by us. As soon as we come across a position, where $s_i < s_{th}$, we stop and return one position back. This position is denoted as variable *pos*.
- Then we proceed from the position *pos* back towards the MSB, however, this time we consider the entropy values H_{intra} and H_{inter}. We proceed backwards until both entropies satisfy our criteria ($H_{intra}(i) > H_{th} \wedge H_{inter}(i) > H_{th}$, where H_{th} is the chosen threshold value). This procedure is stopped as soon as the entropy is no longer sufficient. The width w of our selection is determined by the difference of the current position and position *pos*.

This whole procedure is shown in Fig. 3.3.

3.1.4 The proposed ROPUF circuit

The maximum amount of bits that we are able to extract using the above described method is $\binom{n}{2} \times w$. It is the number of all possible combinations of RO pairs multiplied by the number of bits that we select from the measured value from one pair.

The circuit used for measurements is shown in Fig. 3.4. There are two sets of ring oscillators and during the measurement of one pair, one RO is selected from each set using multiplexer and a select signal. All of the ROs are enabled and running during the measurement. The outputs of the two selected ROs are fed into the two counters. When one of the counters overflows, the

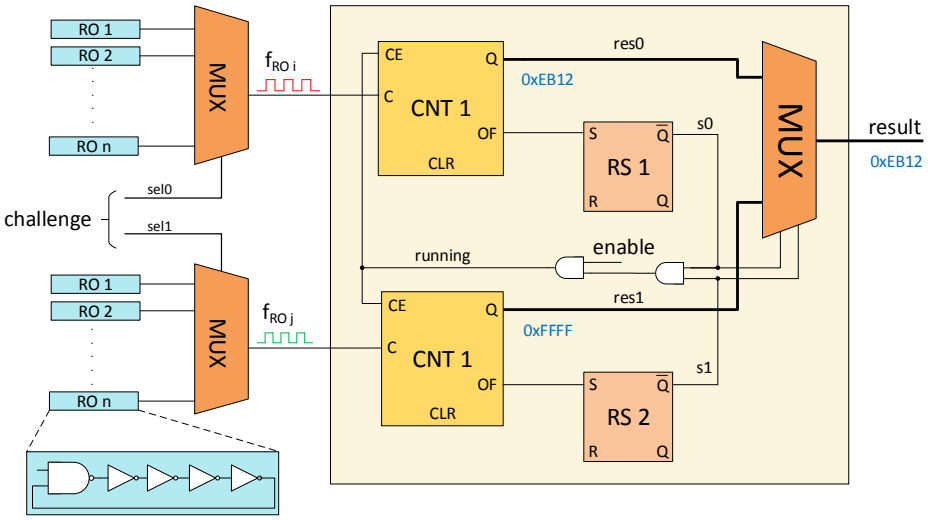


Figure 3.4: The design of the proposed ROPUF.

measurement is stopped and the value of the counter that did not overflow is selected as result. The detection of overflow is realised by two RS Flip-flops as shown in Fig. 3.4.

3.1.5 Performance metrics

We need some metrics in order to evaluate the quality of PUF and its statistical properties. We already discussed how to select good positions of the counter values for the PUF based on their statistical properties such as stability and entropy. After selecting the suitable positions, the PUF outputs made up of these positions need to be evaluated by some additional parameters to validate the selection of positions. In this subsection we describe the evaluation method which we used to determine the qualities of the PUF outputs. First we review two common parameters that are used to evaluate the properties of PUFs, namely *Intra-Hamming distance* (HD_{intra}) and *Inter-Hamming distance* (HD_{inter}). Then we also depict the evaluation of randomness of the PUF outputs.

Intra-Hamming distance

To evaluate the mutual similarity of the PUF outputs, we use Intra-Hamming distance as a metric. HD_{intra} is estimated as:

$$HD_{intra} = \frac{1}{m \times k} \sum_{i=1}^{m} \sum_{j=1}^{k} HD(R_{r_i}, R_{i,j}) \times 100\ [\%], \tag{3.10}$$

where m is the number of FPGAs, R_{r_i} is the reference output of the i-th PUF, which the other outputs are compared to, and k is the number of compared outputs from each PUF. As R_{r_i}, we can use either any output from the given PUF or the mean output of several outputs (this may result in lower HD_{intra}). There are several influences that affect the value of HD_{intra} such as changes in voltage or temperature which cause HD_{intra} to be of higher value.

Inter-Hamming distance

Another important metric that is used to evaluate the PUF quality is the uniqueness of the generated outputs among different FPGAs. We can determine the uniqueness of the generated outputs by calculating the Inter-Hamming distance, which is defined as:

$$HD_{inter} = \frac{1}{\binom{m}{2}} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} HD(R_{r_i}, R_{r_j}) \times 100 \ [\%], \tag{3.11}$$

where m is the number of FPGAs and R_{r_i} is the reference output of the i-th PUF which is the mean output made of all outputs from the given PUF.

Randomness

In addition to HD_{intra} and HD_{inter} we also need to evaluate the randomness of the PUF outputs, since one of the requirements on PUF is that the PUF outputs should be unpredictable. Therefore the PUF outputs should be random (in the context of various challenges and devices, not random for one particular challenge and device) so that a potential adversary will not be able to predict the PUF output based on the knowledge of some other PUF outputs. It is required to have long bit strings in order to evaluate the randomness of them. However, in case of PUF, we are limited by the fact that for very long bit strings we would need a large population of devices.

To evaluate randomness, we used the statistical tests in NIST SP 800-22 [30]. The version of the NIST software used is STS 2.1.2. Most of the tests in this test battery require long input bit sequences; however, some tests can work with sequences that are at least 100 bits long. These tests are the Frequency, Block Frequency, Cumulative Sums, Runs, Longest Run and Approximate Entropy tests. The Block Frequency and Approximate Entropy tests require an additional parameter m, which is the length of the block. As the input sequences for these tests we used the mean PUF outputs from 1000 measurements concatenated to one long bit string. In case of selecting one bit from each RO pair, the bit string would be 3600 bits long (24 FPGAs, 150 RO pairs on each FPGA). The bit string is then split into 10 sequences of the same length that are used as the input sequences for the NIST software. The output of the software are pass rates for each of the tests and the distribution of p-values, which is checked for uniformity [30]. The minimum pass rate of

each of the tests is determined by the NIST software, in this case it is 8/10. The significance level on which we test the distribution of p-values is 0.1, corresponding to 10 input sequences.

3.2 Properties of the proposed PUF design

In Section 3.1 we introduced the reader to our PUF proposal that is based on ring oscillators. We described the main concept of the proposed PUF design which consists of counting the number of oscillations for two selected ROs simultaneously using two counters and processing the value of the counter that did not overflow. In this section we discuss the properties of this PUF design.

3.2.1 Global versus separate selection of the appropriate part of the counter values

The method of processing the obtained counter values was described in previous Section 3.1. In this case, processing means selecting the appropriate part of the counter value, which is used to form the PUF output. In order to determine the appropriate part of the counter value, it is necessary to statistically evaluate each bit position of the counter value from the perspective of stability and entropy. After determining the suitable bit positions, all we have to do to generate the PUF output is to perform one measurement for each RO pair and build the PUF output from the selected parts of the counter values we obtained.

In other words, the proposed method consists of two phases:

1. First, repeated measurements for various RO pairs and various devices have to be performed in order to obtain the counter values. Counter values obtained in this way are statistically evaluated in terms of their stability and entropy. A suitable part of the counter values is determined.

2. When generating a PUF output, the measurement is performed only once for each selected RO pair. The part of the counter value that was determined in the previous phase is extracted from each measured counter value and used to form the PUF output by concatenating it with other extracted values.

So far, we have assumed the statistical evaluation to be performed globally, i.e. for all RO pairs and devices. However, since the ROs in our proposed PUF design can be mutually asymmetric (therefore they may have very different frequencies) and each RO pair can exhibit different behaviour in terms of statistical properties of its bit positions, the suitable bit positions for PUF can also be determined for each RO pair separately.

When determining the suitable bit positions for each RO pair separately, the method remains the same, but the statistical evaluation is performed for each RO pair on each device separately. By determining the suitable bit positions for each RO pair separately, we may achieve higher stability of the PUF outputs or more bits extracted from each counter value. However, it has the disadvantage that the suitable bit positions for each RO pair have to be stored, so that the same part of the counter value is extracted in each measurement. Storing the suitable positions for each RO pair is the main difference from selecting the positions globally for all RO pairs where we have to store only one position (e.g. the start position and the number of bits that are extracted from this position) that is later used for all RO pairs and devices.

3.2.2 Independence on the maximum operating frequencies of the counters

Another property of this design that we observed is the independence of this design on the maximum operating frequencies of the counters. It is possible that due to variations in voltage, temperature or other factors, the frequency of the ROs may exceed the maximum operating frequency of the counter. Even though this phenomenon should be avoided when designing the circuit, it can still happen that due to variation in physical conditions the frequencies of ROs will increase significantly and exceed the operating frequencies of the counters.

When this phenomenon occurs, it results in an incorrect counter value that is read from the counter at the end of the measurement, because the counters miss some clock pulses. However, even in these cases the counter values exhibit correct behaviour and the statistical properties of the PUF outputs remain the same. More about this issue will be shown in Chapter 5.

3.2.3 Partial overflow of the counter value

As mentioned in Section 3.1, the counter values are represented as binary values and by selecting an appropriate part of the counter value we can use a specific part of this counter value to form a PUF output. However, since it is a binary value, the number of bits that are changed when the value is different in repeated measurements can be more than only e.g. one bit. Because the counter values depend on ROs, the measured counter values are not always the same due to the instability of ROs and variations in current physical conditions when the measurement is performed.

As it was explained, the resulting counter value is affected by the frequencies of the ROs that depend on various physical conditions. This is reflected by the stability that is evaluated for each bit position, so the instability of the frequencies of the ROs is considered when selecting an appropriate part of

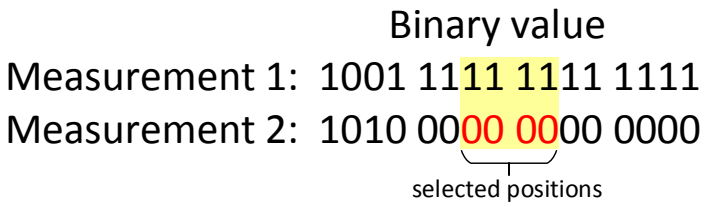


Figure 3.5: Example of a partial overflow of a counter value represented in binary code. Yellow colour highlighted area of the counter value represents the part of the counter value that would be selected for PUF. The bits that change with increasing the counter value by one in the next measurement are highlighted in red.

the counter values for PUF. However, there are still cases in which some RO pairs will produce unstable bits even in stable environmental conditions, because a considerable amount of bits are changed even when the counter value is increased or decreased by one in the next measurement.

Such a case is shown in Fig. 3.5. This figure shows that even though the counter value is increased by one, four bits are changed. When this happens, it shows as a burst of errors in the PUF output and it increases the complexity of correcting the PUF output if it is to be used for cryptographic key generation.

3.2.4 Influence of physical conditions

The basic building element of this PUF design is a RO, hence the physical conditions such as variations in temperature or supply voltage will have impact on the frequencies of ROs and therefore the behaviour of the PUF. In this proposal, one RO pair is connected to two counters in each measurement and the number of oscillations is counted for the two ROs until one of the counters overflows and the measurement is stopped. Therefore, if we knew the exact frequencies of the ROs during measurement, we could derive the resulting counter value as it was shown in Eq. 3.1:

$$Counter\ value = \frac{f_2}{f_1} \times 2^{16},$$

where f_1 is the frequency of the faster RO and f_2 is the frequency of the slower RO.

From Eq. 3.1 it is clear that the resulting counter value is dependent on the ratio of the frequencies of the two selected ROs. It can be expected that when the supply voltage or temperature is changed, then the frequencies of the ROs should be affected in almost the same way.

Ideally, we want the ratio of the frequencies to remain constant in time. In the context of Eq. (3.1) it means that the frequencies of ROs in a pair would

be modified (multiplied) by the same constant k. However, as it will be shown in Chapter 5, the ratio of the frequencies is not constant, but the ratios of the frequencies of the ROs measured in various physical conditions are close each other and in some cases the change in ratio does not affect the bit positions of the counter values that are used for the PUF.

The fact that the ratio of the frequencies of ROs is not constant also implies that the frequencies of the two ROs in a pair are not multiplied by the same constant k but rather two different constants k_1 and k_2. In order to achieve stable PUF outputs, these constants should be almost the same ($k_1 \approx k_2$).

Since the resulting counter value is determined by the ratio of the frequencies of the two ROs in pair (see Eq. (3.1)), this method can be considered as a differential measurement. It will be shown in Chapter 5 that the influence of supply voltage or temperature on the frequencies of ROs is significant. However, since the resulting counter value depends on the ratio of the frequencies, the changes in frequencies will not have such a large impact on the resulting counter value if the change of the frequencies is almost the same.

Chapter 4

Improvements to the proposed PUF

In this chapter we present the modifications of our PUF design that improve its properties in terms of stability and its resistance to the influence of physical conditions. Therefore, this chapter is closely related to Section 3.2 where we discussed the properties of the proposed PUF design.

The first improvement of our PUF design is the application of Gray code on the obtained counter values. This improvement is related to the partial overflow of the counter values described in Section 3.2.3 and will be presented in Section 4.1.

The second improvement is related to the issue described in Section 3.2.4, i.e. the influence of various physical conditions on the behaviour of the PUF. Since the PUF design is based on ROs, the physical conditions will have a significant impact on the frequencies of ROs, however, our goal is to make the differential measurement (frequency ratio) more robust against such effects. In Section 4.2 we present a possible solution to this problem and in Chapter 5 we will show the results of the performed experiments.

4.1 Gray code

One of the issues of selecting a block of bits from each counter value is that all of the selected bits may change in the next measurement since they are represented in binary code and they are a part of a counter value. So even if the final value is increased or decreased just by one, all of the bits can be influenced by this change. The first step to solving this issue involves the application of Gray code to the obtained counter values. The reason for using Gray code is the fact that two successive values differ in only one bit, so this can eliminate the partial overflow and increase the overall stability of the selected bits and even increase the number of extractable bits from each

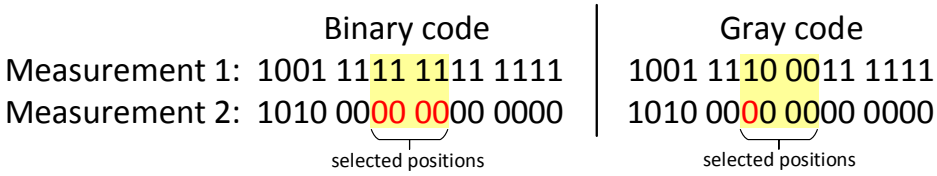


Figure 4.1: Example of counter value overflow and its behaviour when the counter value is represented in binary and Gray code (Gray code is applied only on the selected part of the counter value in this example). Areas of the counter value highlighted with yellow colour represent the part of the counter value that would be selected for PUF and the bits that change with increasing the counter value in the next measurement are highlighted in red.

counter value. An example is shown in Fig. 4.1. The Gray code is used in the following form:

$$\begin{aligned} g_1 &= b_1 \\ g_i &= b_i \oplus b_{i-1}, \end{aligned} \tag{4.1}$$

where g_i is the i-th bit of the value represented in Gray code and b_i is the i-th bit of the value in binary code. b_1 and g_1 are MSBs of the value encoded into Gray code.

There are two possible ways to apply Gray code on the counter values. Gray code can be applied either to the whole counter value or only to the selected part. Our observations indicate that when Gray code is applied to the whole counter value, the stability of the PUF outputs is higher and the uniqueness of the PUF outputs is lower than when encoding only the selected part of the counter value. Therefore the choice must be determined by the actual situation and our preference.

The difference of statistical properties is caused by the definition of Gray code (4.1). This definition implies that each bit in the encoded value depends only on the current and the preceding bit and is not influenced by any bits in the direction to LSB. Therefore the statistical properties of the PUF will remain the same if we apply Gray code either on the selected part of the counter value (e.g. positions 7–8) or on the whole part of the counter value from the first selected bit position to LSB (e.g. 7–16 in case of 16-bit counter value). However, different statistical properties may be achieved, when Gray code is applied from some of the bits closer to MSB or to the whole counter value. This is shown in Fig. 4.2 on 4-bit values, where the behaviour of the last three bits is observed and the values are compared to the first one. When the value is increased by 1 and 2, then in case of Gray code is applied only to the last three bits, the difference is in one and two bits. But when Gray code is applied on the whole value, then there is no difference (value increased by

	Binary	Gray (part)	Gray (whole)
	0111	0100	0100
(+1)	1000	1000	1100
(+2)	1001	1001	1101

Figure 4.2: Comparison of binary and Gray code. Yellow area corresponds to bits that are encoded in Gray code and the differences to the original value are highlighted with red colour. Only the last three bits are observed in this example (the first bit is neglected).

1) or only in one bit (value increased by 2). The described behaviour implies that the PUF will be more stable when Gray code is applied on the whole counter value.

As mentioned before, the method of application of Gray code to the counter value doesn't influence only the stability but also the uniqueness of the PUF outputs. The uniqueness of the PUF outputs is lower when Gray code is applied on the whole counter value for the same reason why the stability is higher (see Fig. 4.2). The counter values for the same RO pair on different FPGAs are close to each other, but they also differ enough so that we can distinguish them. However, by applying the Gray code on the whole counter values, the difference between these values may not affect the selected positions that are used for the PUF outputs if the difference between the counter values is not large enough.

4.2 Placement of ROs

In our PUF proposal, we stated that compared to classical approach [32], our PUF design does not require the ROs to be mutually symmetric. However, it will be shown in Chapter 5 that when using symmetrical ROs, the proposed PUF exhibits better behaviour in terms of stability at varying voltage or temperature.

In Section 3.2.4 we presented how the resulting counter value is determined (ratio of the frequencies of the two ROs in pair) and the physical conditions that may affect the frequencies of ROs. In order to achieve high stability of the PUF outputs at varying physical conditions, the ratios of the frequencies of the ROs need to remain the same, or at least as close as possible.

It can be expected that the frequencies of ROs will change in a similar way when the ROs are mutually symmetric. Therefore, the ratios of the

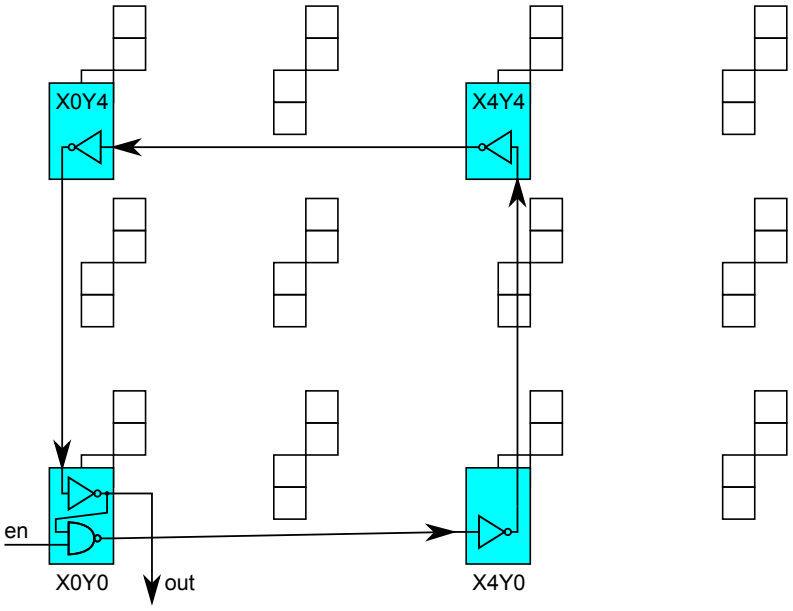


Figure 4.3: Relative mutual placement of logic gates used for each RO on Xilinx Spartan3E-100 CP132.

frequencies of RO pairs should be almost constant. In our experiments we placed the logic gates of each RO so that the ROs placed on the FPGA are mutually symmetric. This is a different approach than the one we used so far because the placement of ROs was originally left to the automatic placer of ISE 14.7.

The RO gates were placed using RLOC constraints that were the same for all ROs. We used 5-stage ring oscillators (see Fig. 2.10) that were placed into four slices, with each inverter occupying one slice and the last one sharing the slice with the NAND gate. The relative locations of the slices were (X0, Y0), (X4, Y0), (X4, Y4) and (X0, Y4) as shown in Fig. 4.3. Other placements were left unconstrained and the choice was left to the automatic placer of ISE 14.7. It should be also noted that the constraints were put only on the relative placement of the gates of each RO but not on the interconnects between them. Therefore, even though the placement of ROs is mutually symmetric, the ROs themselves are not absolutely symmetrical since the interconnects between the gates of each RO can be different.

Using symmetric ROs we were able to achieve higher stability of the PUF outputs at varying voltage or temperature. The statistical properties of the PUF itself remained similar. All of this will be shown in the following Chapter 5.

Chapter 5

Experimental results and analysis

This chapter presents the results of our experiments. The experiments were performed mainly on Digilent Basys 2 FPGA boards (Xilinx Spartan3E-100 CP132) [6]. To verify that the proposed PUF design works properly on other types of FPGAs, we also performed measurements on Digilent Nexys 3 FPGA boards (Xilinx Spartan-6), which is manufactured with 45nm technology while Spartan-3E is a 90nm technology. As a measurement circuit we used the design shown in Fig. 3.4 with 300 ROs divided into two sets of 150 ROs each.

The designs of logical circuits and their bitstreams for FPGAs were created using Xilinx ISE 14.7 [36]. The FPGAs were then programmed with the resulting bitstreams with Digilent Adept 2 [4]. The communication between PC and FPGA was realised through USB using DEPP interface (Digilent Adept Asynchronous Parallel Port Interface). It is a library that is part of Adept SDK [5] which among other things, enables data transfer between PC and the target device. Adept SDK provides API in C language that can be used for interaction with various FPGA boards.

5.1 Selection of suitable positions

In this section we present the results achieved in our previous work [13]. The measurements were performed on 24 Digilent Basys 2 FPGA boards and as mentioned before, the circuit we used contained 300 ROs divided into two groups of 150 ROs each (see Fig. 3.4). The ROs are ordinary 5-stage ROs as shown in Fig. 2.10 and their oscillations are counted by 16-bit counters.

We performed the measurements for 150 RO pairs (each oscillator from the first group is paired with another unused oscillator from the second group) and 1000 measurements were executed for each pair. The next measurement was performed for 450 pairs with 500 measurements for each pair.

150 pairs of ring oscillators					450 pairs of ring oscillators				
pos(i)	s_i	H_{intra}	H_{inter}	$P(b_i=1)$	pos(i)	s_i	H_{intra}	H_{inter}	$P(b_i=1)$
1	1	0.1414	0	0.98	1	0.9999	0.5473	0.0132	0.8736
2	0.9996	0.9477	0.0557	0.3663	2	0.9997	0.9831	0.0395	0.4239
3	0.9995	0.9944	0.0794	0.5430	3	0.9995	0.9949	0.0937	0.5410
4	0.9985	0.9962	0.1657	0.5336	4	0.9989	0.9979	0.1898	0.5248
5	0.9983	0.9992	0.2955	0.4998	5	0.9980	0.9982	0.3598	0.5233
6	0.9950	0.9941	0.7183	0.4765	6	0.9961	0.9973	0.6585	0.5233
7	**0.9908**	**0.9958**	**0.9475**	**0.5056**	**7**	**0.9920**	**0.9982**	**0.9232**	**0.5118**
8	**0.9815**	**0.9954**	**0.9663**	**0.4959**	**8**	**0.9841**	**0.9986**	**0.9682**	**0.4978**
9	0.9650	0.9946	0.9639	0.4931	9	0.9677	0.9984	0.9692	0.4971
10	0.9297	0.9954	0.9681	0.4997	10	0.9347	0.9983	0.9706	0.5047
11	0.8624	0.9943	0.9701	0.4960	11	0.8709	0.9984	0.9707	0.5042
12	0.7268	0.9957	0.9728	0.4985	12	0.7441	0.9981	0.9709	0.4969
13	0.5569	0.9932	0.9732	0.5001	13	0.5691	0.9987	0.9706	0.5005
14	0.5139	0.9961	0.9654	0.4985	14	0.5185	0.9982	0.9727	0.4996
15	0.5135	0.9957	0.9647	0.4975	15	0.5180	0.9981	0.9663	0.4991
16	0.5140	0.9897	0.9613	0.4967	16	0.5182	0.9972	0.9687	0.4977

Table 5.1: Statistical evaluation of 16-bit counter values obtained from 24 Digilent Basys 2 FPGA boards.

First, we want to present the results of stability (s_i), entropy (H_{intra}, H_{inter}) and bias ($P(b_i = 1)$) for each position of the 16-bit counter values. See Section 3.1 for the description of these parameters. As can be seen in Table 5.1, the entropy rises and is approaching the ideal value of 1 (especially H_{inter}, H_{intra} is high since position 2) while the stability is decreasing with the increasing position. The bias is very close to the ideal value of 0.5 on all positions apart from the first few. The positions are numbered from the most significant bit (MSB), i.e. position 1 corresponds to the MSB and position 16 corresponds to the least significant bit (LSB).

The next Table 5.2 presents the results for PUF that uses different selections of positions. We used the same data that we measured before and assembled PUF responses from them. PUF responses created in this fashion were $150 \times w$ or $450 \times w$ bits long, where w is the number of bits selected from each RO pair. When calculating HD_{intra}, we used the first response as the reference response. We calculated HD_{inter} among all FPGAs using mean responses from all obtained responses for each FPGA. Since one of our goals was to achieve HD_{inter} close to 50%, it was desirable to select bits starting at position 7. However, since we need the PUF responses to be stable enough, we have to select bits so that HD_{intra} is very small, ideally 0%. Bit positions that fulfil both these requirements are for example positions 7–8.

150 pairs of ring oscillators				
positions	6–8	**7–8**	7–9	8–9
w [-]	3	**2**	3	2
HD_{intra} [%]	1.61	**2.05**	3.10	3.95
HD_{intra} interval [%]	$\langle 0.00, 6.44\rangle$	$\langle$**0.00, 7.00**$\rangle$	$\langle 0.00, 8.67\rangle$	$\langle 0.00, 10.00\rangle$
HD_{inter} [%]	44.27	**49.15**	49.31	49.70
HD_{inter} interval [%]	$\langle 36.67, 52.44\rangle$	$\langle$**39.67, 58.33**$\rangle$	$\langle 42.22, 56.67\rangle$	$\langle 40.33, 58.33\rangle$
450 pairs of ring oscillators				
positions	6–8	**7–8**	7–9	8–9
w [-]	3	**2**	3	2
HD_{intra} [%]	1.37	**1.78**	2.79	3.60
HD_{intra} interval [%]	$\langle 0.00, 3.56\rangle$	$\langle$**0.00, 4.56**$\rangle$	$\langle 0.74, 6.37\rangle$	$\langle 1.11, 8.22\rangle$
HD_{inter} [%]	42.69	**48.42**	48.94	49.96
HD_{inter} interval [%]	$\langle 34.67, 52.30\rangle$	$\langle$**42.33, 56.11**$\rangle$	$\langle 44.74, 54.74\rangle$	$\langle 45.44, 54.67\rangle$

Table 5.2: The results of statistical tests performed on the PUF outputs composed of various bit selections and a comparison of the PUF quality. The measurements were performed on 24 Digilent Basys 2 FPGA boards (Xilinx Spartan-3E)

5.2 Timing analysis

For further investigation of the behaviour of ROs we performed measurements using an oscilloscope. We sampled the signal coming out of ROs in the measured pairs of ROs with the oscilloscope and then processed the obtained waveforms in software. To determine the impact of the delay of the circuit that detects the overflow and stops the counting on the resulting counter value, we performed measurements in which the chosen RO pairs were enabled and from this moment these ROs were sampled. We read the counter values from the FPGA and processed the waveforms obtained from the oscilloscope. By counting the rising edges in the waveforms for the two ROs in a pair we determined the correct value that should be in the counter.

Due to variation in voltage or other reasons, the frequency of a selected RO may exceed the maximum operating frequency of the respective counter. This should be avoided by design, but if it happens, the counter starts missing some clock pulses and it results in incorrect counter value read from the FPGA. We have measured several instances where the reported counter values differ from the exact values calculated by counting clock edges measured with the oscilloscope. However, even in these instances the counter values are consistent in time and the statistics for the PUF outputs remains the same. Table 5.3 shows the mean and standard deviations of differences of counter values from the correct ones for five RO pairs where at least one RO frequency exceeded the maximum operating frequency of the counters. The standard deviations of

RO pair	1	2	3	4	5
Mean difference	2812.43	3502.78	3085.41	4037.65	2861.42
Standard Deviation	0.5366	0.9054	0.7797	0.9987	0.7808

Table 5.3: The difference of measured counter values and the correct ones for five RO pairs.

differences are very small, indicating that the counter values remain consistent when the measurements are repeated. The measurements were carried out 100 times for each pair of ROs.

When using proper (fast enough) counters, the difference of counter values and the correct ones should ideally be 0. The difference we measured was 0 or 1. When the difference is 1, it is caused by the overflow detection logic of one of the counters – before the other counter is stopped, it manages to perform one additional count. This way we verified that the stopping circuit works correctly and consistently.

5.3 Gray code

Table 5.4 shows results for the PUF outputs that were extracted from 150 and 450 pairs of ROs and the Gray code was applied to the selected part of the counter values. The measurements were repeated 1000 times in case of 150 pairs of ROs and 500 times in case of 450 pairs. The experiment was realised on 24 Digilent Basys 2 FPGA boards. The results for the PUF outputs with applied Gray code are compared to the outputs without Gray code. The PUF outputs are composed of different positions of the counter values. The value w is the number of bits selected from each counter value. The next parameters are HD_{intra} and HD_{inter}.

From the observation of HD_{intra} in Table 5.4, it can be seen that the PUF outputs are more stable when Gray code is used. Without Gray code, we determined positions 7–8 as good positions for PUF. But with Gray code, we were able to extract more bits from each counter value with almost the same values of HD_{intra} and HD_{inter}. For example, when we select positions 7–10, the average HD_{intra} is 2.71% and the average HD_{inter} is 49.32%. So now we can extract twice as many bits as before with almost the same value of HD_{intra}.

Table 5.5 presents the results for Gray code applied both to the selected part of the counter values and the whole counter values. These results confirm that when Gray code is used on the whole counter values, the PUF outputs are more stable (lower HD_{intra}) but uniqueness is negatively affected (lower HD_{inter}).

Without Gray code

150 pairs of ring oscillators

positions	6–8	**7–8**	7–9	8–9
w [-]	3	**2**	3	2
HD_{intra} [%]	1.61	**2.05**	3.10	3.95
HD_{intra} interval [%]	$\langle 0.00, 6.44\rangle$	$\langle$**0.00, 7.00**$\rangle$	$\langle 0.00, 8.67\rangle$	$\langle 0.00, 10.00\rangle$
HD_{inter} [%]	44.27	**49.15**	49.31	49.70
HD_{inter} interval [%]	$\langle 36.67, 52.44\rangle$	$\langle$**39.67, 58.33**$\rangle$	$\langle 42.22, 56.67\rangle$	$\langle 40.33, 58.33\rangle$

450 pairs of ring oscillators

positions	6–8	**7–8**	7–9	8–9
w [-]	3	**2**	3	2
HD_{intra} [%]	1.37	**1.78**	2.79	3.60
HD_{intra} interval [%]	$\langle 0.00, 3.56\rangle$	$\langle$**0.00, 4.56**$\rangle$	$\langle 0.74, 6.37\rangle$	$\langle 1.11, 8.22\rangle$
HD_{inter} [%]	42.69	**48.42**	48.94	49.96
HD_{inter} interval [%]	$\langle 34.67, 52.30\rangle$	$\langle$**42.33, 56.11**$\rangle$	$\langle 44.74, 54.74\rangle$	$\langle 45.44, 54.67\rangle$

With Gray code

150 pairs of ring oscillators

positions	7–8	7–9	**7–10**	8–9
w [-]	2	3	**4**	2
HD_{intra} [%]	1.37	1.77	**2.71**	2.63
HD_{intra} interval [%]	$\langle 0.00, 4.33\rangle$	$\langle 0.00, 4.44\rangle$	$\langle$**0.50, 5.50**$\rangle$	$\langle 0.00, 6.33\rangle$
HD_{inter} [%]	48.49	49.06	**49.32**	50.00
HD_{inter} interval [%]	$\langle 40.00, 56.99\rangle$	$\langle 42.67, 55.56\rangle$	$\langle$**44.00, 55.00**$\rangle$	$\langle 39.00, 58.33\rangle$

450 pairs of ring oscillators

positions	7–8	7–9	**7–10**	8–9
w [-]	2	3	**4**	2
HD_{intra} [%]	1.19	1.60	**2.45**	2.40
HD_{intra} interval [%]	$\langle 0.00, 2.78\rangle$	$\langle 0.52, 3.70\rangle$	$\langle$**0.78, 5.33**$\rangle$	$\langle 0.78, 5.56\rangle$
HD_{inter} [%]	47.44	48.30	**48.74**	49.97
HD_{inter} interval [%]	$\langle 35.89, 60.33\rangle$	$\langle 39.48, 57.11\rangle$	$\langle$**41.61, 56.39**$\rangle$	$\langle 45.67, 56.11\rangle$

Table 5.4: The results of statistical tests performed on PUF outputs composed of various bit selections, and a comparison of the PUF quality with and without Gray code. The measurements were performed on 24 Digilent Basys 2 FPGA boards (Xilinx Spartan-3E)

5.4 Evaluation of the proposed PUF on Nexys 3 FPGA boards

To verify that the proposed PUF design works properly on other types of FPGAs, we performed measurements on six Digilent Nexys 3 FPGA boards that contain Xilinx Spartan-6 which is manufactured with 45nm technology (Spartan-3E is 90nm technology). Table 5.6 shows the results of these mea-

	Gray (part)	Gray (whole)
HD_{intra} [%]	1.37	1.03
HD_{intra} interval [%]	$\langle 0.00, 4.33 \rangle$	$\langle 0.00, 3.67 \rangle$
HD_{inter} [%]	48.49	40.70
HD_{inter} interval [%]	$\langle 40.00, 56.99 \rangle$	$\langle 30.67, 50.67 \rangle$

Table 5.5: The difference of the PUF outputs with Gray code applied to the selected part of the counter values and to the whole counter values for 150 RO pairs and for selected positions 7–8.

positions	6–7	7–8	7–9	7–10
w [-]	2	2	3	4
HD_{intra} [%]	0.81	1.56	1.98	2.88
HD_{intra} interval [%]	$\langle 0.00, 2.00 \rangle$	$\langle 0.33, 3.00 \rangle$	$\langle 0.74,\ 3.33 \rangle$	$\langle 1.28, 4.00 \rangle$
HD_{inter} [%]	40.27	49.36	49.30	49.54
HD_{inter} interval [%]	$\langle 28.67, 55.67 \rangle$	$\langle 45.56, 51.89 \rangle$	$\langle 47.33,\ 51.26 \rangle$	$\langle 47.67, 51.50 \rangle$

Table 5.6: The results of statistical tests performed on PUF outputs composed of various bit selections on Digilent Nexys 3 FPGA boards (Xilinx Spartan-6).

surements for 450 RO pairs and 500 measurements for each pair with Gray code applied to the selected part of the counter values. The results are similar to those obtained from Basys 2 (Spartan-3E).

5.5 Evaluation of randomness

As mentioned in Section 3.1, long bit strings are required to evaluate randomness. Since we had limited possibilities, we applied particular tests from the NIST STS that are suitable for evaluating short input sequences. We evaluated the randomness of 10 sequences that were 360 bits long (or its multiple, depending on the number of bits selected from each counter value) and the minimum pass rate for these tests was 8/10. But for the test of the distribution of p-values, at least 55 input sequences are required in order to obtain statistically meaningful results according to [30]. Therefore, the results from these tests have to be taken with caution.

Table 5.7 contains the results of the applied tests. Empty cells mean that the pass rate for that test is 10/10 and red colored cells indicate that the test failed for the distribution of p-values. The upper part of this table shows the results for PUF outputs made of each position of the counter values. It can be seen that for positions 1 to 5 the tests indicate that the input sequences made from these positions are not random. This result is expected since these positions are close to the MSB and therefore they have low entropy and the

PUF outputs made from these bits would have low HD_{inter}. The positions in the direction to the LSB have better results, but still with some failures in the test of the distribution of p-values. As mentioned before, this can be caused due to the small number of input sequences. The bottom part of Table 5.7 presents the results for more extracted bits from the counter values for various selections of positions. It is divided into two parts, where the left side contains results for the parts of the counter values in binary code, while the other side contains results for the parts of the counter values encoded with Gray code. It can be seen that Gray code does not have a negative impact on the randomness of the PUF outputs.

positions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Frequency	0/10	0/10													9/10	9/10
Block Frequency (m=2)	0/10															
Block Frequency (m=3)	0/10				9/10							9/10				
Block Frequency (m=4)	0/10				8/10											
Cumulative Sums	0/10	0/10													9/10	9/10
Runs	0/10	0/10							9/10			9/10	9/10			
Longest Run	0/10	0/10							9/10							
Approximate Entropy (m=2)	0/10	0/10		1/10												9/10
Approximate Entropy (m=3)		0/10		3/10	8/10	9/10										9/10
Approximate Entropy (m=4)		0/10	9/10	2/10								8/10				

	Binary code					Gray code					
positions	6–7	7–8	8–9	6–8	7–9	6–7	7–8	8–9	6–8	7–9	7–10
Frequency						9/10					9/10
Block Frequency (m=2)											
Block Frequency (m=3)											
Block Frequency (m=4)											
Cumulative Sums						9/10					
Runs	9/10			8/10							
Longest Run											
Approximate Entropy (m=2)	9/10			8/10							
Approximate Entropy (m=3)	9/10			8/10							
Approximate Entropy (m=4)						9/10					

Table 5.7: The results of the tests from NIST STS. Each cell contains pass rate and indicates whether the test failed for the distribution of p-values (red cells). Empty cells mean that the pass rate for those cells was 10/10. Minimum allowed pass rate was 8/10. The top table contains results for PUF outputs made from each position of the counter values. The bottom table presents results for various selections of positions of the counter values in binary and Gray code.

50 pairs of ring oscillators

pos(i)	s_i	H_{intra}	H_{inter}	$P(b_i=1)$
1	1.0000	0.0000	0.0000	1.0000
2	1.0000	0.0000	0.0000	1.0000
3	0.9995	0.6911	0.0726	0.8143
4	0.9985	0.8667	0.1347	0.7107
5	0.9975	0.9953	0.2652	0.5197
6	0.9913	0.9926	0.6779	0.5364
7	0.9842	0.9610	0.8659	0.6016
8	0.9729	0.9865	0.9067	0.5162
9	0.9434	0.9759	0.9313	0.5524
10	0.8801	0.9940	0.9093	0.5054
11	0.7869	0.9926	0.9338	0.4918
12	0.6292	0.9833	0.9297	0.5078
13	0.5210	0.9912	0.9022	0.5023
14	0.5136	0.9922	0.9133	0.5007
15	0.5130	0.9787	0.9296	0.4987
16	0.5134	0.9789	0.9001	0.4977

150 pairs of ring oscillators

pos(i)	s_i	H_{intra}	H_{inter}	$P(b_i=1)$
1	1.0000	0.0000	0.0000	1.0000
2	1.0000	0.0000	0.0000	1.0000
3	0.9992	0.6527	0.0730	0.8319
4	0.9987	0.8623	0.1357	0.7143
5	0.9971	0.9746	0.3320	0.5897
6	0.9936	0.9850	0.7524	0.5655
7	0.9866	0.9890	0.9413	0.5376
8	0.9710	0.9951	0.9648	0.5213
9	0.9415	0.9950	0.9718	0.5111
10	0.8895	0.9951	0.9657	0.5032
11	0.7839	0.9941	0.9662	0.5206
12	0.6214	0.9956	0.9655	0.4997
13	0.5186	0.9936	0.9706	0.5005
14	0.5131	0.9946	0.9679	0.5001
15	0.5129	0.9941	0.9661	0.4993
16	0.5130	0.9840	0.9588	0.4971

Table 5.8: Statistical evaluation of 16-bit counter values for symmetric ROs. The first table contains statistical evaluation for 50 RO pairs measured 1000 times on 10 Digilent Basys 2 FPGA boards (Xilinx Spartan-3E). In the second table, there are results of statistical evaluation for 150 RO pairs measured 1000 times on 24 of the same FPGA boards.

5.6 Evaluation of the proposed PUF with symmetric ROs

As described in Section 4.2, we placed the gates of each ROs as shown in Fig. 4.3. We used the same RLOC constraints for the placement of all 5-staged ROs and put each RO into four slices with each inverter occupying one slice and the last one sharing it with the NAND gate. The placement of the ROs themselves and also the interconnects between the gates were left unconstrained and the choice was left to the automatic placer of ISE 14.7. The circuit used for measurements remained the same (see Fig. 3.4), but the number of ROs was smaller due to the RLOC constraints that were used. There were 100 ROs divided into two groups of 50 ROs each.

The left part of Table 5.8 shows the results of the statistical evaluation of each bit position of the counter values for 50 RO pairs measured 1000 times on 10 Digilent Basys 2 FPGA boards. It can be seen that the results are very similar to what was presented in Table 5.1. Only the values of H_{inter} are a little bit lower. It is caused by the smaller number of boards which were used for the measurements.

The right part of Table 5.8 contains the statistical evaluation of each bit

Gray code (part)

50 pairs of ring oscillators

positions	7–8	7–9	7–10	8–9
w [-]	2	3	4	2
HD_{intra} [%]	1.96	2.89	4.60	4.30
HD_{intra} interval [%]	$\langle 0.00, 7.00\rangle$	$\langle 0.00, 8.00\rangle$	$\langle 0.00, 9.50\rangle$	$\langle 0.00, 12.00\rangle$
HD_{inter} [%]	48.51	49.53	49.93	50.18
HD_{inter} interval [%]	$\langle 38.00, 60.00\rangle$	$\langle 40.67, 60.00\rangle$	$\langle 41.50, 60.00\rangle$	$\langle 40.00, 63.00\rangle$

150 pairs of ring oscillators

positions	7–8	7–9	7–10	8–9
w [-]	2	3	4	2
HD_{intra} [%]	2.16	2.81	4.21	4.21
HD_{intra} interval [%]	$\langle 0.00, 5.67\rangle$	$\langle 0.22, 6.00\rangle$	$\langle 0.83, 8.33\rangle$	$\langle 0.33, 9.00\rangle$
HD_{inter} [%]	48.81	49.23	49.45	49.88
HD_{inter} interval [%]	$\langle 40.00, 60.33\rangle$	$\langle 42.67, 56.22\rangle$	$\langle 43.67, 55.17\rangle$	$\langle 42.33, 56.67\rangle$

Gray code (whole)

50 pairs of ring oscillators

positions	7–8	7–9	7–10	8–9
w [-]	2	3	4	2
HD_{intra} [%]	1.35	2.48	4.30	3.26
HD_{intra} interval [%]	$\langle 0.00, 6.00\rangle$	$\langle 0.00, 7.33\rangle$	$\langle 0.00, 9.50\rangle$	$\langle 0.00, 11.00\rangle$
HD_{inter} [%]	47.00	48.52	49.18	51.04
HD_{inter} interval [%]	$\langle 34.00, 60.00\rangle$	$\langle 39.33, 57.33\rangle$	$\langle 41.00, 56.50\rangle$	$\langle 38.00, 65.00\rangle$

150 pairs of ring oscillators

positions	7–8	7–9	7–10	8–9
w [-]	2	3	4	2
HD_{intra} [%]	1.72	2.52	3.99	3.22
HD_{intra} interval [%]	$\langle 0.00, 5.00\rangle$	$\langle 0.00, 5.56\rangle$	$\langle 0.67, 7.83\rangle$	$\langle 0.00, 7.67\rangle$
HD_{inter} [%]	46.28	47.54	48.19	49.78
HD_{inter} interval [%]	$\langle 34.33, 55.33\rangle$	$\langle 38.44, 57.11\rangle$	$\langle 41.33, 56.33\rangle$	$\langle 41.33, 58.00\rangle$

Table 5.9: The results of statistics carried out for PUF outputs composed of various bit selections and a comparison of the PUF quality with Gray code applied on the whole counter value or only on the selected part. The measurements were performed on Digilent Basys 2 FPGA boards (Xilinx Spartan-3E)

position of the counter values for 150 RO pairs measured 1000 times on 24 Digilent Basys 2 FPGA boards. As we can see, the results are also very similar to Table 5.1 and also the suitable bit positions for PUF are the same (the positions starting from position 7).

From the same set of measurements we statistically evaluated the PUF outputs composed of various selections of bit positions. Table 5.9 contains the results of this evaluation. We provide this evaluation with Gray code applied

either to the selected part of the counter values or to the whole counter values. We can see a similar behaviour of the metrics HD_{intra} and HD_{inter} to that in Table 5.5 which was described in Section 4.1. When Gray code is applied to the whole counter values, the HD_{intra} is lower, but HD_{inter} is lower as well. However, compared to results presented in Table 5.5 for positions 7–8, the difference of these two cases of Gray code usage is not so significant in case of symmetric ROs. The difference in HD_{inter} for asymmetric ROs is 7.79% (from 48.49% to 40.70%) while for symmetric ROs the difference is about 2% (48.51% to 47.00% and 48.81% to 46.28%).

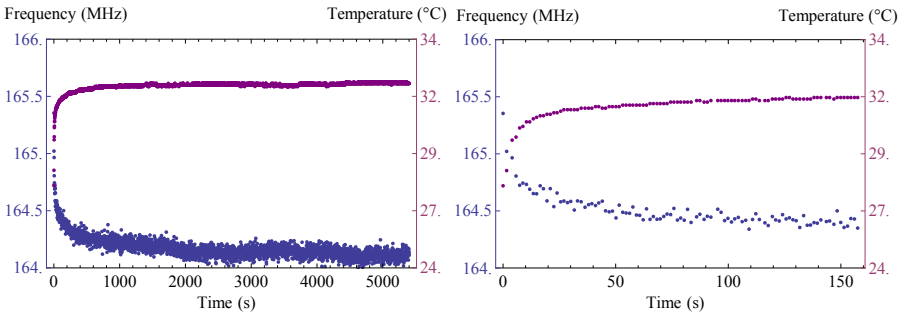
5.7 Influence of supply voltage

All of the previous measurements were performed at normal environmental conditions with stable temperature and supply voltage. In this section, we present the results of the experiments performed at varying voltage. First, we will describe the influence of temperature conditions on the measurements to evaluate the correctness of our approach. Then we will present the results of the experiments and show the difference of the results obtained for both asymmetric and symmetric ROs.

5.7.1 Temperature conditions during the measurements

All of the measurements regarding the influence of voltage were performed at stable environmental conditions with stable room temperature $24.5 \pm 1\,°C$. Before the measurements were started, the FPGAs were turned on with running oscillators for at least 2 minutes to eliminate other influences than the influence of change of voltage. The reason for this is the fact that in the short time period after enabling the ring oscillators, there is a significant change in temperature of the FPGA which could affect the measurement.

To evaluate the correctness of our approach, we observed the behaviour of ring oscillators on three different FPGAs during warm-up. The duration of the measurement was at least 30 minutes and up to 90 minutes. For this measurement we selected one RO pair, which is common for all three FPGAs, and then two other random RO pairs for each FPGA (three RO pairs for each FPGA in total). The values of temperature, RO frequency and counter value were sampled in intervals of 1.5 to 3 seconds. We analysed the results and the maximum difference of the gathered counter values in the whole time interval was such that it influences positions 9–16 of the counter value. For the time after 2 minutes (the waiting time that we used for the measurements of the influence of the supply voltage), the maximum difference affects positions 10–16, and when extending the delay to 30 minutes, it still affects positions 10–16 of the counter value. These results indicate that the influence of change in temperature during the warm-up of FPGAs with stable room temperature is negligible compared to the influence of voltage.



(a) The frequency progress for approximately 90 minutes.

(b) A more detailed frequency progress at the beginning of warm-up of FPGA.

Figure 5.1: Frequency behaviour during warm-up of FPGA (Xilinx Spartan-3E). Blue dots represent the measured frequency of RO. The temperature at the time of the frequency measurement is marked with purple colour.

Fig. 5.1 shows the frequency behaviour during warm-up of FPGA for one RO on one FPGA. The graphs contains both the frequency of RO and the temperature that was recorded at the time of measurement. Fig. 5.1(a) shows the behaviour of one RO on one FPGA for the whole recorded time. It can be seen that the largest change in both temperature and frequency occurs at the beginning of the measurement, which is shown in better detail in Fig. 5.1(b).

5.7.2 Asymmetric ROs

In previous subsection we described the temperature conditions during the measurements performed in this experiment. The next measurement concerns the influence of voltage on the behaviour of the proposed ROPUF design. The measurements were performed on two Digilent Basys 2 FPGA boards containing Xilinx Spartan3E-100 CP132. The main power supply for the FPGA's internal logic is V_{ccint} and its nominal voltage is 1.2V. The maximum ratings for V_{ccint} are -0.5V and 1.32V, with manufacturer's recommended range from 1.14V to 1.26V. The circuit remained the same and the results presented in Table 5.10 relate to 1000 measurements for 150 RO pairs; they show how the PUF outputs are different from those obtained at nominal voltage, which is 1.2V. The range of tested voltages is from 1.018V to 1.286V and the selected positions of counter values for the PUF outputs are 7–8 and 7–10.

For the purpose of obtaining values of HD_{intra}, we averaged the PUF outputs at each voltage. The average PUF output can be determined as the majority of each column when the PUF outputs are written in rows.

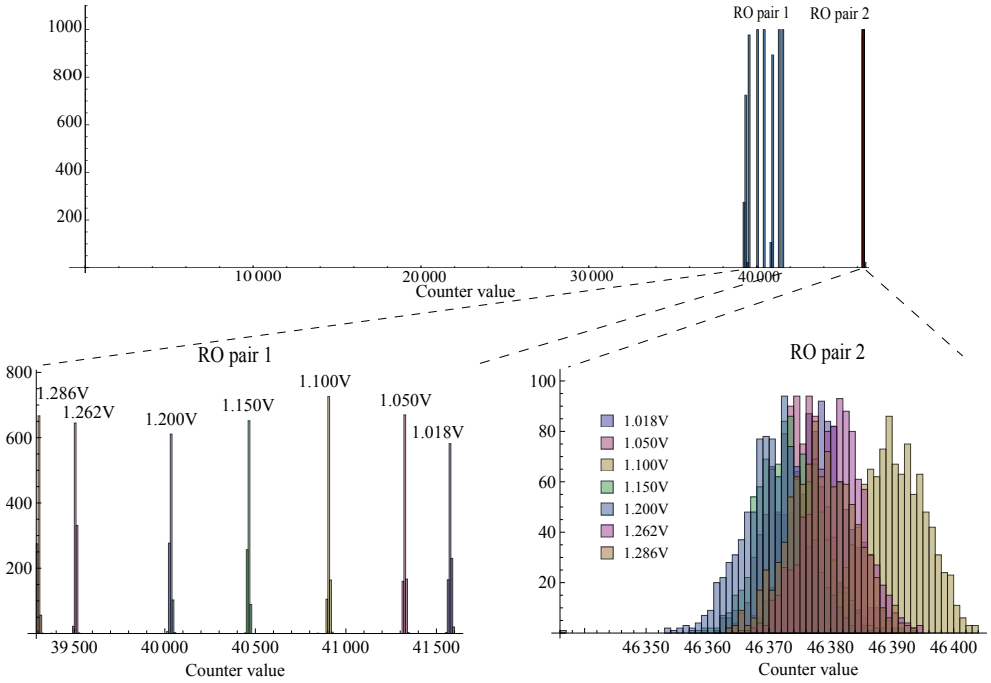
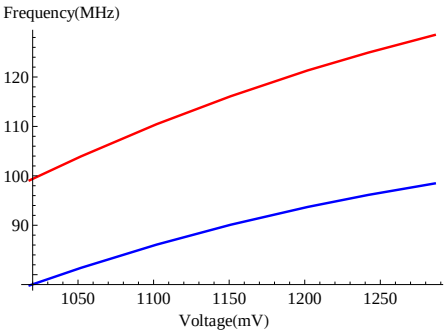


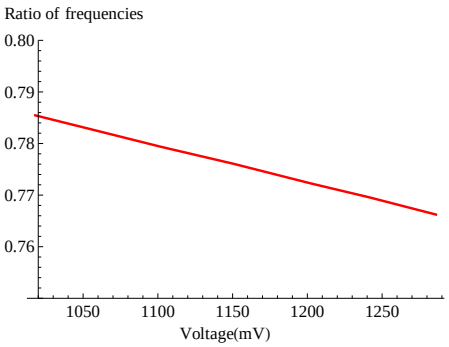
Figure 5.2: Histograms for two examples of behaviour of measured counter values for two RO pairs for seven different voltages (range from 1.018V to 1.286V). The histogram for RO pair 1 shows the undesired behaviour of counter values while the histogram for RO pair 2 shows the desired behaviour because the frequency distribution of counter values is approximately the same for all voltages.

		7–8	7–10
Voltage [V]	ΔU [mV]	HD_{intra} [%]	HD_{intra} [%]
1.200 → 1.018	-182	53.56	51.55
1.200 → 1.050	-150	51.65	48.05
1.200 → 1.100	-100	41.89	47.33
1.200 → 1.150	-50	23.67	36.45
1.200 → 1.200	**0**	**0.55**	**2.00**
1.200 → 1.262	62	34.19	42.01
1.200 → 1.286	86	38.66	43.49

Table 5.10: The difference of the PUF outputs measured at various voltages and the PUF outputs measured at nominal voltage 1.2V for 150 RO pairs and for selected positions 7–8 and 7–10.



(a) Difference of the change of frequency for two ROs in pair. Red curve represents the faster RO and blue curve is the slower RO.



(b) The change of ratio of frequencies for one RO pair in dependence on voltage.

Figure 5.3: Dependency of the frequencies of two ROs and their ratio on the change of voltage.

It can be seen that the influence of voltage on this ROPUF design is significant since the PUF outputs are very different. The influence of voltage is even larger for positions 7–10 since it includes the positions closer to the LSB, therefore, the stability is lower than for positions 7–8. The cause of this behaviour can be seen in Fig. 5.2. These histograms show how the counter values for two chosen RO pairs change with voltage. The upper histogram contains the counter values for both of the RO pairs and the bottom histograms show the counter values for each of the RO pairs in detail. We can see that the mean values of the counter values for RO pair 1 are very different for each voltage. Ideally, we would need the behaviour of another RO pair (RO pair 2) where the counter values would not significantly change with varying voltage. The reason of the difference in behaviour of the two RO pairs is still unclear and therefore our future work will address this issue. There is a suspicion that this issue is related to the mutual symmetry of ROs, i.e. with the placement of ROs which is discussed in Section 4.2.

The next experiment was to determine whether the length of ROs influence the behaviour of PUF when the voltage is changed. We performed measurements 1000 times for 90 RO pairs where ROs were five and seven stages long and the selected positions of counter values were 7–8. Table 5.11 shows the results of these measurements and as it can be seen, the change of length of ROs did not affect the PUF behaviour.

The high sensitivity of this ROPUF design to voltage is caused by the change of ratios of two frequencies of ROs in each pair. If we want the PUF outputs to remain stable at varying voltage, the ratios of the frequencies for each pair of ROs have to be the same. The reason is that the value of the

	Voltage [V]	ΔU [mV]	HD_{intra} [%]
5-stage ROs	1.204 → 1.101	-103	47.57
	1.204 → 1.289	85	53.28
7-stage ROs	1.204 → 1.101	-103	48.51
	1.204 → 1.289	85	50.76

Table 5.11: The difference of the PUF outputs with various voltages and the PUF outputs measured at nominal voltage 1.2V for selected positions 7–8 with 5-stage and 7-stage ROs.

Voltage [V]	ΔU [mV]	HD_{intra} [%]
1.201 → 1.180	-21	19.79
1.201 → 1.190	-11	10.49
1.201 → 1.193	-8	8.12
1.201 → 1.196	-5	5.76
1.201 → 1.201	**0**	**1.50**
1.201 → 1.207	6	5.04
1.201 → 1.212	11	9.78
1.201 → 1.222	21	19.40

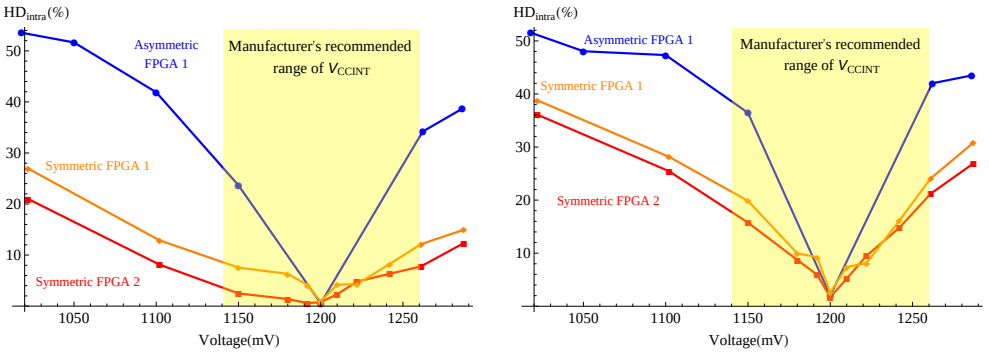
Table 5.12: The PUF outputs with various voltages from the range of 1.180V to 1.222V compared to the PUF outputs measured at nominal voltage 1.2V for selected positions 7–8.

16-bit counter can be determined as shown in Eq. 3.1:

$$Counter\ value = \frac{f_2}{f_1} \times 2^{16},$$

where f_1 is the frequency of the faster RO and f_2 is the frequency of the slower RO. Fig. 5.3(a) shows the dependency of the frequencies of two ROs in a pair on voltage. The higher the voltage is, the higher is the frequency of the oscillator. The change of ratio for one RO pair (the same pair as in Fig. 5.3(a)) is shown on the next Fig. 5.3(b). Ideally, the ratio should be constant, however the ratio is changing with the change of voltage.

In order to determine the interval of voltage in which the PUF may be operating, we performed more measurements. This time the voltage ranged from 1.180V to 1.222V and we used 150 RO pairs and positions 7–8. The results are presented in Table 5.12. The nominal voltage is 1.2V. If we want HD_{intra} to be about 5% on each side from the nominal voltage, then the interval for voltage might be approximately from 1.195V to 1.205V. Under these circumstances the correct behaviour of the PUF should be guaranteed.



(a) Dependence of HD_{intra} on voltage for positions 7–8.

(b) Dependence of HD_{intra} on voltage for positions 7–10.

Figure 5.4: Comparison of the behaviour of the proposed PUF when using mutually symmetric and asymmetric ROs for positions 7–8 and 7–10. The reference output for calculating HD_{intra} is the mean output from the PUF outputs measured at nominal voltage 1.2V. The yellow area represents the manufacturer's recommended range of FPGA's main power supply voltage V_{ccint}, which is from 1.14V to 1.26V.

5.7.3 Symmetric ROs

In the previous subsection we presented the results of the measurements for the proposed ROPUF design at varying voltage. It was shown that the influence of supply voltage is significant because the ratios of the frequencies of ROs in pairs change when the voltage varies. These results were obtained for the ROPUF design where the ROs were not mutually symmetric and there was no emphasis on the placement of the logic gates of each RO.

We can expect that the frequencies of ROs will change in a similar way when the ROs are mutually symmetric. Therefore, the ratios of the frequencies of RO pairs should be almost constant. In the next experiment, we placed the logic gates of each RO so that the ROs were mutually symmetric.

Table 5.13 shows the results of evaluated measurements for various voltages ranging from 1.022V to 1.287V on two FPGAs. These measurements were performed 1000 times for 50 RO pairs where all ROs had the same mutual placement of logic gates and Gray code was applied to the whole counter values. It can be seen that the improvement is significant compared to results presented in Table 5.10 where ROs were not mutually symmetric. Fig 5.4(a) and Fig. 5.4(b) present the comparison of the behaviour of the proposed ROPUF when using mutually symmetric and asymmetric ROs for positions 7–8 and 7–10. The results for HD_{intra} are not ideal, but they demonstrate the improvement when using symmetric ROs compared to asymmetric ROs and show the way for further investigation of the influence of the placement of ROs on the stability of the PUF outputs.

Positions 7–8

		FPGA 1	FPGA 2
Voltage [V]	ΔU [mV]	HD_{intra} [%]	HD_{intra} [%]
1.202 → 1.022	-180	21.01	26.97
1.202 → 1.102	-100	8.13	12.89
1.202 → 1.150	-52	2.49	7.55
1.202 → 1.180	-22	1.39	6.32
1.202 → 1.192	-10	0.58	4.13
1.202 → 1.202	**0**	**0.79**	**0.91**
1.202 → 1.210	8	2.31	4.21
1.202 → 1.222	20	4.80	4.30
1.202 → 1.242	40	6.34	8.26
1.202 → 1.261	69	7.79	12.15
1.202 → 1.287	85	12.34	14.95

Positions 7–10

		FPGA 1	FPGA 2
Voltage [V]	ΔU [mV]	HD_{intra} [%]	HD_{intra} [%]
1.202 → 1.022	-180	36.10	38.80
1.202 → 1.102	-100	25.45	28.16
1.202 → 1.150	-52	15.76	19.86
1.202 → 1.180	-22	8.72	9.95
1.202 → 1.192	-10	5.94	9.28
1.202 → 1.202	**0**	**1.69**	**2.58**
1.202 → 1.210	8	5.20	7.39
1.202 → 1.222	20	9.51	8.13
1.202 → 1.242	40	14.79	16.14
1.202 → 1.261	69	21.20	24.09
1.202 → 1.287	85	26.91	30.82

Table 5.13: The PUF outputs with various voltages from the range of 1.022V to 1.287V compared to the PUF outputs measured at nominal voltage 1.2V and for selected positions 7–8 and 7–10 for two FPGAs.

5.8 Influence of temperature

This section examines the influence of changes in temperature on the proposed ROPUF. The statistical properties of PUF using both symmetric and asymmetric ROs will be compared. Fig. 5.5 depicts our measurement setup. For the purpose of our experiment, we performed measurements at different temperatures. For these measurements, FPGA was preheated to a preset temperature (e.g. 40 °C) with ROs enabled. Each of the measurements was

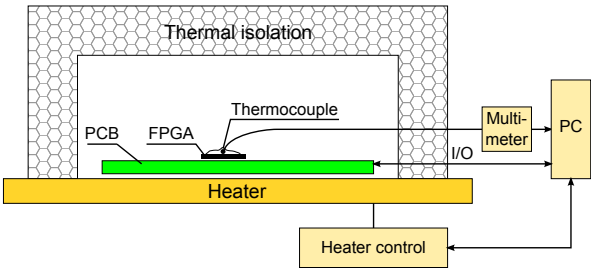


Figure 5.5: Measurement setup for measuring at elevated temperatures. PCB is the Digilent Basys 2 prototyping board. The FPGA is a Xilinx Spartan3E-100 CP132. The heater was used to preheat the FPGA to a target temperature.

carried out when the temperature measured on the package of the FPGA stabilised at the given value. We used three Digilent Basys 2 FPGA boards for this experiment.

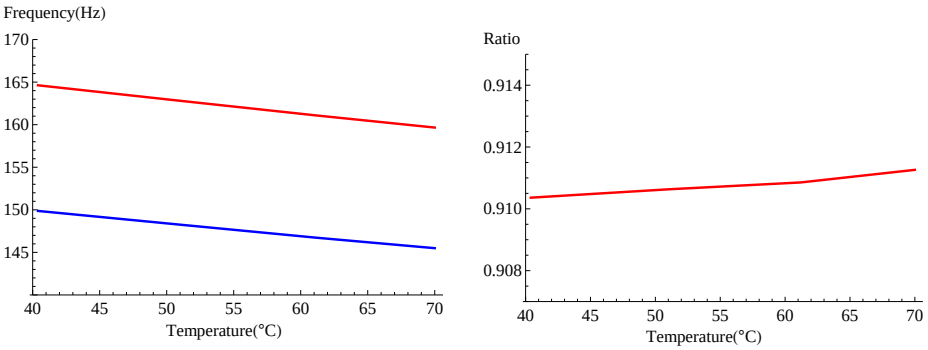
The values of HD_{intra} were obtained by averaging the PUF outputs at each temperature. The average PUF output is obtained as the majority of each column when the PUF outputs are written in the form of matrix where each row represents one PUF output.

Table 5.14 displays the values of HD_{intra} for three FPGAs for asymmetric and symmetric ROs. As in the previous Section 5.7, the RO symmetry consists of the mutual placement of the gates of each RO but not of the interconnects between them. The column "temperature" presents the temperatures at which the PUF outputs are compared. The values of HD_{intra} for symmetric and asymmetric ROs are almost equivalent for small differences in temperature, but for larger changes in temperature there is a visible improvement of the PUF behaviour when symmetric ROs are used. This is a very similar result to that which was presented in Section 5.7 where the stability of the PUF outputs was also increased by using symmetric ROs.

It was shown before that the resulting counter value depends on the ratio of the frequencies of the two ROs which are influenced by various physical conditions. However, it can be expected that the frequencies should be affected in a similar way. Therefore, the change of the frequencies due to varying temperature should be eliminated by their ratio. Fig. 5.6(a) shows the change of the frequencies of two ROs with increasing temperature. The higher the temperature, the smaller are the frequencies of ROs.

Asymmetric ROs					
FPGA 1		FPGA 2		FPGA 3	
Temperature [°C]	HD_{intra} [%]	Temperature [°C]	HD_{intra} [%]	Temperature [°C]	HD_{intra} [%]
36.7 → 41.2	2.67	38.4 → 42.3	2.67	37.7 → 41.8	1.0
36.7 → 51.8	7.67	38.4 → 50.1	6.67	37.7 → 50.9	5.0
36.7 → 60.4	9.33	38.4 → 60.3	9.33	37.7 → 61.3	7.0
36.7 → 71.1	11.33	38.4 → 69.9	12.67	37.7 → 70.1	12.0
Symmetric ROs					
FPGA 1		FPGA 2		FPGA 3	
Temperature [°C]	HD_{intra} [%]	Temperature [°C]	HD_{intra} [%]	Temperature [°C]	HD_{intra} [%]
33.0 → 42.4	2.67	34.4 → 40.9	1.67	34.5 → 41.1	3.67
33.0 → 50.5	3.67	34.4 → 50.5	3.0	34.5 → 51.4	6.0
33.0 → 60.6	3.67	34.4 → 60.8	4.67	34.5 → 60.6	7.0
33.0 → 71.0	4.67	34.4 → 70.2	5.33	34.5 → 70.4	7.33

Table 5.14: Evaluation of HD_{intra} for 150 asymmetric/symmetric RO pairs and selected positions 7–8 at different temperatures.



(a) Difference in the change of frequency for two ROs in pair. Red curve represents the faster RO and blue curve is the slower RO.

(b) The change of ratio of frequencies for one RO pair in dependence on temperature.

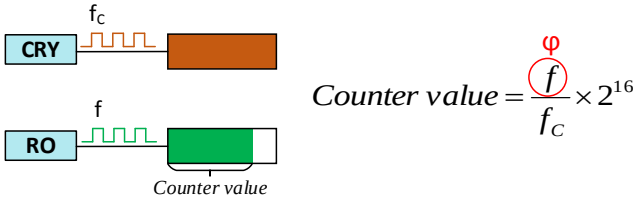
Figure 5.6: Dependency of the frequencies of two ROs and their ratio on the temperature change.

Ideally, we need the ratios of the frequencies to remain constant in time, but as it is shown in Fig. 5.6(b), the ratio is not constant at varying temperatures. Therefore, we need to minimize the difference in the ratio of the frequencies at varying temperatures (or other influences) and using symmetric ROs may be one of the possible solutions. By using symmetric ROs the frequencies of the ROs are closer to each other, hence they should be more likely influenced in the same way when the physical conditions are changed. However, it is still necessary to investigate the relationship between PUF stability at varying environmental conditions and the symmetry of ROs further.

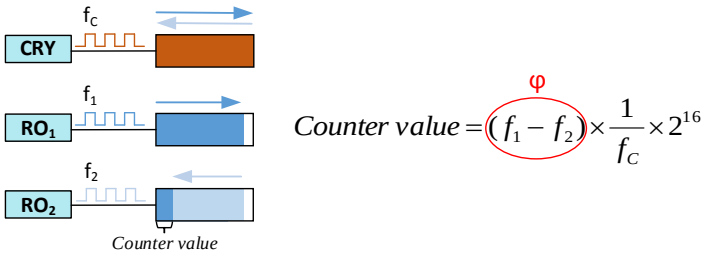
5.9 Comparison of different methods

The whole chapter was devoted to experiments and their evaluation in order to present the properties of the proposed PUF design. In this section we compare three different methods of using the ROs for generating the PUF outputs. We will show that the proposed one is the most stable against change of physical conditions, specifically the supply voltage and temperature.

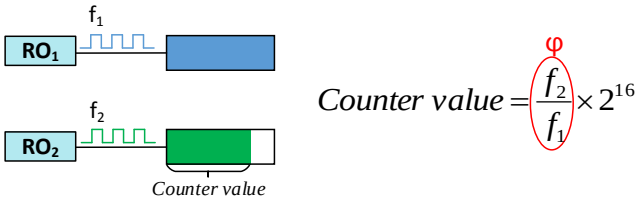
Since the resulting counter value in our proposal is determined by the ratio of the frequencies of the two ROs in a pair, this method can be considered as a differential measurement because even though the physical conditions such as supply voltage or temperature have a significant influence on the frequencies of the ROs, the change in the ratio of the frequencies is much smaller. To show the difference between some other possible approaches, Fig. 5.7 presents three different methods for using ROs to obtain some counter value that would be later processed. The first two approaches require a stable reference clock such as a crystal oscillator. There is a formula for each method to determine



(a) A basic approach, where the counter value is calculated for each RO for the same amount of time given by a crystal oscillator (CRY) with a stable frequency f_C which serves as a reference clock. It is assumed that the frequency of the crystal oscillator is greater than the frequency of any RO.



(b) A subtraction method. First, the value of the counter is incremented for every rising edge of the first RO (RO_1). Then the counter value is decremented for every rising edge of the second RO (RO_2). The incrementing and decrementing runs for the same amount of time given by a crystal oscillator (CRY) with a stable frequency f_C. The assumption is that the frequency of the crystal oscillator is greater than the frequency of any RO and that $f_1 > f_2$.



(c) A frequency ratio method that is used in our PUF design. The resulting counter value is given by the ratio of the frequencies f_1 and f_2. There is no reference clock as opposed to the previous cases. It is assumed that $f_1 > f_2$

Figure 5.7: Three different methods of using the ROs for PUF.

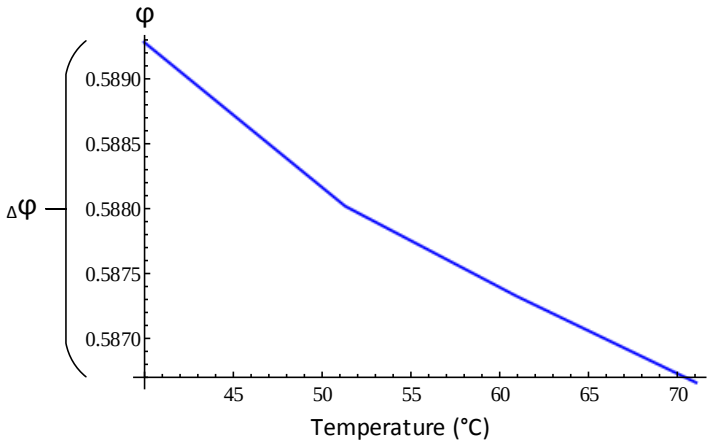


Figure 5.8: Behaviour of φ at varying temperature. In this example, $\varphi = \frac{f_2}{f_1}$, therefore, φ represents ratio of the frequencies of two ROs.

the resulting counter value and the part of the equation that is dependent on the physical conditions is highlighted with a red circle and we will denote this part as φ.

The first method is similar to our approach, but in place of two ROs forming a pair there is always one stable reference clock (crystal oscillator) and one RO. In the case shown in Fig. 5.7(a) it is assumed that the frequency of the reference clock is greater than the frequency of any RO and the resulting counter value is then calculated from the ratio of the frequency of RO and reference clock. However, because we assume the reference clock to be stable at varying temperatures, then φ is only the frequency of the RO ($\varphi = f$), since the resulting counter value depends solely on the frequency of the RO.

In addition, if we have n ROs, we can extract only $n \times w$ (w is the number of bits extracted from each counter value) bits for the PUF response. Therefore, even though the design would be very simple, it would not be very efficient since it requires a lot of ROs in order to generate long PUF outputs.

The second method shown in Fig. 5.7(b) also uses a stable reference clock and it is assumed that its frequency is greater than the frequency of any RO. At first, the counter is incrementing with the frequency given by the first RO. Then the value in the counter is decremented by the second RO and both the incrementation and decrementation take the same time given by the reference clock. In this case, $\varphi = f_1 - f_2$ because the resulting counter value is derived from the difference of the frequencies.

From the perspective of using the ROs in pairs, this method is similar to the method we proposed. The theoretical maximum number of bits that can be extracted using this method is $\binom{n}{2} \times w$ just like in our proposal.

	Temperature			Supply voltage
	FPGA 1	FPGA 2	FPGA 3	FPGA 1
	$\frac{\Delta\varphi}{\varphi_{max}}$[%]	$\frac{\Delta\varphi}{\varphi_{max}}$[%]	$\frac{\Delta\varphi}{\varphi_{max}}$[%]	$\frac{\Delta\varphi}{\varphi_{max}}$[%]
crystal	3.42	3.01	2.97	22.18
subtraction	4.08	4.29	3.78	33.21
ratio	0.27	0.16	0.14	2.41

Table 5.15: The change of φ at varying temperatures and voltages for the three different approaches. The frequencies were measured for 300 ROs. In case of temperature, the measurements were performed on three FPGAs at temperatures ranging from 40°C to 71°C. For supply voltage, the measurements were performed on one FPGA and the range of supply voltage was from 1.018V to 1.286V.

The last method is shown in Fig. 5.7(c) and it is the method used in our PUF design described in Chapter 3. As it was explained before, it is based on a ratio of the frequencies of two ROs in a pair, hence $\varphi = \frac{f_2}{f_1}$. The maximum number of bits for PUF is $\binom{n}{2} \times w$ as was the case with the previous method based on subtraction.

To evaluate the resistance of each method to varying physical conditions, we measured the frequencies of 300 ROs at different physical conditions. In case of varying temperature, we performed the measurements on three FPGAs, while the measurements concerning the change of voltage were performed on one FPGA. Depending on the method, we calculated the values of φ from the frequencies at different physical conditions and determined the value of $\Delta\varphi$ as $\Delta\varphi = \varphi_{max} - \varphi_{min}$. See Fig. 5.8 for an example of determining $\Delta\varphi$ for varying temperatures. From $\frac{\Delta\varphi}{\varphi_{max}}$ we can observe the resistance of each method to the change of physical conditions.

The results of this evaluation are shown in Table 5.15 for all presented methods. It can be seen that the change of φ at varying temperatures and voltages is the lowest for the method used in our PUF design (ratio of the frequencies). We can also notice that the change of φ is considerably larger for varying voltage than for varying temperature. This may indicate that the frequencies of ROs are more dependent on supply voltage than on temperature.

Conclusion

Physical Unclonable Functions are now a very popular research topic in hardware security because they are increasingly used in proposals of cryptographic protocols and security architectures as they provide a unique fingerprint of the device. One of the major applications of PUF is key generation which is used to generate cryptographic keys instead of storing them in a non-volatile memory which is difficult to secure. However, there are more applications and scenarios in which PUFs can be used, including device identification, authentication, anti-counterfeiting, binding software to hardware platforms; they can also be integrated into cryptographic algorithms.

In this thesis we dealt with the statistical analysis of behaviour of the proposed PUF design and improvements to this proposal. It builds upon our previous work [13], which was later extended and published in [14, 15, 16]. The goals of this thesis were to analyse and discuss the properties of the proposed PUF design and further improve the design in order to enhance its statistical properties.

The literature research regarding PUFs is provided in the first two chapters. In Chapter 1 we introduced the topic of PUFs, described PUFs in general and also presented the properties that we may require of PUFs. Finally, this chapter presented the applications where PUFs can be used. Chapter 2 discussed possible classifications of PUFs in dependence on their properties or the nature of their features. After the discussion, the chapter introduced examples of various PUF constructions. We focused mainly on PUFs suitable for FPGAs and we put emphasis on PUF constructions based on ROs since our PUF design is based on them.

Chapter 3 is devoted to the description of the PUF, proposed in [13]. Its main feature is the ability to extract more output bits from each RO pair compared to the classical approach, which additionally requires the ROs to be mutually symmetric. Our design no longer requires the ROs to be mutually symmetric, hence it is easier to implement and also more area efficient. The second part of this chapter discussed the properties of the proposed PUF

design.

In Chapter 4 we proposed improvements of the PUF design in order to enhance its statistical properties. Since the proposed PUF is based on ROs whose frequencies are dependent on various physical conditions, our goal was to eliminate this dependence. Therefore the improvements were proposed mainly to decrease the influence of physical conditions on the stability of the PUF outputs.

Finally, Chapter 5 was devoted to experiments, measurements and also presenting and discussing their results. The measurements were performed mainly on Digilent Basys 2 FPGA boards (Xilinx Spartan3E-100 CP132). We have shown that the proposed design can be used as PUF on FPGAs.

To evaluate the PUF behaviour on other types of FPGAs, we performed measurements on Digilent Nexys 3 FPGA boards (Xilinx Spartan-6). The results of statistical evaluation indicated that the PUF implemented on Nexys 3 works properly.

We also investigated the behaviour of PUF when using counters whose maximum operating frequency is too slow for the incoming signal from the ROs. This causes the counters to miss some clock pulses and the resulting counter value is then incorrect. Our finding was that it does not have any impact on the statistics of the PUF outputs.

In Section 5.3 we presented the results for one of our improvements of the design, which is the application of Gray code to the selected parts of the counter values. This increased the stability of the PUF outputs and allowed us to extract four bits of the PUF output from each RO pair with almost the same quality as two bits without Gray code.

Another improvement presented in this work is the use of symmetric ROs in order to enhance the stability of the PUF outputs at varying physical conditions. We statistically evaluated the PUF with symmetric ROs at stable physical conditions and the results were shown in Section 5.6. These results were very similar to those for the PUF design with asymmetric ROs.

Section 5.7 and Section 5.8 analysed the influence of change in voltage and temperature. It turned out that the influence of supply voltage on the proposed PUF design is significant. In order to achieve HD_{intra} to be approximately 10%, the range of supply voltage would have to be from 1.190V to 1.212V. This corresponds to the change of voltage $\Delta U = 10\text{mV}$ from the nominal supply voltage.

We proposed a placement of the logic gates of ROs such that all ROs are mutually symmetric. As a result, the PUF outputs were considerably more stable and the influence of voltage was not so significant as in the case of asymmetric ROs.

The evaluation of the influence of temperature on the statistical properties of the proposed PUF design was performed on three Digilent Basys 2 FPGA boards. The stability of the PUF outputs are almost equivalent for small

changes in temperature, however, when the change in temperature is greater, the ROPUF using symmetric ROs gives better results.

Finally, we compared three different methods of using the ROs for generating the PUF outputs. They are based on measuring the frequency against a fixed crystal oscillator, difference of frequencies and ratio of frequencies (our method). We have shown that our method, which is based on ratio of frequencies, is the most resistant against varying physical conditions.

We can conclude that the goals of this thesis were fully accomplished. We analysed the properties of the proposed PUF and tested the quality of the PUF outputs at various physical conditions. We proposed suitable modifications of the PUF that improved the quality of its output. Both Gray code and symmetry of ROs improved the stability of the PUF outputs when the physical conditions are varying.

The results presented in this work are a motivation for future research in this area. Here are the possibilities for future research:

- Further investigate the influence of supply voltage and temperature together with the placement of ROs.
- Investigate the influence of aging on the proposed PUF design.
- Examine possible attacks on this PUF.
- Design a TRNG based on the proposed PUF.

Bibliography

[1] Bossuet, L.; Ngo, X. T.; Cherif, Z.; Fischer, V. A PUF based on a transient effect ring oscillator and insensitive to locking phenomenon. In *IEEE Transactions on Emerging Topics in Computing*, 2014, 2.1, pp. 30–36.

[2] Busch, H.; Sotáková; M.; Katzenbeisser, S.; Sion, R.: *The PUF Promise (Short Paper)*. 2010, [Cited 2016-05-02]. Available from: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.187.324&rep=rep1&type=pdf`

[3] Delvaux, J.; Verbauwhede, I. Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise. In *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*, 2013, pp. 137–142.

[4] Digilent: *Digilent, Adept*. [Cited 2016-05-02]. Available from: `https://reference.digilentinc.com/digilent_adept_2`

[5] Digilent: *Digilent, Adept SDK*. [Cited 2016-05-02]. Available from: `https://reference.digilentinc.com/digilent_adept_2`

[6] Digilent: *Digilent Basys2 Board Reference Manual*. [Cited 2016-05-02]. Available from: `https://reference.digilentinc.com/_media/basys2:basys2_rm.pdf`

[7] Digilent: *Digilent Nexys3 Board Reference Manual*. [Cited 2016-05-02]. Available from: `https://reference.digilentinc.com/_media/nexys:nexys3:nexys3_rm.pdf`

[8] Gassend, B. *Physical Random Functions*. Dissertation thesis. Massachusetts Institute of Technology, 2003, [Cited 2016-05-02]. Available from: `http://www.textfiles.com/bitsavers/pdf/mit/lcs/tr/MIT-LCS-TR-881.pdf`

[9] Gassend, B.; Clarke, D.; Dijk, M.; Devadas, S. Silicon Physical Random Functions. In *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 148–160.

[10] Herder, C.; Yu, M. D.; Koushanfar, F.; Devadas, S. Physical Unclonable Functions and Applications: A Tutorial. In *Proceedings of the IEEE*, 2014, 102.8, pp. 1126–1141.

[11] Holcomb, D. E.; Burleson, W. P.; Fu, K. Power-up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. In *IEEE Transactions on Computers*, 2009, 58.9, pp. 1198–1210.

[12] Katzenbeisser, S; Kocabaş; Ü.; Rožić; V.; Sadeghi, A.; Verbauwhede, I.; Wachsmann, Ch. PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon. In *Cryptographic Hardware and Embedded Systems–CHES 2012*. Springer Berlin Heidelberg, 2012, pp. 283–301.

[13] Kodýtek, F. *Physical Unclonable Function on FPGAs*. Bachelor thesis. Czech Technical University in Prague, Faculty of Information Technology, 2014.

[14] Kodýtek, F.; Lórencz, R. A Design of Ring Oscillator Based PUF on FPGA. In *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2015 IEEE 18th International Symposium on*. IEEE, 2015, pp. 37–42.

[15] Kodýtek, F.; Lórencz, R. Proposal and Properties of Ring Oscillator-Based PUF on FPGA. In *Journal of Circuits, Systems and Computers*, 2016, 25.03, 1640016.

[16] Kodýtek, F.; Lórencz, R.; Buček, J. Improved ring oscillator PUF on FPGA and its properties. In *Microprocessors and Microsystems*, 2016. (In Press)

[17] Kumar, S. S.; Guajardo, J.; Maes, R.; Schrijen, G.; Tuyls, P. Extended abstract: The butterfly PUF protecting IP on every FPGA. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 67–70.

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

[19] Lim, D.; Lee, J. W.; Gassend, B.; Suh, G. E.; Dijk, M.; Devadas, S. Extracting secret keys from integrated circuits. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2005, 13.10, pp. 1200–1205.

[20] Maes, R. *Physically Unclonable Functions: Constructions, Properties and Applications*. Dissertation thesis. Katholieke Universiteit Leuven, 2012, [Cited 2016-05-02]. Available from: `https://securewww.esat.kuleuven.be/cosic/publications/thesis-211.pdf`

[21] Maes, R.; Tuyls, P.; Verbauwhede, I. Intrinsic PUFs from Flip-flops on Reconfigurable Devices. In *3rd Benelux workshop on information and system security (WISSec 2008)*, 2008, [Cited 2016-05-02]. Available from: `https://www.researchgate.net/profile/Roel_Maes/publication/228615879_Intrinsic_PUFs_from_flip-flops_on_reconfigurable_devices/links/0912f51126478b8661000000.pdf`

[22] Maes, R.; Verbauwhede, I. Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In *Towards Hardware-Intrinsic Security*. Springer Berlin Heidelberg, 2010, pp. 3–37.

[23] Maiti, A.; Schaumont, P. Improving the quality of a Physical Unclonable Function using configurable Ring Oscillators. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*. IEEE, 2009, pp. 703–707.

[24] Marchand, C.; Bossuet, L.; Cherkaoui, A. Enhanced TERO-PUF Implementations and Characterization on FPGAs. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 282–282.

[25] Morozov, S.; Maiti, A.; Schaumont, P. An Analysis of Delay Based PUF Implementations on FPGA. In *Reconfigurable Computing: Architectures, Tools and Applications*. Springer Berlin Heidelberg, 2010, pp. 382–387.

[26] Pappu, R. S. *Physical One-Way Functions*. Dissertation thesis. Massachusetts Institute of Technology, 2001, [Cited 2016-05-02]. Available from: `http://alumni.media.mit.edu/~pappu/pdfs/Pappu-PhD-POWF-2001.pdf`

[27] Petit, J.; Bösch, Ch.; Feiri, M.; Kargl, F. On the potential of PUF for pseudonym generation in vehicular networks. In *Vehicular Networking Conference (VNC), 2012 IEEE*. IEEE, 2012, pp. 94–100.

[28] Platonov, M. *SRAM-Based Physical Unclonable Function on an Atmel ATmega Microcontroller*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2013.

[29] Platonov, M.; Hlaváč, J.; Lórencz, R. Using Power-Up SRAM State of Atmel ATmega1284P Microcontrollers as Physical Unclonable Function for Key Generation and Chip Identification. In *Information Security Journal: A Global Perspective*. 2013, 22.5–6, pp. 244–250.

[30] Ruhkin, A. et al. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. *NIST Special Publication 800-22 Revision 1a.* 2010, [Cited 2016-05-02]. http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf

[31] Su, Y.; Holleman, J.; Otis, B. A 1.6pJ/bit 96% Stable Chip-ID Generating Circuit using Process Variations. In *IEEE Journal of Solid-State Circuits*, 2008, 43.1, pp. 69–77.

[32] Suh, G. E.; Devadas, S. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 9–14.

[33] Suzuki, D.; Shimizu, K. The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes. In *Cryptographic Hardware and Embedded Systems, CHES 2010*. Springer Berlin Heidelberg, 2010, pp. 366–382.

[34] Tuyls, P.; Schrijen, G.; Škorić, B.; Geloven, J.; Verhaegh, N.; Wolters, R. Read-Proof Hardware from Protective Coatings. In *Cryptographic Hardware and Embedded Systems-CHES 2006*. Springer Berlin Heidelberg, 2006, pp. 369–383.

[35] Varchola, M.; Drutarovsky, M. New High Entropy Element for FPGA Based True Random Number Generators. In *Cryptographic Hardware and Embedded Systems, CHES 2010*. Springer Berlin Heidelberg, 2010, pp. 351–365.

[36] Xilinx: *ISE Design Suite*. [Cited 2016-05-02]. Available from: http://www.xilinx.com/products/design-tools/ise-design-suite.html

[37] Xilinx: *Spartan-3E FPGA Family Data Sheet*. [Cited 2016-05-02]. Available from: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

[38] Xin, X.; Kaps, J.; Gaj, K. A Configurable Ring-Oscillator-Based PUF for Xilinx FPGAs. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*. IEEE, 2011, pp. 651–657.

[39] Yin, C. -E. D.; Qu, G.: LISA: Maximizing RO PUF's secret extraction. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 100–105.

Appendix A

FPGA

Field-programmable gate array (FPGA) is an integrated circuit which can be configured by a designer even after manufacturing. FPGAs have a general structure consisting of an array of programmable logic blocks and programmable interconnections between these blocks. The logic blocks can be configured to perform various combinational functions. The configuration of a FPGA is created using a hardware description language (HDL), such as VHDL or Verilog.

Fig. A.1 shows the architecture of Xilinx Spartan-3E which we used for the implementation of our PUF design and performing the measurements. It consists of five fundamental programmable functional elements [37]:

- **Configurable Logic Blocks (CLBs)** are probably the most important elements of the FPGAs. They can be configured so that they can perform complex combinational functions or store data. Each CLB consists of *slices*, which are further divided into *logic cells*. These logic cells contain Look-Up Tables (LUTs), multiplexers, carry logic and flip-flops. LUT is a RAM-based function generator. In case of Spartan-3E, LUTs have four logic inputs and a single output, therefore any four-variable Boolean logic operation can be implemented in one LUT. When functions with more inputs are needed, they can be implemented by cascading LUTs.

- **Input/Output Blocks (IOBs)** realise the communication with the outside world. They control the flow of data between the I/O pins and the internal logic of the device. The IOBs are located on the edge of the FPGA.

- **Block RAM (BRAM)** provides data storage in the form of 18-Kbit dual-port blocks; its content can be defined in the design for the FPGA. It is initialised after power-up, therefore it cannot be used as a source of randomness for PUF.

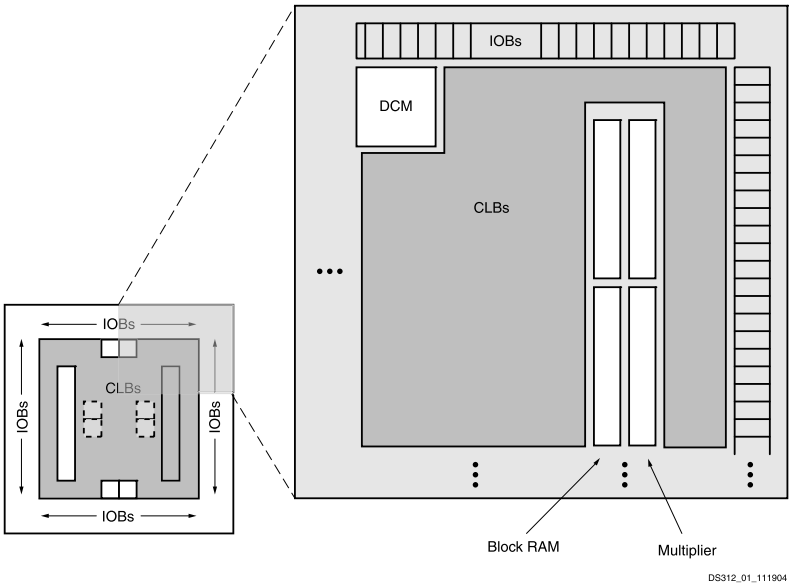


Figure A.1: Spartan-3E family architecture.[37]

- **Multiplier Blocks** calculate the product of two input 18-bit binary numbers.
- **Digital Clock Manager (DCM) Blocks** provide solutions for distributing, delaying, multiplying, dividing and phase-shifting clock signals.

All of the configurable elements are interconnected by a rich network of traces, transmitting signals among them. Each functional element has a switch matrix that permits multiple connections to the routing.

Digilent Basys 2

Most of the measurements presented in this work were performed on Digilent Basys 2 FPGA boards [6]. This board contains FPGA from the family of Xilinx Spartan-3E, specifically Xilinx Spartan3E-100 CP132. This FPGA contains 240 CLBs and each CLB consists of four slices. Each slice contains two LUTs and two flip-flops, resulting in 1920 LUTs and flip-flops that are available for the design. This FPGA also contains a fast dual-port BRAM 72KBit in size.

For circuit inputs, there are four pushbuttons and eight slide switches available. The outputs can be displayed using a four-digit seven-segment LED display or eight LEDs. For the communication with FPGA, the board provides a PS/2 port, VGA, USB and others.

The FPGA has to be configured in order to perform some useful function. The configuration is stored in a *bitstream*, which can be transferred to the

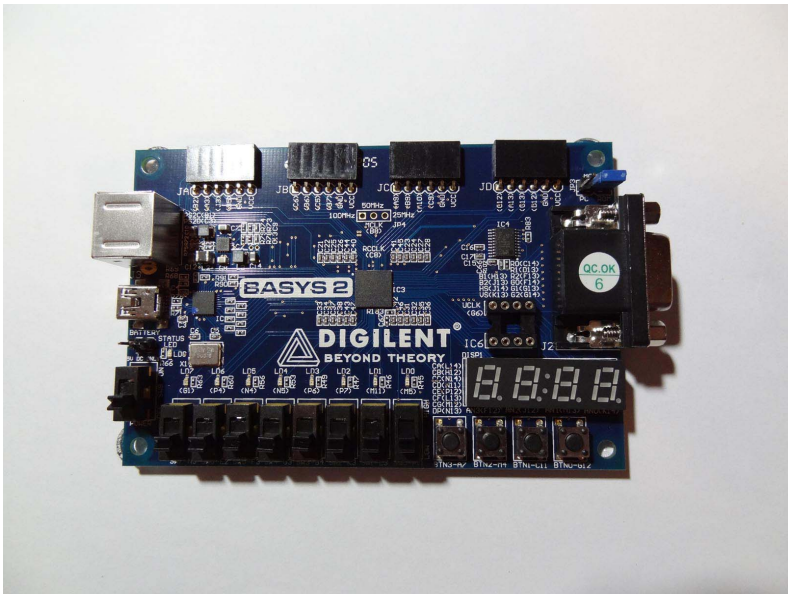


Figure A.2: Digilent Basys 2.

FPGA via the FPGA's JTAG programming port using the Adept software [4]. The bitstream can be loaded directly into FPGA, thereby configuring it, or it can be transferred to non-volatile ROM called "Platform Flash" that can automatically transfer a stored bitstream to the FPGA at power-on or reset event.

For more detailed description of this board, see [6].

Appendix B

Acronyms

CLB Configurable logic block

CRP Challenge-response pair

ECC Error correcting code

FPGA Field-programmable gate array

HD Hamming distance

LSB Least significant bit

PUF Physical Unclonable Function

MSB Most significant bit

RFID Radio-frequency identification

RO Ring oscillator

ROPUF Ring oscillator physical unclonable function

SRAM Static random-access memory

TERO Transient effect ring oscillator

TRNG True random number generator

Appendix C

Contents of the enclosed CD

- `readme.txt` ..... the file with CD contents description
- `app` ..... the directory with executables
- `data` ..... the directory containing measured data
  - `nist_data` ..... data for NIST STS
  - `of_detect_verif` ..... data for evaluation of overflow detection
  - `RO_intervals` ..... data for PUF evaluation
    - `Basys2` ..... data obtained from Basys 2
    - `Nexys3` ..... data obtained from Nexys 3
  - `RO_intervals_cb` ..... data obtained from circuit with slow counters
  - `temperature` ..... data obtained at various temperatures
  - `voltage` ..... data obtained at various voltages
- `src` ..... the directory of source codes
  - `auxiliary_materials` ..... Wolfram Mathematica notebooks
  - `C_programs` ..... implementation sources of programs for measurements
  - `Digilent_Adept_SDK` ..... Digilent Adept SDK
  - `scripts` ..... scripts used for statistical evaluation
  - `thesis` ..... the directory of LaTeX source codes of the thesis
  - `Xilinx_projects` ..... projects for Xilinx ISE
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format