2012

Design and evaluation of a delay-based FPGA physically unclonable function

Aaron Mills
Iowa State University

Design and evaluation of a delay-based FPGA physically unclonable function

by

Aaron Joseph Mills

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:

Joseph Zambreno, Co-major Professor

Doug Jacobson, Co-major Professor

Chris Chu

Iowa State University

Ames, Iowa

2012

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to those who helped me with various aspects of research and the writing of this thesis. I want to thank my fellow graduate students Sudhanshu Vayas and Michael Patterson for their help and patience in brainstorming. I want to thank my committee members for the input and sense of direction they provided: Dr. Joe Zambreno, Dr. Chris Chu, and Dr. Doug Jacobson. I also want to thank Dr. Phillip Jones for his support on the usage of the Xilinx design tools.

I thank my many undergraduate friends at ISU who provided quite convenient distractions when I needed a break.

Of course, last but not least, I thank my parents who have supported and encouraged me in every way, and who raised me with the skills to succeed. It was they who instilled in me the love of learning which has come to fruition in this thesis.

All the FPGAs and test equipment used for this thesis was made available by the Reconfigurable Computing Laboratory at Iowa State University.

ABSTRACT

The Physically Unclonable Function (PUF) is gaining increasing interest for its potential use as a hardware primitive in secure computing systems. In the most basic sense, a PUF is a device that harnesses the natural entropy in a physical system. A delay-based PUF in particular depends on the process variation that is inherent in the manufacturing of any integrated circuit. A particular instance of an ideal PUF will consistently output a particular bit pattern. Each instance of the exact same circuit, however, will produce a substantially different pattern. As a result, having full knowledge of its design will not help an attacker to predict its output, nor to successfully clone such a device.

For this research, a new PUF variant was developed on an FPGA, and an evaluation of its quality is performed. It is conceptually similar to PUFs developed using standard SRAM cells, except it utilizes general FPGA reconfigurable fabric, which offers several advantages. First, it allows greater control over the position and arrangement of each PUF cell. This flexibility increases our ability to study the various factors that impact a PUF's performance. Second, the PUFs can be reset without requiring the entire device to be reset, which is needed for the application of error correction. Third, it becomes possible to access the output of a large PUF array in parallel rather than word-by-word as in the case with standard SRAM. This can decrease the time it takes to retrieve the PUF output.

A quantitative comparison between our approach and other recent PUF designs indicates that our design is competitive in terms of repeatability within a given instance, and uniqueness between instances. A single PUF cell consumes only a single FPGA Slice, and has very low dynamic power dissipation, making it suitable for authentication applications on resource-constrained embedded systems. However, the design can also be tuned to achieve desired response characteristics which broadens the potential range of applications.

CHAPTER 1. INTRODUCTION

A great deal of attention is traditionally given to the problem of securing computing systems at the software level. Comparatively less is given to securing the hardware on which the software runs. A Xilinx whitepaper [20] discusses a number of threats to hardware design.

- *Reverse engineering* involves a third party which examines a design (layout, components, firmware) with the goal of reconstructing it for future development.
- *Cloning* involves less of a desire to understand a design; instead the attacker simply intends to copy and resell a design, without incurring the overhead of development costs. This is also referred to as counterfeiting.
- *Overbuilding* occurs when a subcontractor builds more units than have been ordered for fabrication by an Original Equipment Manufacturer. The intent is to sell the units for themselves. This is also a form of counterfeiting.
- *Tampering* occurs when an attacker attempts to gain unauthorized access to an electronic system, such as an to modify its firmware.

Two major themes here are the theft of intellectual property and the production of counterfeit components. Counterfeit components are not just a concern due to the increasing cost burden they place on companies that design or supply integrated circuits. They are also a potential threat to public safety, since counterfeit components are often produced from poorly controlled processes or from discarded defective materials–a counterfeiting operation might be quite successful simply re-marking and selling scrap components [25]. Components produced in these ways are likely to cause the failure of the system in which they are used. In a recent

example, it is suspected that the failure of the Russian Phobos-Grunt spacecraft was due to counterfeit memory chips which were not sufficiently hardened against radiation [21].

Another theme is the ability of attacker to physically tamper with electronic hardware with an intent to extract secret information. This can be done in a variety of ways. One method is differential power analysis. Another is method is to chemically etch away the casing of an integrated circuit and simply examine the die; a third is to induce instruction-flow faults in a CPU by using clock glitches. The potential for these kinds of techniques to cheaply extract RSA and DES secret keys from smart card memories has been recognized since at least 1997[4]. Of course today, extracting secret keys from smart cards is even easier–algorithms such as COMP128-1 can be broken in just a few minutes with kits purchased cheaply online [7]. This demonstrates the great challenge of ensuring security when an attacker has physical access to a device.

Thus, the idea of the Physically Unclonable Function was born from the the need for tamper-resistant, unclonable hardware [34]. Although absolute security has been shown time and again to be an unreachable ideal, if properly implemented and integrated, PUFs have the potential to offer a very high degree of security at very low cost.

1.1 Contributions

The main goal of this project is to explore the properties of the delay-based physically unclonable function, using a unique memory-type PUF that has been designed and implemented on an FPGA. This new design is conceptually simple and consumes minimal FPGA resources. The design is demonstrated using a variety of empirical tests to be suitable for hardware authentication applications. The project applies Hardware Description Language (HDL) hard-macros and an architecture-level understanding of the FPGA to ensure the accuracy and reliability of test results.

1.2 Organization

This thesis is organized as follows. In Chapter 2, a general description of Physically Unclonable Functions (PUFs) is provided, as well as a brief discussion of their major applications. A discussion of the differences between ASICs and FPGAs, as far as PUF research is concered, is also included here. Since this thesis is mainly concerned with the design of PUFs, in Chapter 2 a literary survey is presented which summarizes the major delay-based PUFs which have been prosed to date. A commentary is also provided which places the PUF designed for this thesis in context with the existing designs. Chapter 4 focuses on the design of the PUF created during this project. Here the principles behind its operation are discussed at length. Chapter 5 describes the metrics that were applied and experimentation that was performed to evaluate the performance of the PUF. It also includes a discussion of the applications for which it may best be suited. Finally, Chapter 6 concludes the thesis and discusses possible future efforts related to the discoveries made in this thesis.

CHAPTER 2. BACKGROUND

Research on PUFs and process variation has been gaining increasing interest since the concept of the PUF was formally introduced by Pappu, et. al. in [34] in 2001. In the simplest sense, a PUF is a device whose transfer function exploits physical phenomena in a way that cannot be replicated, even if the full design is known. The PUF designs that have been proposed over the years are diverse, ranging from Pappu's original design, which relies on the unpredictability of light refraction on a textured surface, to the delay-based silicon PUFs which this paper focuses on.

At the behavioral level, a PUF is often thought of as a hardware version of a cryptographic hash function. It is sometimes also referred to as a physical one-way hash function when implemented in a *challenge-response framework*[1]. PUFs reduce the ability of attackers to circumvent security mechanisms, as these mechanisms are implemented in tamper-resistant hardware rather than at the software level. This property of tamper evidence has already been demonstrated for optical PUFs [34] and coating PUFs [1].

Furthermore, the devices are conceptually unclonable in the sense that, although they may be physically copied, this provides no advantage to an attacker, because each copy will behave differently. PUF designs exist that consume very little power, meaning a high degree of security can be applied to embedded applications with extremely limited resources, such as RFID cards.

2.1 Significance of Process Variation

The physical phenomena that underlie a PUF should be computationally difficult to model, and this is no less true for the delay-based PUFs which concern this thesis. While sophisticated models for modeling propagation delay in semiconductor devices exist, much of the

[1] described in Section 4.4

process variation inherent in any manufacturing process can only be modeled as a statistical distribution. These variations exist within a die, between dies, between wafers, and between lots or production batches. These variations appear in the length and width of components such as interconnects and discrete transistor features, as well as the thickness of oxide layers. This variation exists for every property of a silicon device, any of which can have an impact on the PUF's output.

It is well-known that process variation is becoming harder and harder to control as feature size shrinks. It has been shown that, at least between 90nm and 45nm processes, not only is variation increasing but it is also becoming less systematic and more random, or stochastic [33]. These kinds of errors are caused by vibrations during manufacturing or nanometer-scale wafer unevenness–they typically cannot be reduced by improving the process. In [35] ring oscillators are used on a 90nm Field-Programmable Gate Array (FPGA) to estimate the impact of process variation on delay variation. In the study, the amount of variation is projected out to future process nodes. The delay through a lookup table (LUT) was measured to have a mean variation (3σ) of +/-3.5% . The authors projected that for 65nm this will increase to 4.5%, for 45nm 5.5% and 22nm 7.5%. The estimation for 45nm aligns well with the empirical study performed in [33] in 2008, suggesting the projection may be quite accurate.

An interesting consequence is that one might expect that a given delay-based PUF circuit, without any modification, will have increasing reliability at each new technology node–in some sense piggybacking on Moore's Law. Although modern FPGAs are available at various scale nodes, it is unfortunately very difficult to demonstrate unequivocally that a design's performance has improved with scale. FPGA architectures are simply too different across device families–there are many more variables involved than simply diminishing process scale. Thus the idea is provided only as an unproven hypothesis.

2.2 PUF Applications

A few applications for PUFs have been proposed over the years, most with a strong security focus. As more is learned about PUFs, and the phenomenon on which they are based, it is certain that the number of applications will expand. It is important to mention that the suitability

of a PUF for a particular application significantly depends on the application requirements. For example, applications related to secure communications must have very low error rates, while applications for random number generators typically rely on much noisier responses.

2.2.1 Signature Generation

An array of individual PUF structures can be used to generate a unique signature for device authentication. For these kinds of applications, the PUF can be used to identify physical objects in the same way that biometrics can be used to identify people. The signature is tamper-resistant and cannot be duplicated, as it has been generated dynamically from the physical properties of the device in which it is embedded. This used in a variety of intellectual property (IP) protection schemes.

The basic idea for IP protection is that each device's unique ID can be enrolled in a database by the manufacturer before it is released on the market. Then, even if a third party illicitly obtains the full design of the hardware, they will still not be able to produce a device that can be authenticated, since the signature is generated by processes that cannot be precisely controlled during manufacture. This process is described in [36].

There is particular interest in applying PUFs' ability to generate unique signatures to enhancing the security of RFID authentication–just a few examples are [9, 5, 2]. PUFs are also used in the symmetric-key authentication protocol proposed by [12]. Finally, PUFs are applied to authentication of mobile sensor network nodes in [41]. The idea is to prevent an array of attacks which are possible when physical access to a node is possible.

2.2.2 Cryptography

Another application is the generation of secrets for cryptography. The advantage of a PUF is that these secrets do not have to be stored anywhere on the hardware, since they are generated dynamically at device reset. This is especially interesting for embedded devices. An example of a cryptographic application involves a mobile phone whose firmware must be decrypted on each startup. The cryptographic key must somehow be stored securely. Solutions using nonvolatile memory or volatile memory with a battery are vulnerable to physical attacks or side channel

attacks. PUFs can reduce these vulnerabilities, since physically disassembling such a circuit will destroy its delay characteristics and therefore change its output.

As far as secure communication is concerned, there are several RFID (Radio Frequency Identification) authentication schemes proposed that intend to strongly reduce many of the vulnerabilities in today's RFID systems . These designs must be extremely efficient both in energy and complexity since a typical RFID card may only offer a few thousand logic gates. A proposed mutual-authentication scheme for RFID using PUFs appears in [22]. The work in [8] uses a PUF's output to encrypt the challenge-response pairs exchanged during RFID communication.

In [16] SRAM PUFs are used to implement a PKI system to encrypt the transmission of a bitstream to an FPGA. FPGA bitstream encryption is also performed in [15] using Anderson's PUF. Both of these kinds of PUFs are described in Section 3.

2.2.3 Random Number Generator

With some modification a PUF design can also be turned into a true, or cryptographically secure, random number generator. True random number generators have been created by exploiting D-Flip flop metastability [31], Ring Oscillators [39], and SRAM PUFs [38]. Ring Oscillator PUFs and SRAM PUFs are discussed in Section 3.

In a similar way deterministic random bit generators (DRBG) can be created, such as in [38]. DRBGs employ a deterministic algorithm to create pseudo-random numbers, but seed it with the random signature generated by a PUF. As long as the seed remains secret, the numbers that are generated are not predictable. This system can create large numbers of random numbers very quickly.

2.3 PUFs and FPGAs

In the PUF literature, actual device implementation and testing is either done on an ASIC (Application-Specific Integrated Circuit) or an FPGA (Field-Programmable Gate Array). FPGAs are invaluable to PUF research for several reasons, not least of which being related to FPGA reconfigurability.

- Experiments can be performed rapidly with the whole testbench loaded onto the same hardware as the PUFs.
- The effects of small changes to routing and component placement can be verified quickly.
- A host of dedicated hardware components are already on-chip for testing, usually with nominal delay characteristics provided by the manufacturer. These can usually be instanciated in an HDL with little effort. Xilinx has documentation on all hardware elements that can be instantiated, such as [40] for the Spartan 3E. This document was used extensively for this project.

On the other hand, FPGA PUFs are more constrained than their ASIC counterparts.

- Routing is constraint to the resources provided on-chip. A good amount of control over routing is possible using design constraints, but arbitrary routes cannot be generated. When routing is tightly controlled, such as with the application of hard macros (used by many PUF designs, including the one in this thesis), the automatically generated routes may be significantly less optimal since they are blocked by the hard macro. In the worst case large arrays of hard macros can cause the overall circuit to become unroutable.
- Placement of circuitry on-chip is constrained to discrete locations. What is more, usable locations may not be distributed regularly across a die.
- Some dedicated hardware resources may simply be unavailable for experimentation. This is especially the case for analog circuitry such as amplifiers or analog-to-digital converters.
- It is often difficult, if not impossible, to transfer a design between FPGA technologies due to the large variation in available resources, coupled with the degree of low-level control that is required. An example of such a resource variation is the number of inputs per LUT.

Conceptually, FPGAs are primarily useful as vehicles for studying PUFs cheaply and at a large scale. There are negative security implications for an FPGA-based PUF, since simply relocating a PUF on a die can change its outcome. In fact, by trial and error one can induce any

signature one desires. Furthermore it would not be difficult for a malicious party to add side-channels to a design to capture and transmit the PUF signature. From a practical standpoint and ASIC can be used to optimize an FPGA design. For these reasons, and reasons related to design optimization, FPGA-based PUFs are more interesting for their potential for migration to dedicated FPGA hardware, or migration to ASIC hardware.

CHAPTER 3. LITERARY SURVEY

A variety of PUF designs have appeared over the past decade. In fact, in [29] it is noted that a new PUF design has appeared roughly each year since 2000. A literary survey was performed for this thesis to obtain a good idea of the state-of-the-art in PUF design. Several of the more interesting designs are discussed briefly in this section. A more thorough survey that includes some non-delay-type PUFs appears in [27].

3.1 SRAM PUF

An SRAM PUF is a kind of memory-based PUF. Memory-based PUFs exploits the unpredictability of that the startup value of volatile memory cells, which is caused by slight asymmetries in the cell's internal routing and transistor characteristics. SRAM PUFs are quite interesting in that they rely on commodity SRAM cells. In fact, after the PUF signature is extracted it is possible to use the same cells as regular non-volatile memory. As an example, SRAM PUFs have even been evaluated on a commodity microcontroller [6]. In that work a set of criteria and metrics are proposed to determine whether an given SRAM can function as a PUF.

In their raw, uncorrected state, this type of PUF suffers from a relatively high error rate. Instability occurs when the internal cell layout is too symmetrical–it becomes susceptible to environmental noise, temperature changes, and power supply transients. One proposed approach to combat unstable bits is to place more PUFs than needed, and add ADC circuitry to automatically select the most stable ones [18]. Unfortunately, this approach is not practical for FPGA-based studies since there is generally no flexible way to measure the analog aspect of an internal signal. Another technique applies *helper data algorithms* to normalize the output [16].

3.2 Butterfly PUF

Conceptually, the Butterfly PUF is somewhat similar to the SRAM PUF, in that they both are memory cells whose startup value is hard to predict. However, it happens that FPGA SRAM cells are all reset to a known state upon device reset. Therefore the Butterfly PUF was developed in [23] as a way to enable the study of memory-type PUFs on an FPGA. It exploits cross-coupled D Flip-Flops, shown in Figure 3.1.

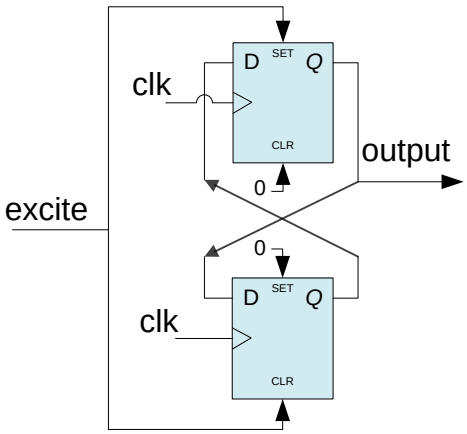


Figure 3.1: Butterfly PUF

Initially the "excite" signal is raised high for a few clocks. Since the preset and clear pins on the D Flip-Flops are asserted, and due to the cross-coupling of the outputs, the circuit is held in an indeterminate, unstable state. When "excite" is released, the circuit output will resolve itself as either '1' or '0' based on the delay mismatch between the interconnects. In the ideal case, in which the routes are totally symmetrical, this outcome is caused by the effect that process variation has on the delay. The advantages of this design are that it uses only D- Flip flops which are ubiquitous in FPGAs as well as in general design processes. One disadvantage is that it requires extra care to route, due to the constraints of FPGA routing.

3.3 Ring Oscillator PUF

Generally PUFs based around design symmetry have been deemed less suited for implementation on FPGAs due to the limitations of routing [32]. This is one of the reasons for the

popularity of RO-based designs on FPGAs, since absolute symmetry is not necessary to create an oscillator, and the error associated with making a single measurement is amortized across many oscillator cycles. The ring oscillator (RO) is one of the earliest and mature classes of delay-based silicon PUFs, first introduced in [13, 14]. A ring oscillator simply a loop of inverters having an odd number of stages. The circuit will spontaneously begin to oscillate with a frequency that can be determined from the delay of each inverter stage. A typical 5-stage RO including an enable signal is shown in Figure 3.2.

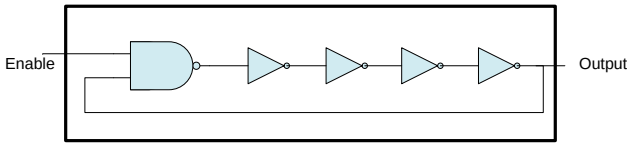


Figure 3.2: Ring Oscillator With Enable

The RO PUF relies on the fact that any two rings will not oscillate at the exact same frequency, even if they are laid out exactly the same. This is due to process variation which impacts the delay of the signal propagating around the ring. A more recent RO PUF variant affixes a counter to each RO, and compare the counts after a period of time, in pair-wise fashion [37]. This "differential" measurement has been shown to give better results than the basic RO design. A typical RO PUF with such a configuration is shown in Figure 3.3. The example produces a single response bit.

In [28] is performed the largest-scale analysis of RO behavior that is known to date, using 90nm FPGAs as test platforms. The study confirmed that the RO PUFs generated signatures that were unique among different chips, and quite consistent within a given chip.

3.4 Arbiter PUF

The arbiter PUF is another well-studied design, published in 2004 [24]. In the general sense, an arbiter PUF sets up a set of closely-matched race tracks with an arbiter at the end to determine which signal reached the end first–typically this is a D Flip-Flop with one signal attached to the clock pin and another attached to the data pin. The basic arbiter PUF design

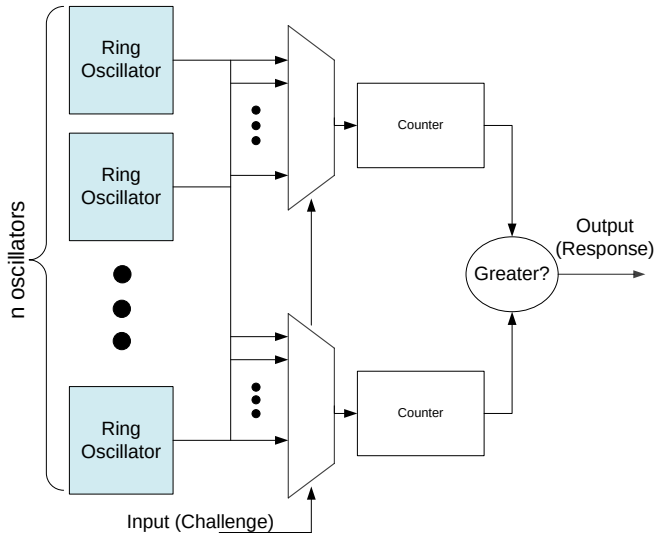


Figure 3.3: Ring Oscillator PUF

is shown in Figure 3.4.

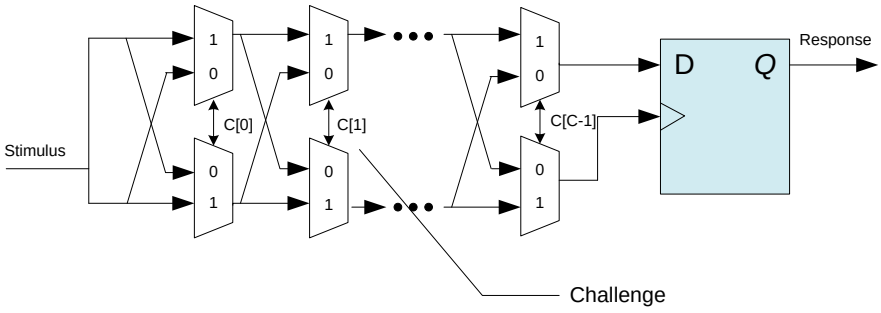


Figure 3.4: Arbiter PUF

Although shown as multiplexers, the adjustable delay portion of the circuit is implemented in different ways. In [30] LUTs are used to create extremely precise programmable delay lines.

A rigorous large-scale analysis of this kind of PUF is performed by [19]. In that work, it is demonstrated that is quite feasible to make a fully-functional arbiter PUF on an FPGA, despite the routing constraints. Interestingly these results fall contrary to the results of [32] which used timing tools to conclude that FPGA routes could not be configured which are matched closely enough. This discrepancy demonstrates the challenge of measuring process variation and the importance of empirical study.

While arbiter PUFs have been shown quite good in terms of adhering to PUF properties,

it has been shown that the basic form is vulnerable to model-building attacks [10]. Using machine learning, after observing a sufficient number of sufficient challenge-response pairs, it was possible to guess the outcome the PUF with 0.6% error rate. Subsequent designs add additional complexity in order for the challenge to control the delays in a non-linear way. An early attempt to introduce non-linearity is the feat-forward arbiter PUF [10]. Since then, there have been several rounds of attack proposal followed by design modification.

3.5 Anderson PUF

The Anderson PUF [3] is quite unique in that it is the first PUF designed expressly for implementation on FPGAs. Unlike many PUFs designed for FPGAs, it does not require hard macros to control symmetry. Instead it uses the carry chain multiplexers present in certain FPGA components. A simplified depiction of Anderson's PUF is shown in Figure 3.5. Both LUTs A and B are configured as shift registers, and initialized with bit strings that are inverses of each other. Thus when clocked, the two LUTs output square waves that are 180 degrees out of phase. The inputs to the LUTs are connected as necessary to perpetuate the output pattern.

Due to process variation in the LUTs and the multiplexers that they control, the propagation delay from the input to the output will vary from LUT to LUT. For some instantiations, the LUT outputs will be sufficiently out of phase to produce a brief rising glitch at the output, which can be captured by a flip-flop. The presence or absence of the glitch determines the PUFs output bit.

3.6 Commentary

From a classification standpoint, the PUF designed for this thesis (described in Chapter 4) is a memory-type PUF, and therefore inherits similar properties. Compared to the designs discussed thus far, the one proposed in this thesis:

- similar to the Butterfly PUF it uses general FPGA reconfigurable fabric, making testing more convenient than for SRAM PUFs. The circuit can be reset arbitrarily without

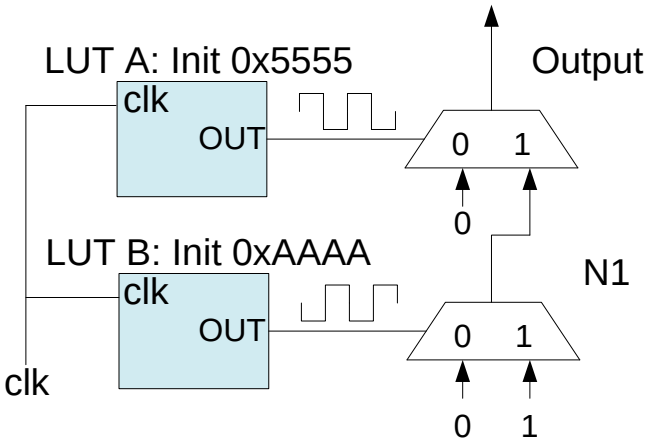


Figure 3.5: Anderson's PUF

requiring the whole device to be reset. Furthermore the cells can be placed around the IC arbitrarily which can make reverse engineering more difficult.

- is conceptually less complex than the non-linear Arbiter PUFs. Despite that, this kind of circuit is (thus far) much less vulnerable to modeling attacks, since each cell behaves independently of the others.

- intuitively has a lower power requirement than the RO PUF, since it does not oscillate.

On the other hand, a disadvantage of this kind of PUF is that the space that an n-bit response requires at least n cells. If a large number of cells are needed, for example if extensive error correction is applied, the area consumed could become fairly large. Additionally in their raw form this kind of PUF tends to suffer from a slightly higher-than-average level of noisiness.

CHAPTER 4. DESIGN AND IMPLEMENTATION

4.1 Design Goals

The broad goals of this design were threefold:

1. Minimize PUF area.
2. Minimize dynamic power consumption.
3. Minimize the evaluation time.

Ignoring error-correction, a memory-type PUF needs n cells to create an n-bit signature. To accommodate longer signatures, it is desirable for each cell to be relatively small to improve spacial efficiency. It is also important to recall that the PUF itself is only a small component of a larger application. Most PUFs do require some kind of post-processing to correct for irregularities in the PUF output. Both overall size and evaluation time are affected by the complexity of the post-processing phase. Therefore it is also desirable for the PUF to adhere as closely as possible to the the ideal PUF properties before any post-processing is applied. Then the cost of post-processing can be minimized. Power consumption is a particularly an important factor for resource constrained embedded applications such as RFID or sensor networks.

4.2 Principle of Operation

As described in Chapter 3, memory-type PUF uses a small cell whose contents cannot be are not known until it has stabilized after reset. For example, in Figure 4.1, if both switches are initially closed, then the capacitors at Q1 and Q2 are both to charged to logical 1. Then at at t=0, the switches are opened and the circuit must resolve to a stable state which depends on

the propagation delay through the inverter, the delay of the interconnects, and the switching threshold of the logic. If Route R1 has a shorter delay, it will remain at logical 1 and force Q2 to logical 0. If Route R2 has a shorter delay, it will remain at logical 1 and force Q1 to logical 0. Thus Q1 and Q2 will always resolve to opposing values, but which has which value depends on the physical properties of the device into which it is instantiated.

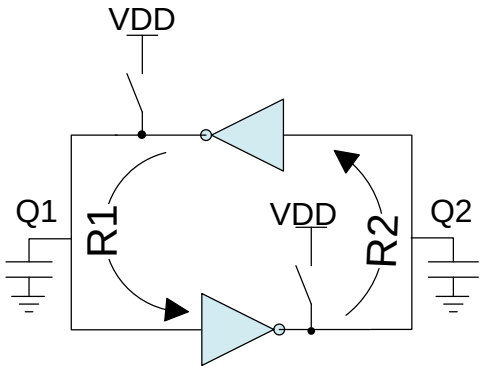


Figure 4.1: Conceptual design

The PUF designed for this thesis is shown in Figure 4.2. The switches are replaced by combinational logic which is implemented in LUTs. Thus the multiplexer and inverter are only displayed to demonstrate the LUT configuration. The D flip-flop is included because it allows the internal reset lines to be routed locally. The local routing means the routes can be conveniently included in the PUF hard macro so that reset skew will not be a variable across instances.

4.3 Implementation Details

Although the design is quite simple, given the routing constraints of an FPGA, ensuring symmetrical routing is still a challenge.Another problem is ensuring that each instance of the circuit is identical. Neither of these requirements can be met by allowing the design tools to place and route on their own.

The Xilinx toolset includes a tool called FPGA Editor that not only allows the physical

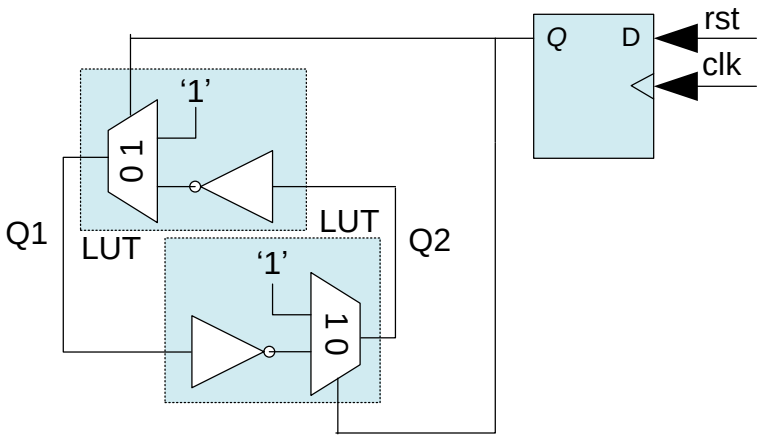


Figure 4.2: Logical design

placement and routing of a design to be verified, but also allows creation of what are known as *hard macros*. Figure 4.3 shows a single PUF instance implemented as a hard macro on a Spartan 3. Figure 4.4 shows a regular grid of such instances. There are no external routes shown, such as the reset lines or the outputs, only the two local loops emerging from the SLICE on the right, entering the switch matrix on the left, and then returning to the SLICE. These two routes correspond to the nets Q1 and Q2 in Figure 4.2. The results in [35] suggest that even the positions of CLB pins for signals not on the critical path should be consistent, as their placement can have a small effect on the overall circuit delay. The combination of mapping constraints and hard macros maintains pin placement across instances.

Conceptually the circuit delay can be broken down into two main parts. There is the fixed delay of the design, and the uncertainty in delay due to the process variation. The concept is portrayed in Figure 4.5.

The bias introduced by allowing the design tools to perform all routing is exemplified in Figure 4.5a. The delay of the two critical routes, R1 and R2 are compared. The fixed design delay is shown as the empty region, and the deviation due to process variation is shown as the shaded region. It does not matter how much variation is there is, as the routing mismatch is too great and therefore its effect dominates. This circuit will produce the same output regardless of placement and is not suitable for most PUF applications.

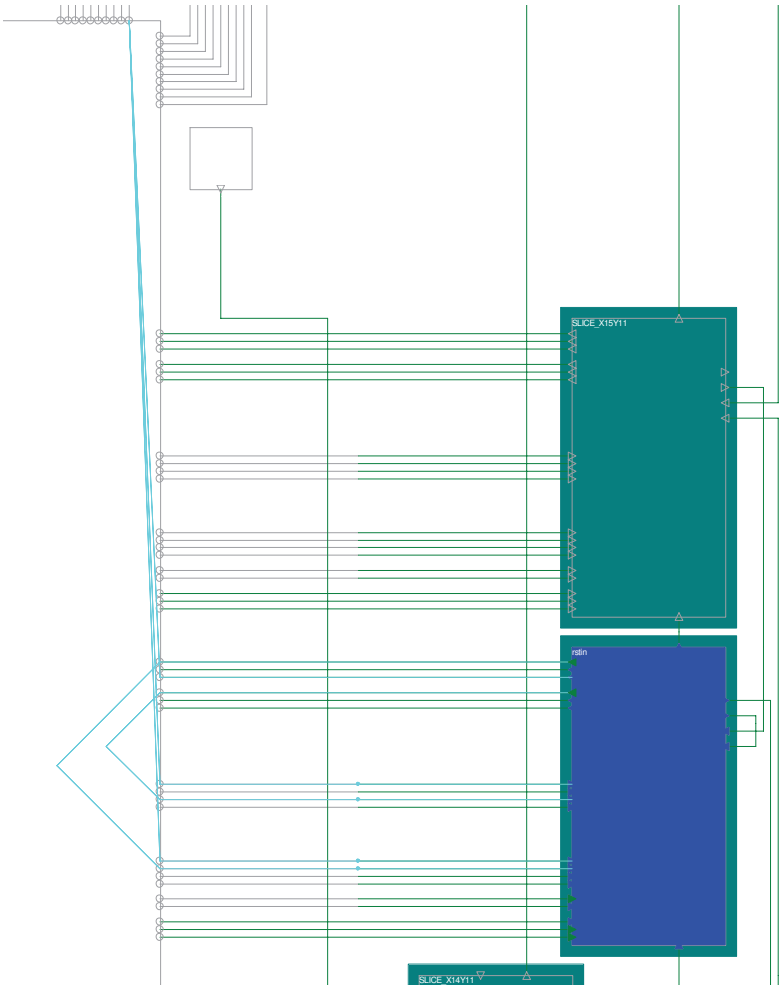


Figure 4.3: PUF layout in FPGA Editor

The use of hard macros to constrain routing of the critical signals is exemplified in Figure 4.5b. The delay bias is caused by the uncertainty of process variation. The greater the process variation, the better chance that the relative delay will be strongly biased, and therefore always produce '0' or '1'.

Finally, the comparison in Figure 4.5c also has closely-matched route delays–however, the delay variation is quite small and therefore the circuit may be unstable in the face of varying environmental conditions such as thermal noise, supply voltage as well as the switching activity of nearby circuits. A PUF whose routes happen to be too closely matched may show random inconsistencies in its output. Thus greater variation is desirable as it reduces the possibility that these effects can temporarily bias the circuit one way or the other.

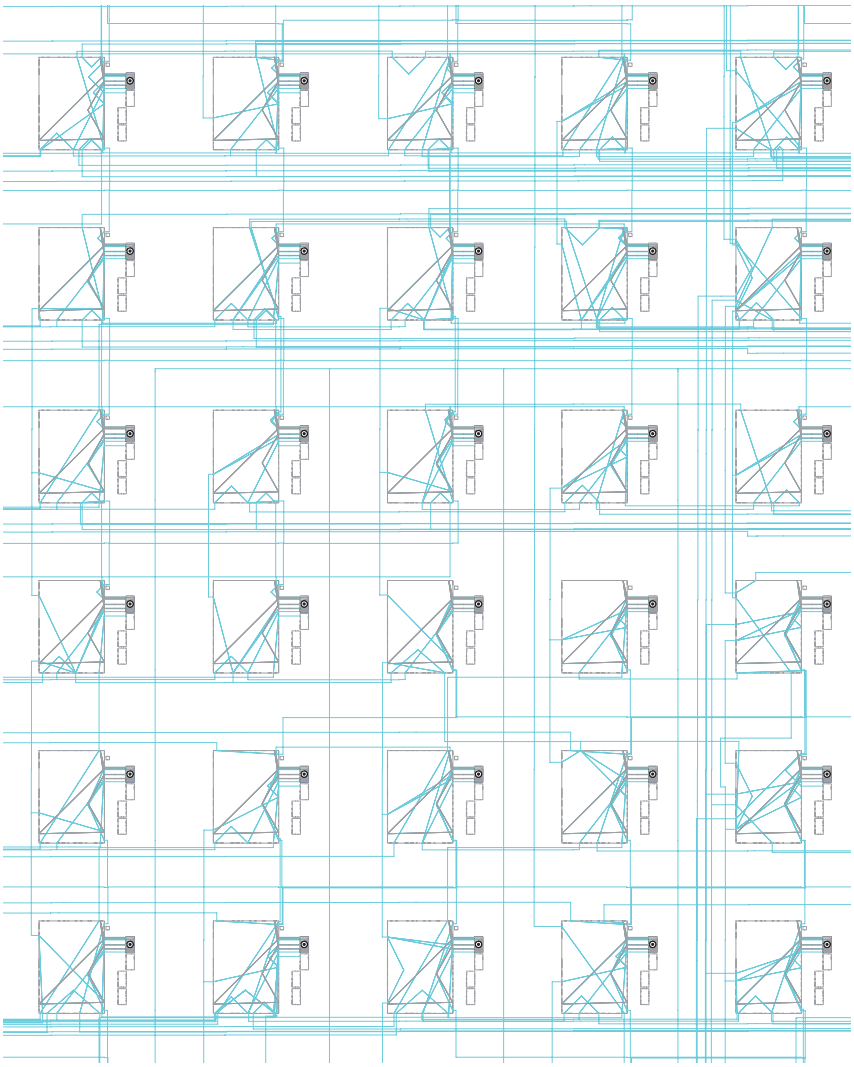
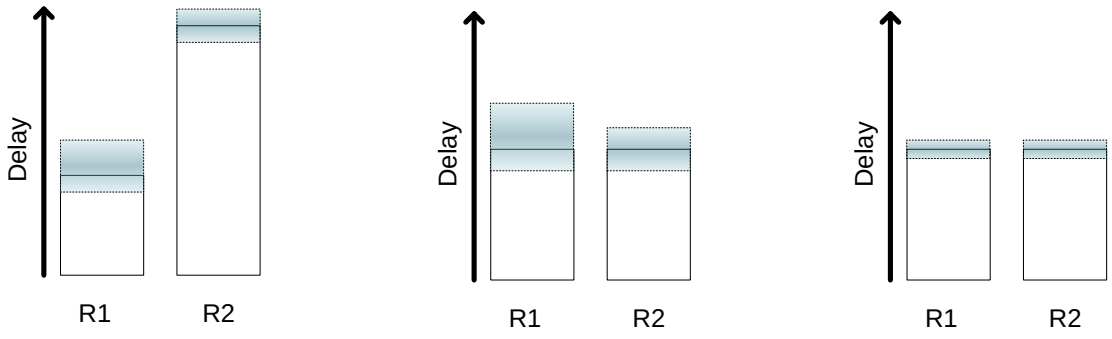


Figure 4.4: Portion of 128 cell PUF Array

The function of the PUF can be described a bit more formally. The following equation shows the delay of a net d_N. d_S represents the static delay that is estimated by the design tool. On the other hand, d_R is a random variable representing the uncertainty in net delay due to process variation. Finally d_{NOISE} is a dynamic random variable representing the effects of temperature and voltage variation as well as interaction between circuits. Both d_R and d_{NOISE} may be either negative or positive quantities.

$$d_N = d_S + d_R + d_{NOISE} \tag{4.1}$$

Next, we can model the characterize the delay of the two nets, Q1 and Q2. Both d_{L1}



(a) The deviation in delay due to process variation is too small compared to the routing mismatch, causing the output to be fixed for all instances.

(b) The routes are closely matched at design time, and the deviation in delay due to process variation causes a bias that differs for each instance.

(c) The routes are closely matched at design time, but the process variation is small (or highly correlated), so that noise effects dominate.

Figure 4.5: Delay Model

and d_{L2} are additional random variables representing the delay through the two LUTs that are employed by the design to create the necessary logic. They are always positive. As an example, the Spartan3E datasheet lists a maximum propagation delay of around 0.66ns to-0.76ns depending on speed grade.

$$d_{q1} = d_{L1} + (d_{Sq1} + d_{Rq1} + d_{NOISEq1}) \tag{4.2}$$

$$d_{q2} = d_{L2} + (d_{Sq2} + d_{Sq2} + d_{NOISEq2}) \tag{4.3}$$

Ideally the quantity $\Delta d_S = (d_{Sq1} - d_{Sq2})$ should be zero so that the effect of the random LUT and route delay components dominate. The difference between the delay of the routes, Δd, dictates the circuit outcome for a given evaluation.

$$\Delta d = (d_{L1} - d_{L2}) + (d_{Sq1} - d_{Sq1}) + (d_{Rq1} - d_{Rq2}) + (d_{NOISEq1} - d_{NOISEq2}) = \Delta d_L + \Delta d_S + \Delta d_R + \Delta d_{NOISE} \tag{4.4}$$

Finally, the PUF can be described by a pair of piecewise functions depending on Δd. This

equation is validated in Section 5.2.

$$Q1 = \begin{cases} 1 & : \Delta d < 0 \\ 0 & : \Delta d > 0 \end{cases} \tag{4.5a}$$

$$Q2 = \begin{cases} 1 & : \Delta d > 0 \\ 0 & : \Delta d < 0 \end{cases} \tag{4.5b}$$

Notice that these functions are not completely defined. If Δd_R is zero, or close to it, Q1 and Q2 essentially become random variables due to the dynamic Δd_{NOISE} term. This is results in metastability, stressing the importance of minimizing the static route skew.

A few desirable properties can be deduced from this analysis.

1. Δd_S should be as close to 0 as possible.

2. d_{NOISE}should be as close to 0 as possible.

3. $d_R \gg d_S$ and $d_R \gg d_{NOISE}$ to maximize reliability

4.4 Challenge-Response Framework

At the application level, PUFs are quite often integrated into a *challenge-response framework*, and it is what makes them useful in authentication applications. This is primarily why PUFs are referred to as functions. A PUF challenge is a bit string that is offered as input, which the PUF's output depends on–broadly speaking it behaves quite like a hashing function.

The PUF design in this section represents a single bit generator. It can be arranged as an array to produce an arbitrary number of bits. In order to adapt such an array into challenge-response system, one of the easiest modifications is to add comparison. A commonly proposed circuit is a set of wide multiplexers which perform a pairwise comparison of the generated signature bits, based upon the challenge.

This thesis focuses on the PUF design itself; such a higher-level construct is left to future work.

CHAPTER 5. EXPERIMENTATION AND EVALUATION

5.1 PUF Properties

In recent years there have been greater efforts to standardize the terminology used to describe PUF properties and the metrics used to evaluate them. An example is [29] which attempts to classify and summarize the terminology and evaluation techniques used by [19] and others. This paper attempts to use such terminology where applicable to better facilitate comparisons between designs. It should be noted, however, that there is still not full consensus on what the full list of properties should look like. The four used in this thesis appear below.

- Reliability: the output of the PUF should be consistent.
- Uniformity: there should be an equal distribution of 1's and 0's in the output. This is also called randomness in [19].
- Uniqueness: For two instances of the PUF structure, the responses to the same challenge should be substantially different. This is necessary for the creation of many unique keys or signatures.

Many other properties are defined by other authors. Based on the fact that this PUF is a memory-type PUF, the selected property, borrowed from [6], is shown:

- Intra-die correlation: the bits of the response vector should be uncorrelated.

These properties are validated in the following sections.

Configuration	Q1 Delay (ns)	Q2 Delay (ns)	Skew (ns)	Uniformity (%)	HDINTRA (%)
a1b1	0.694	0.414	0.28	100	0
a1b2	0.444	0.373	0.071	18.8	11.4
a1b3	0.463	0.02	0.443	100	0
a2b1	0.066	0.414	-0.348	0	0
a2b2	0.066	0.373	-0.307	0	0
a2b3	0.066	0.02	0.046	17.6	11.7
a3b1	0.329	0.414	-0.085	0.1	0.1
a3b2	0.329	0.373	-0.044	0.1	0.1
a3b3	0.329	0.02	0.309	100	0

Table 5.1: Effect of Routing Skew on HDINTRA and Uniformity

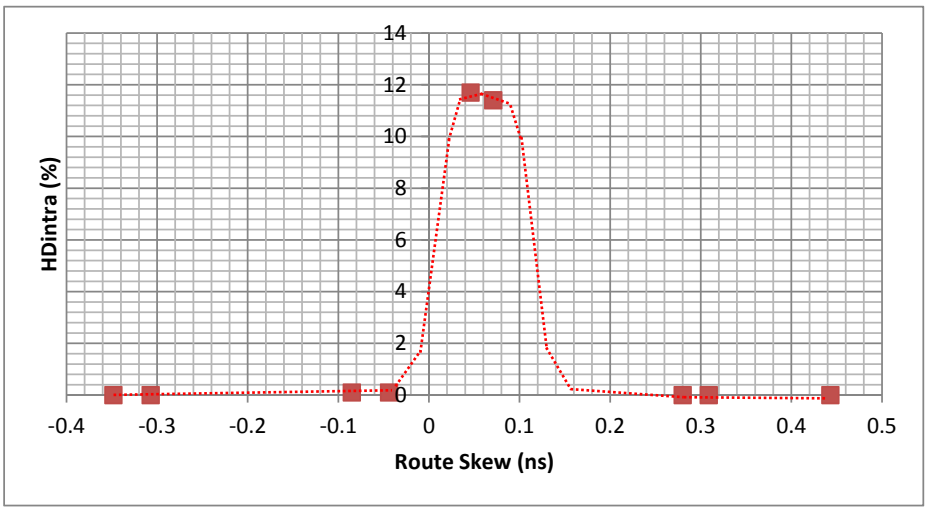
5.2 Effect of Routing Skew

An experiment was performed on a Spartan3E to attempt to confirm the equations described in Section 4.3, particularly Equation 4.7. The same PUF design was used as described in Section 4, but the lengths of the routes Q1 and Q2 were adjusted. This was done by changing the LUT pins that were used at the source code level, portrayed graphically in Figure 5.2. Since the pins of the reset lines were kept fixed, and each LUT has 4 inputs, a total of 9 routing configurations were left. The thus-configured PUFs were read 100 times each to obtain each data point. There results are shown in Figure 5.1 and Table 5.1.

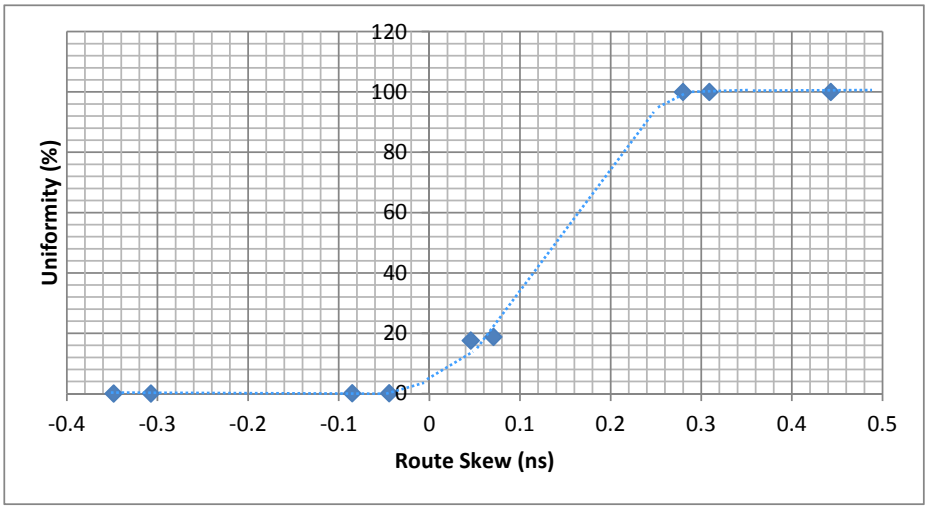
The skew values shown were calculated from the static delay analysis tool, and the values for HD_{INTRA} and Uniformity in Figure 5.1 were gathered empirically.

The apparent systematic offset from 0ns in Figure 5.1 may exist for two reasons. First, there could be a systematic error in the static timing analysis (ie, wrong speed grade selected). Second, as reported by the timing tool, there does exist a small fixed skew on the reset lines for each configuration, which will create a small bias in the response.

The data that was gathered appears to confirm the assumptions made in the delay model described in Section 4. It is evident from the data that HD_{INTRA} and Uniformity are coupled– it is not particularly easy to optimize both. For example, looking at Table 5.1, configuration "a1b1" achieves an error rate of 0%, but at the cost of a total bias towards generating 1's. Configuration "a2b2" achieves a0% error rate with a total bias towards generating 0's. Configuration "a2b2" was used in subsequent sections. It may not be the optimal configuration,

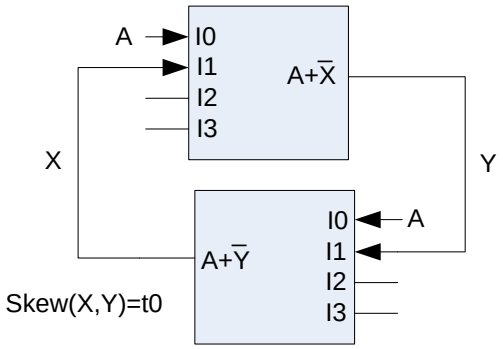


(a) Although the amount of available data is quite small, HDINTRA (ie, the error rate) appears to form a bell curve when compared to skew. The highest error rate is seen when the routes are nearly perfectly matched (with some systematic offset), which follows from the assumption that noise effects tend to dominate under such conditions. Outside of this region the error rate is zero since the response is fixed.

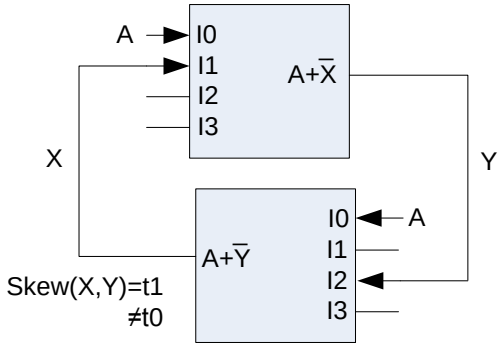


(b) As the skew increases, the Uniformity saturates at 100% (1's dominate) or 0% (0's dominate).

Figure 5.1: Effect of Routing Skew on HDINTRA and Uniformity



(a) As described in Section 4, two LUTs are used to create a combinational loop, both implementing the same function. The skew between X and Y is simply the difference of the delays.



(b) The two LUTs are still implementing the same function, but one LUT has a different pin selection. As a result of FPGA routing constraints, the skew (and thus circuit behavior) may be radically different from (a), even though the logic is the same.

Figure 5.2: Effect of LUT Input Selection on Route Skew

but an exhaustive search through the full set of pin permutations was not possible due to time constraints. Future work might attempt to automate this process of finding the optimal solution.

This experiment demonstrates the degree of impact that tiny changes in routing can have on PUF behavior, and the small window of allowable routing skew for which the circuit will function as a PUF.

5.3 Error Correction

Similar to biometrics, PUFs typically require some form of error correction to increase reliability. Various error-correction schemes have been proposed to stabilize the output of PUFs. A very simple scheme that requires minimal hardware is the majority vote. Two forms are known as temporal majority voting, and spatial majority voting [26]. These techniques are used to increase the reliability and uniformity of our PUF's output.

Temporal majority voting (TMV) is also sometimes referred to as the *repetition code*. It involves making an odd-numbered N_T readings of the PUF, and then determining how many 1's were read. If more than $M_T = \frac{N_T - 1}{2}$ 1's are read, the output is considered a 1. In some sense this acts as a low-pass filter that is especially useful for PUF bits that occasionally toggle. Thus, it improves the reliability, but at the expense of the process taking N_T times longer.

Spatial majority voting (SMV) involves logically grouping small sets of N_S PUFs for the purpose of generating a single bit. Each PUF in a group produces a bit in parallel, and if the number of 1's produced exceeds M_S then the overall group is considered to have produced a 1. This form of majority voting helps to move the distribution of 1's and 0's in a string to uniformity. Of course, like TMV, there is a trade off in that N_S times as many cells are required.

The post-processing that is applied in the following sections below first applies TMV with $N_T = 3$ and $M_T = 1$, and then applies SMV with $N_S = 2$ and $M_S = 0$. These values were determined experimentally to show an improvement in both Repeatability and Uniformity, at the cost of requiring 3 times as many readings, and 2 times as many cells, as the raw PUF circuit.

Another method of error correction which can correct noisy and non-uniform key data uses a so-called *fuzzy extractor* as described in [11] and applied to PUF technology in [26] . This method, however, is rather more complex and an examination of its effect on this particular circuit is left to future work.

5.4 Suitability as a PUF

A circuit needs to be thoroughly tested before it can be deemed suitable as a PUF. Although these devices can be simulated to a certain extent, due to the increasingly unpredictable properties of process variation, on-chip testing is more accurate.

Each of three Spartan3E FPGAs was divided into 8 regions in which the PUF is tested. The regions are shown in Figure 5.3. It has been shown that interchip variation is at least as great as intrachip variation[24], so the lack of distinction between the two in the analysis that follows is acceptable.

In the discussion that follows, HD stands for Hamming Distance, which is simply the total number of bits that differ between two bit vectors. Hamming Distance is a very common metric applied in PUF evaluation.

5.4.1 Reliability

The ideal PUF should exhibit perfectly consistent, or reliable, behavior for a given instance. In other words, it should always output the same value under any operating condition. The extent to which a PUF deviates from this property can be called its error rate. A simple way to express the error rate, as defined by [6], calculates the average intra-chip Hamming Distance between a series of samples, for a particular PUF instance i. A baseline n-bit response R_i is extracted from the circuit, and compared to m further samples. The expression to obtain a single value based on a set of intra-chip HD[1] calculations is shown below.

$$HD_{INTRA} = \frac{1}{m}\sum_{t=1}^{m}\frac{HD(R_i, R'_{i,t})}{n} \times 100\% \tag{5.1}$$

[1]This value is sometimes referred to as μ_{intra}

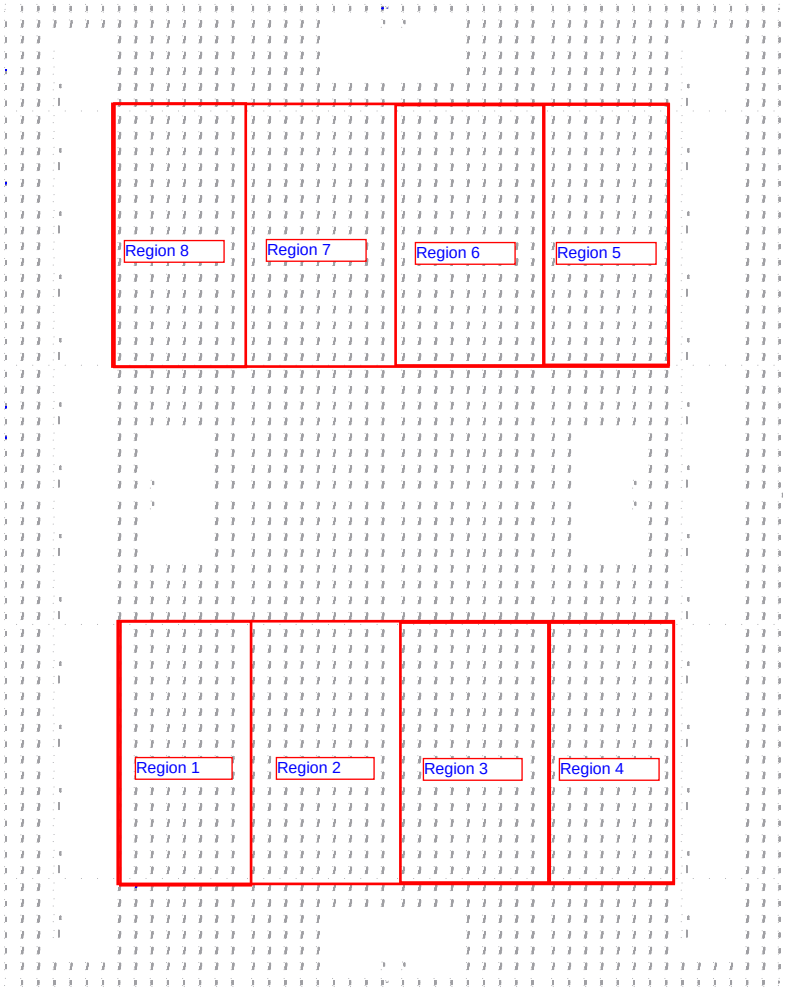


Figure 5.3: Placement Regions Defined on Spartan3E

For this experiment, m, the number of repetitions, is 100. Table 5.2 shows the experiment results. In the ideal case this value is 0%. The raw results show a relatively high error rate, which is typical for SRAM PUFs [6]. With the application of lightweight post-processing the error rate is improved.

5.4.2 Uniformity

Another metric is the uniformity of the PUF response. It is so-named because there should be a uniform probability distribution of '0's and '1's in a given response r for circuit instance i. It is effectively the mean value of an n-bit response, and can be expressed as follows.

Instance	$HD_{INTRA}Raw$	HD_{INTRA} Post-processed
1.1	12.2	4.4
1.2	11.3	4.2
1.3	15	5.2
1.4	13.2	8.7
1.5	15	5.3
1.6	13.4	8.9
1.7	14	5.9
1.8	13.9	4.7
2.1	12.7	6.2
2.2	11.3	10.4
2.3	14.9	8.3
2.4	11.9	1.7
2.5	14.2	2.5
2.6	13.5	5
2.7	13.6	8.9
2.8	11.7	5.3
3.1	6.9	7.3
3.2	10.7	8.6
3.3	15.2	8.8
3.4	9.3	9.2
3.5	10.6	8
3.6	14.2	7.1
3.7	16.3	12.7
3.8	14	3.2
Average	12.9	6.8

Table 5.2: Design Reliability

$$U_i = \frac{1}{n}\sum_{l=1}^{n} r_{i,l} \times 100\% \tag{5.2}$$

An ideal PUF would show a uniformity of 50%. Since the PUF error rates are non-zero, the Uniformity is averaged over 100 responses. The results of this experiment are shown in Table 5.3. For the raw circuit, the Uniformity is quite low suggesting a strong preference towards producing 0's. After post-processing, the output is closer to uniformity, with a small bias towards producing 1's.

Instance	U_i	U_i Post-processed
1.1	20.4	59.5
1.2	16.2	48.5
1.3	23.8	63.5
1.4	21.9	56.9
1.5	26.6	67.4
1.6	24.2	58.1
1.7	22.3	60.6
1.8	23.6	66.5
2.1	23	57.8
2.2	19.9	52.7
2.3	25.2	62.7
2.4	26.3	61
2.5	24.7	60
2.6	23.9	59
2.7	26.3	60
2.8	24.3	59
3.1	9.4	31.8
3.2	17.6	44.7
3.3	15.1	46
3.4	13.9	39
3.5	15.5	43
3.6	24.6	59
3.7	22.4	60
3.8	23.7	66
Average	21.5	55.8

Table 5.3: Design Uniformity

5.4.3 Uniqueness

The uniqueness property of a PUF is the correlation between chips. A PUF duplicated on another chip should produce a signature with a Hamming Distance of around 50%, which means half the bits are different.

The following equation can be applied to determine the uniqueness of a PUF across a population of k chips using pairwise calculations of HD, called HD_{inter}[2].

$$Uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i, R_j)}{n} \times 100\% \tag{5.3}$$

[2]This value is sometimes referred to as μ_{inter}

For this experiment, first pairwise comparisons are performed to determine the HD between all circuit instances. Table 5.4 compares the Raw and Post-processed Uniqueness values for the entire population. It is evident that reducing the error rate significantly improved the Uniqueness of the PUF response.

Raw	Postproccessed
34.2	47.8

Table 5.4: Design Uniqueness

In Figure 5.4 is shown a comparison of the histogram of HD_{INTER} before and after post-processing. It is clear that the histogram has been shifted closer to the ideal 50% average HD.

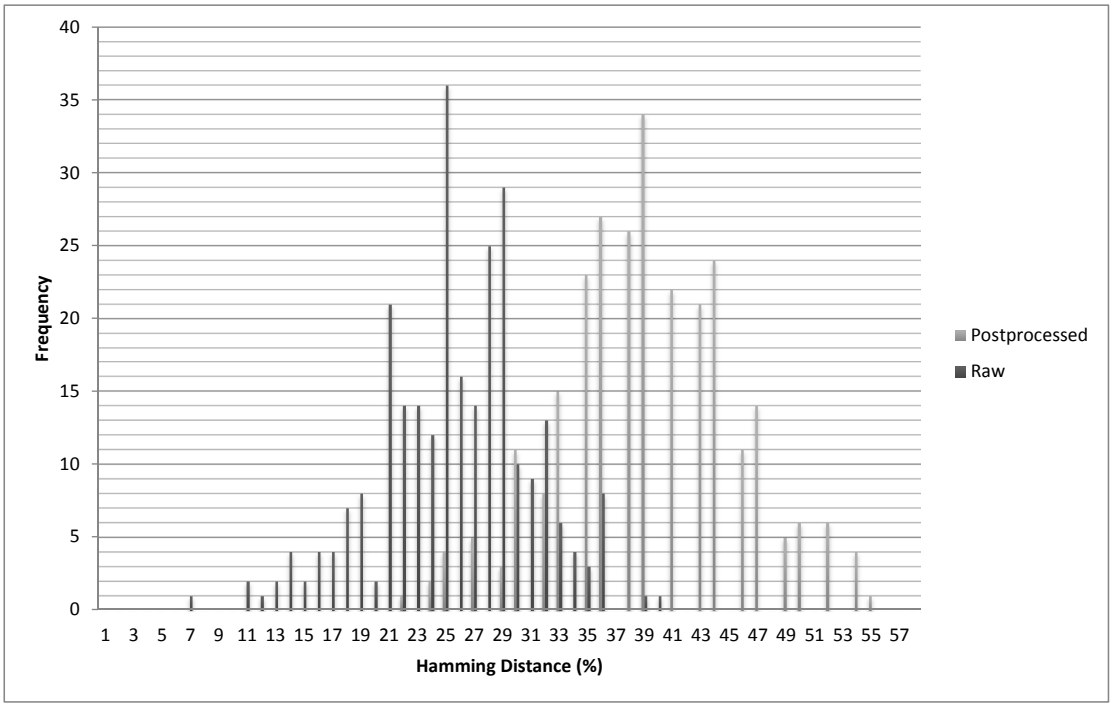


Figure 5.4: Raw HDInter vs Postprocessed HDInter

In Figure 5.5, the histograms of HD_{INTER} and HD_{INTRA} are directly compared. The fact that they do not overlap is perhaps the most significant result of this thesis–it indicates that with in this population of 24 PUFs it is possible to distinguish between a given PUF's noisy response, and the response of other PUFs. This means it is possible to implement a detection algorithm to identify a given device. We can also estimate the minimum number of unique IDs

that could be generated based on the test results. The lowest HD_{INTER} that was observed was 22%, suggesting that in such a case around 28 bits were different between the two signatures. The number of IDs whose HD is 28 from a reference ID is *128 choose 28*, or around 1.3×10^{28}.

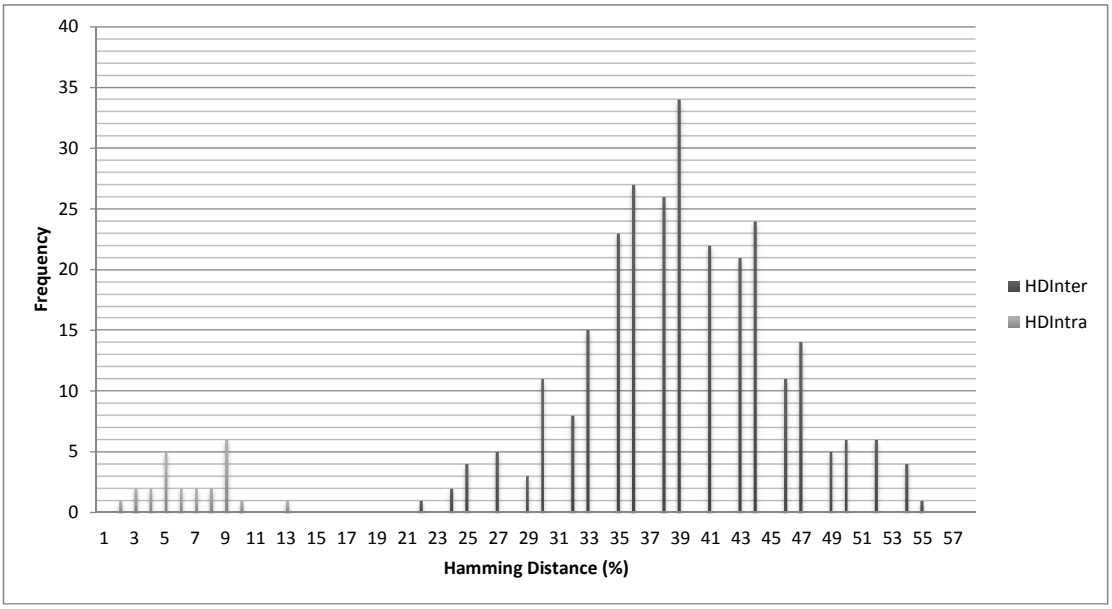


Figure 5.5: HDInter vs HDIntra

5.4.4 Correlation Between Bits

The autocorrelation test presented by [6] can be used to detect correlation between bits. If there is any systematic or stochastic element to the process variation, it may show up as a significant correlation at particular intervals. Because the signature bits are extracted from a common fabric it is possible for spatial correlation to appear due to gradients–the existence of which is demonstrated in [35]. Other factors that might cause correlation include the physical layout of the individual cells, or interaction between circuit elements

The autocorrelation equation used to measure this is shown below.

$$R_{xx}(j) = \sum_{t=1}^{n} x_t x_{t-j} \tag{5.4}$$

For this experiment, as with the others, the relative configuration of the cells is maintained across all tests. A signature sample was taken from each device, and the autocorrelation test

was performed. The 0's in each signature were replaced by -1, so that a correlation value of 0 indicates no correlation, and the values 1 and -1 indicate total correlation. Figure 5.6 shows the experiment results.

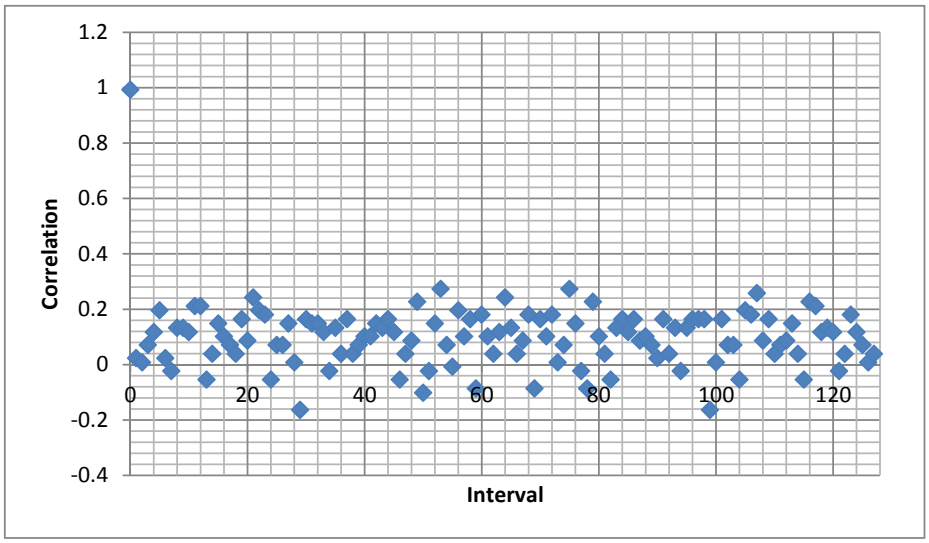
As expected, the autocorrelation for interval length 0 is 1 for each device, since each bit is totally correlated to itself. Although there are a few outliers, the results show no strong correlation patterns between the three devices, suggesting that the physical configuration of the cells does not have a strong impact on the resulting signature. This may appear to contradict the results in [6], but it must be recalled that the array used in that experiment is much larger, so the effect of systematic process variation must be more pronounced.

5.5 Environmental effects

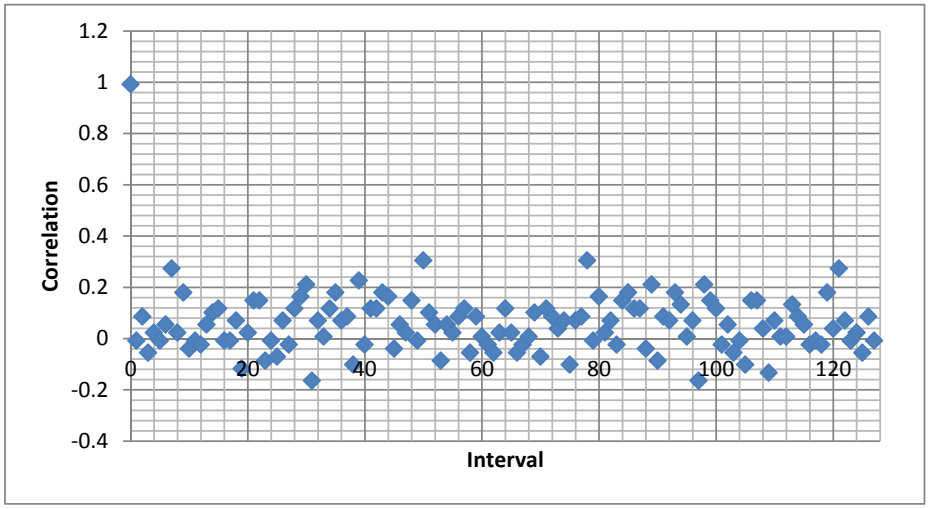
The delay of a circuit is a function of ambient temperature. Although the effect depends on the material, generally the higher the operating temperature of a digital circuit, the greater the propagation delay along interconnects and through semiconductor devices. Thus for all delay-based PUFs, ensuring output stability across a wide operational range is critical. Compared to room-temperature measurements, the error rate of a typical PUF will increase by a few percent as temperature is varied over a wide operational range, such as -40C to +85C which is standard for testing ICs. Variation in supply voltage is another commonly-considered factor. It is expected that this design will exhibit similar behavior in the face of environmental variation, but such testing is left to future work.

5.6 Performance Comparison

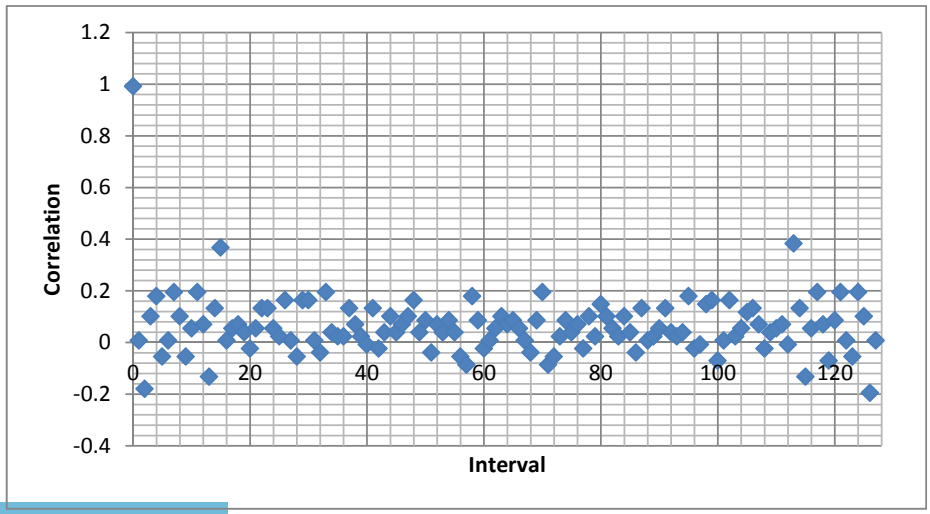
Although the fairly widespread use of measuring Hamming Distance between device responses and within a device's responses helps to provide benchmarks for comparing PUF designs, due to the wide variety of testing procedures and sample sizes it is still quite difficult to make direct comparisons between designs. Furthermore, some authors report results in the raw PUF form only, while others only report the results with post-processing applied. Finally, this design has yet to be integrated into a Challenge-Response Framework. Made of the other



(a) Autocorrelation for device 1.



(b) Autocorrelation for device 2.



(c) Autocorrelation for device 3.

Figure 5.6: Comparison of 128-bit Signature Autocorrelation for three Spartan3E devices.

PUF	Experiment	Reliability	Uniqueness
Proposed	24 128-bit arrays (3072 cells)	6.8%	47.8%
Optical [34]	567 CRPs on 4 tokens	25.25%	49.79%
Coating [1]	31 CRPs on 36 ASICs	$< 5\%$	$\sim$50%
Basic Arbiter [24]	10000 CRPs on 37 ASICs	$< 5\%$	50%
Feed-forward Arbiter [24]	10000 CRPs on 37 ASICs	9.8%	38%
Basic Ring Oscillator [13, 14]	many CRPs on 4 FPGAs	$\sim$0.01%	$\sim$1%
Ring Oscillator with comparator[37]	1024 loops on 15 FPGAs	0.48%	46.14%
SRAM [17]	65440 CRPs on different FPGAs	<12%	49.97%
Butterfly[23]	64 CRPs on 36 FPGAs	<6%	$\sim$50%
Anderson[3]	36 128-bit arrays	3.6%	$\sim$48%

Table 5.5: Performance Comparison

works perform experiments based from the perspective of Challenge-Response Pairs (CRP). With these caveats in mind, Table 5.5 is provided as only a reference point demonstrating that the design proposed in this thesis is indeed competitive. It is also worth mentioning that the design still has room for optimization, since a comprehensive examination of the effect of LUT pin choice on Uniqueness and Reliability was not performed.

5.7 Design Applicability

PUF applications related to secret-key generation in cryptography require absolute repeatability– the error rate should be very close to 0. On the other hand, applications related to authentication and signature generation are more tolerant to errors. A fuzzy acceptance mechanism can be used to distinguish between a device's noisy signature, and the signature of other devices. This particular design would be best relegated to such an application. In particular the low power consumption and relatively small footprint makes it suitable for embedded applications such RFID authentication systems.

Another potentially interesting application, which is only mentioned in brief, is a random number generator. As discussed in [22] a linear-feedback shift register (LFSR) can be added to a PUF, using it as a seed. It is possible that the relative noisiness of the raw design can be leveraged in such a way. For example, the LUT pin configurations could be tuned to achieve good uniformity but high error rate.

CHAPTER 6. CONCLUSION AND FUTURE DIRECTIONS

An exploration of ways to increase circuit reliability before any postprocessing is applied is an important next step. This might involve further testing on the effects of cell layout, as well as a more exhaustive exploration on the effect of permuting the pin arrangement on the LUTs of a cell.

A number of qualitative tests can yet be performed on the PUF design presented in this paper. For example, the actual settling time of the circuit would be an interesting bit of information to determine the maximum sampling rate. Further tests related to durability must be performed, such as design behavior in the presence of varying operating voltage, as well as device aging, have yet to be performed.

Finally, future work should also see a shift in focus to the integration and testing of the design in higher-level applications, such as the basic challenge-response framework, or an authentication scheme. The ability to adjust the relative lengths of the PUF routing is a unique asset in that it can potentially be used to optimize the circuit for different applications.

BIBLIOGRAPHY

[1] "Read-proof hardware from protective coatings." in *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop*, 2006, pp. 369–383. [Online]. Available: http://www.iacr.org/cryptodb/archive/2006/CHES/29/29.pdf

[2] M. Akgun and M. Caglayan, "Puf based scalable private rfid authentication," in *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, aug. 2011, pp. 473 –478.

[3] J. Anderson, "A puf design for secure fpga-based embedded systems," in *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, jan. 2010, pp. 1 –6.

[4] R. J. Anderson and M. G. Kuhn, "Low cost attacks on tamper resistant devices," in *Proceedings of the 5th International Workshop on Security Protocols*. London, UK, UK: Springer-Verlag, 1998, pp. 125–136. [Online]. Available: http://dl.acm.org/citation.cfm?id=647215.720528

[5] R. Bassil, W. El-Beaino, A. Kayssi, and A. Chehab, "A puf-based ultra-lightweight mutual-authentication rfid protocol," in *Internet Technology and Secured Transactions (ICITST), 2011 International Conference for*, dec. 2011, pp. 495 –499.

[6] C. Bohm, M. Hofer, and W. Pribyl, "A microcontroller sram-puf," in *Network and System Security (NSS), 2011 5th International Conference on*, sept. 2011, pp. 269 –273.

[7] C. Brookson. (2002) Can you clone a gsm smart card (sim)? [Online]. Available: http://www.brookson.com/gsm/clone.pdf

[8] W. Choi, S. Kim, Y. Kim, Y. Park, and K. Ahn, "Puf-based encryption processor for the rfid systems," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, 29 2010-july 1 2010, pp. 2323 –2328.

[9] P. F. Cortese, F. Gemmiti, B. Palazzi, M. Pizzonia, and M. Rimondini, "Efficient and practical authentication of puf-based rfid tags in supply chains," in *RFID-Technology and Applications (RFID-TA), 2010 IEEE International Conference on*, june 2010, pp. 182 –188.

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

[11] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, Mar. 2008. [Online]. Available: http://dx.doi.org/10.1137/060651380

[12] S. Eiroa and I. Baturone, "Hardware authentication based on pufs and sha-3 2nd round candidates," in *Microelectronics (ICM), 2010 International Conference on*, dec. 2010, pp. 319 –322.

[13] B. Gassend, "Physical Random Functions," Master's thesis, MIT, MA, USA, 2003.

[14] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 148–160.

[15] S. Goren, O. Ozkurt, A. Yildiz, and H. Ugurdag, "Fpga bitstream protection with pufs, obfuscation, and multi-boot," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on*, june 2011, pp. 1 –2.

[16] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for fpga ip protection," in *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, aug. 2007, pp. 189 –195.

[17] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *Proceedings of the 9th international workshop on Cryptographic Hardware and Embedded Systems*, ser. CHES '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 63–80. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74735-2_5

[18] M. Hofer and C. Bohm, "An alternative to error correction for sram-like puf," in *CHES - Workshop on Cryptographic Hardware and Embedded Systems*, p. 335350.

[19] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on fpgas," in *Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*, dec. 2010, pp. 298 –303.

[20] C. Hu. Solving today's design security concerns. [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp365_Solving_Security_Concerns.pdf

[21] ITAR-TASS News Agency. (2012) Phobos-grunt chips supposedly were counterfeit. [Online]. Available: http://www.itar-tass.com/en/c32/330734.html

[22] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan, "Lightweight mutual authentication and ownership transfer for rfid systems," in *INFOCOM, 2010 Proceedings IEEE*, march 2010, pp. 1 –5.

[23] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly puf protecting ip on every fpga," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, june 2008, pp. 67 –70.

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

[25] R. Lowry. (2007) Counterfeit electronic components - an overview. [Online]. Available: http://www.ors-labs.com/pdf/MASH07CounterfeitDevice.pdf

[26] R. Maes, P. Tuyls, and I. Verbauwhed, "Intrinsic pufs from flip-flops on reconfigurable devices," in *in Benelux Workshop Information and System Security (WISSec 08)*, 2008.

[27] R. Maes and I. Verbauwhede, "Physically unclonable functions: a study on the state of the art and future research directions," in *In Towards Hardware-Intrinsic Security, Security and Cryptology*, 2010.

[28] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large scale characterization of ro-puf," in *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, june 2010, pp. 94 –99.

[29] A. Maiti, V. Gunreddy, and P. Schaumon, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *IACR ePrint*, 2011.

[30] M. Majzoobi, F. Koushanfar, and S. Devadas, "Fpga puf using programmable delay lines," in *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, dec. 2010, pp. 1 –6.

[31] ——, "Fpga-based true random number generation using circuit metastability with adaptive feedback control," in *Proceedings of the 13th international conference on Cryptographic hardware and embedded systems*, ser. CHES'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 17–32. [Online]. Available: http://dl.acm.org/citation.cfm?id=2044928.2044931

[32] S. Morozov, A. Maiti, and P. Schaumon, "An analysis of delay based puf implementations on fpga," in *Reconfigurable Computing: Architectures, Tools and Applications*, vol. 5992, 2010, pp. 382–387.

[33] L.-T. Pang and B. Nikolic, "Measurement and analysis of variability in 45nm strained-si cmos technology," in *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE*, sept. 2008, pp. 129 –132.

[34] R. Pappu, R. Recht, and J. Taylor, "Physical one-way functions," in *Science*, SEP. 2002, pp. 2026–2030.

[35] P. Sedcole and P. Cheung, "Within-die delay variability in 90nm fpgas and beyond," in *Proceedings of IEEE International Conference on Field Programmable Technology*, 2006.

[36] E. Simpson and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," in *Cryptographic Hardware and Embedded Systems CHES 2006*, vol. 4249, sept. 2006, p. 311323.

[37] G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, june 2007, pp. 9 –14.

[38] V. van der Leest, E. van der Sluis, G. J. Schrijen, P. Tuyls, and H. Handschuh, "Efficient implementation of true random number generator based on sram pufs," in *Cryptography and Security*, 2012, pp. 300–318.

[39] K. Wold and C. H. Tan, "Analysis and enhancement of random number generator in fpga based on oscillator rings," in *Reconfigurable Computing and FPGAs, 2008. ReConFig '08. International Conference on*, dec. 2008, pp. 385 –390.

[40] Xilinx. Spartan-3e libraries guide for hdl designs. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/spartan3e_hdl.pdf

[41] K. Yang, K. Zheng, Y. Guo, and D. Wei, "Puf-based node mutual authentication scheme for delay tolerant mobile sensor network," in *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, sept. 2011, pp. 1 –4.