Winter 11-1-2019

Study of Distributed Versus Compressed Layouts for Physically Unclonable Functions

Joshua Trujillo

Follow this and additional works at: https://digitalrepository.unm.edu/ece_etds

Part of the Electrical and Computer Engineering Commons

Recommended Citation

This Dissertation is brought to you for free and open access by the Engineering ETDs at UNM Digital Repository. It has been accepted for inclusion in Electrical and Computer Engineering ETDs by an authorized administrator of UNM Digital Repository. For more information, please contact amywinter@unm.edu, Isloane@salud.unm.edu, sarahrk@unm.edu.
Joshua Trujillo
Candidate

Electrical and Computer Engineering
Department

This dissertation is approved, and it is accepted in quality and form for publication:

Approved by the Dissertation Committee:

Dr. Payman Zarkesh-Ha, Chairperson

Dr. Christos Christodoulou

Dr. Adam Hecht

Dr. Paul De Rego (Honeywell FM&T)
STUDY OF DISTRIBUTED VERSUS COMPRESSED LAYOUTS FOR PHYSICALLY UNCLONABLE FUNCTIONS

BY

JOSHUA J. TRUJILLO, SR.

B.S., Computer Engineering, University of New Mexico, 2006
M.S., Computer Engineering, University of New Mexico, 2012

DISSERTATION
Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Engineering

The University of New Mexico
Albuquerque, New Mexico

December 2019
Dedication

Dedicated to my wife, Alison, our children, Samantha, Joshua Jr., and Abigail, and to my parents, Samuel and Kathryn Trujillo.
Acknowledgements

I would like to thank my committee, including my advisor, Professor Payman Zarkesh-Ha (UNM), Professor Christos Christodoulou (UNM), Dr. Paul De Rego (Honeywell FM&T), and Professor Adam Hecht (UNM). Thank you all for being on my committee and helping me complete my academic journey.

I would also like to thank my colleagues at Honeywell FM&T. You all have encouraged, assisted, and inspired me throughout my Technical Fellowship Program at Honeywell.

I would like to thank my friends at church, for their prayers and encouragement.

Last, and most importantly, I would like to thank my family. I’d like to thank my wife, Alison, for her encouraging words, and for being Superwoman, taking care of our house and children those many nights that I was stuck in the laboratory or in front of a keyboard. You are such a wonderful wife and mother. Thank you to my children, for being patient while I was working long hours finishing this research. I’d also like to thank my parents, for buying my first computers, including the RadioShack TRS-80 that I still have, and for always teaching me to strive for the best and do what is right. Although my dad is in Heaven, I hope he’s proud of what I’ve accomplished.
Study of Distributed Versus Compressed Layouts for Physically Unclonable Functions

by

Joshua J. Trujillo, Sr.

B.S., Computer Engineering, University of New Mexico, 2006
M.S., Computer Engineering, University of New Mexico, 2012
Ph.D., in Engineering, University of New Mexico, 2019

ABSTRACT

Society continues to depend on electronics for everything from smart systems in our homes to cellphones and tablets, which are more powerful today than many desktop computers that are still in use [1]. With increased consumption of these electronic products comes an increase in problems, such as counterfeit integrated circuits being sold as genuine integrated circuits. This is just one of many problems that corporations and end users are having to deal with in this digital age.

Software security has been accepted as a problem by both the media and general public, but only recently has hardware security begun to come out as a problem as important, if not more important, than software security. When there is an issue with software, a software patch can be
released to fix this issue. Due to the nature of hardware, this is not possible with hardware security problems. Often times, firmware can be updated to partially address the problem, as in the case with the Intel and AMD processor flaws in the news recently [2]. However, there are many times that the only solution is to take the hardware system offline and replace the questionable components.

Hardware security researchers have been trying to find ways to use security circuits in hardware design to help combat this problem. One type of security circuit being researched and used today is called a Physically Unclonable Circuit (PUF). PUF circuits can be added onto an existing integrated circuit design, interrogated at fabrication for challenge-response pairs which are stored in a database, then checked against that database at any time in the lifecycle of that integrated circuit [3].

This research contributes to PUF security circuits by showing how different ways of laying out a circuit design on an integrated circuit can change the performance of the circuit, as well as if PUF circuits designed on SiGe BiCMOS can be used in the same way that PUF circuits fabricated using standard CMOS processes are used today.
# TABLE OF CONTENTS

TABLE OF CONTENTS ........................................................................................................... vii

LIST OF FIGURES .................................................................................................................... ix

LIST OF TABLES ......................................................................................................................... xi

1. INTRODUCTION ..................................................................................................................... 1
   1.1. Hardware Security Concerns in ASICs ............................................................................. 2
   1.2. Research Objectives ......................................................................................................... 3

2. BACKGROUND ....................................................................................................................... 6
   2.1. Application Specific Integrated Circuits .......................................................................... 6
   2.2. Physically Unclonable Functions ..................................................................................... 7
       2.2.1. PUF Fundamentals .................................................................................................... 7
       2.2.2. Enrollment and Verification ..................................................................................... 9
       2.2.3. Strong and Weak PUFs ......................................................................................... 12
   2.3. Types of PUFs ................................................................................................................. 13
       2.3.1. SRAM PUF ............................................................................................................. 14
       2.3.2. Arbiter PUF ............................................................................................................ 15
       2.3.3. Ring Oscillator PUF .............................................................................................. 16
       2.3.4. Cutting-Edge PUFs ............................................................................................... 17

3. ROSA PUF HARDWARE DESIGN ....................................................................................... 19
   3.1. ROSA PUF ASIC ............................................................................................................. 20
       3.1.1. ROSA Pad Ring Structure ....................................................................................... 22
       3.1.2. ROSA SRAM PUF Design ..................................................................................... 27
       3.1.3. ROSA Arbiter PUF Design ................................................................................... 36
       3.1.4. ROSA Ring Oscillator PUF Design ....................................................................... 43
   3.2. ROSA PUF Test Setup ..................................................................................................... 54
       3.2.1. Printed Circuit Boards ........................................................................................... 54
       3.2.2. ZedBoard ................................................................................................................ 73
       3.2.3. Personal Computer ............................................................................................... 76

4. PUF SOFTWARE .................................................................................................................... 77
   4.1. Interface Software ........................................................................................................... 77
   4.2. Data Processing ............................................................................................................. 79
       4.2.1. C# .......................................................................................................................... 79
       4.2.2. Matlab .................................................................................................................... 80

5. PUF METRICS .................................................................................................................... 81
   5.1. NIST – Frequency (Monobit) Test ................................................................................ 81
   5.2. NIST – Frequency within a Block Test ......................................................................... 83
LIST OF FIGURES

FIGURE 1.1 ROSA (RING-Oscillator, SRAM, Arbiter) PUF ASIC4
FIGURE 1.2 High-level schematic of the PUF circuits chosen in this project5
FIGURE 2.1 Ring Oscillator Frequency Calculation7
FIGURE 2.2 PUF Challenge and Response8
FIGURE 2.3 PUF Enrollment and Verification10
FIGURE 2.4 TV Corners for Testing PUFs11
FIGURE 2.5 SRAM PUF14
FIGURE 2.6 Arbiter PUF15
FIGURE 2.7 Ring Oscillator PUF16
FIGURE 3.1 ROSA PUF System Diagram19
FIGURE 3.2 PUF Die Blocks21
FIGURE 3.3 Two pad structures available, with 160 µM and 80 µM pitch23
FIGURE 3.4 List of pads available in library25
FIGURE 3.5 I/O pads used in the ROSA ASIC26
FIGURE 3.6 Pad ring arrangement for the ROSA ASIC26
FIGURE 3.7 SRAM Schematic – Unit cell27
FIGURE 3.8 SRAM Schematic – Shift Register29
FIGURE 3.9 SRAM Schematic – 2x2 PUF30
FIGURE 3.10 SRAM Layout – Unit cell31
FIGURE 3.11 SRAM Layout – 2x2 PUF32
FIGURE 3.12 SRAM 150x125 PUF Floorplan33
FIGURE 3.13 SRAM PUF Timing Diagram34
FIGURE 3.14 SRAM Simulation – Unit cell35
FIGURE 3.15 SRAM Simulation – 2x2 PUF36
FIGURE 3.16 Schematic of an arbiter PUF with 3-bit challenge bits37
FIGURE 3.17 Arbiter PUF Layout – 5-Stage38
FIGURE 3.18 Arbiter PUF Layout – 5-Stage38
FIGURE 3.19 Arbiter PUF Floorplan39
FIGURE 3.20 Arbiter PUF Timing Diagram40
FIGURE 3.21 Arbiter PUF Test Parameters41
FIGURE 3.22 Arbiter PUF 5-Stage with 10fF Capacitor on Output 2A42
FIGURE 3.23 Arbiter PUF 5-Stage with 10fF Capacitor on Output 2B43
FIGURE 3.24 Schematic for 5-Stage RO PUF with Counter and Shift Register44
FIGURE 3.25 RO PUF Layout – 11-Stage45
FIGURE 3.26 RO PUF Layout – Compressed and Distributed46
FIGURE 3.27 RO PUF Floorplan46
FIGURE 3.28 RO PUF Timing Diagram47
FIGURE 3.29 RO PUF 7-Stage T-Spice Simulation48
FIGURE 3.30 ROSA PUF ASIC Complete Floorplan49
FIGURE 3.31 ROSA PUF Final Layout50
FIGURE 3.32 Packaged PUF Die51
FIGURE 3.33 Measurements of Bond Pads on Bare Die52
FIGURE 3.34 Close-Up of Top of Die, including Leveling Blocks52
FIGURE 3.35 Close-Up of Top of Die, including Leveling Blocks53
FIGURE 3.36 Complete PUF Test Setup54
Figure 3.37 Schematic for Main PUF Breakout PCB55
Figure 3.38 Layout for Main PUF Breakout PCB56
Figure 3.39 Layout for the Main PUF Breakout PCB, with Copper Pour57
Figure 3.40 Main PUF Breakout - Bare PCB – Top Side58
Figure 3.41 Main PUF Breakout - Bare PCB – Bottom Side59
Figure 3.42 PGA Socket Used for ROSA PUF ASIC60
Figure 3.43 PUF Breakout PCB, Populated61
Figure 3.44 Schematic for Power PCB62
Figure 3.45 Layout for Power PCB63
Figure 3.46 Layout for Power PCB with Copper Pour64
Figure 3.47 PUF Power - Bare PCB – Top Side65
Figure 3.48 PUF Power - Bare PCB – Bottom Side66
Figure 3.49 Power PCB - Populated67
Figure 3.50 Schematic for PMOD-to-Terminal PCB68
Figure 3.51 Layouts of PMOD-to-Terminal PCB - With and Without Copper Pour69
Figure 3.52 PMOD-to-Terminal PCB - Top and Bottom70
Figure 3.53 PMOD-to-Terminal PCB in Use71
Figure 3.54 ZedBoard – Top72
Figure 3.55 ZedBoard - Bottom73
Figure 3.56 FPGA Heatsink74
Figure 3.57 UART Connector on ZedBoard75
Figure 4.1 C# PUF Data Application77
Figure 4.2 C# PUF Data Application - PUF Types78
Figure 5.1 NIST Frequency (Monobit) Test Calculation81
Figure 5.2 NIST Frequency Monobit Example82
Figure 5.3 Incomplete Gamma Function83
Figure 5.4 NIST Frequency within a Block Test Calculation84
Figure 5.5 Hamming Distance Example85
Figure 5.6 Intra-Die Hamming Distance Calculation86
Figure 5.7 Inter-Die Hamming Distance Calculation86
Figure 6.1 SRAM PUF NIST Frequency (Monobit) Test88
Figure 6.2 SRAM PUF NIST Frequency Test within a Block89
Figure 6.3 SRAM PUF Intra-Die Hamming Distance91
Figure 6.4 SRAM PUF Inter-Die Hamming Distance92
Figure 6.5 Distributed Arbiter 4-Stage Bit Flipping - Averages94
Figure 6.6 Distributed Arbiter 4-Stage - Number of Bit Flips Total95
Figure 6.7 Distributed Arbiter 9-Stage Bit Flipping - Averages97
Figure 6.8 Distributed Arbiter 19-Stage Bit Flipping - Averages99
Figure 6.9 RO PUF - 7-Stage Distributed - 3D Surface Plot100
Figure 6.10 RO PUF - 7-Stage Distributed - Heat Map103
Figure 7.1 Arbiter PUF Distributed and Compressed Layout Metrics105
LIST OF TABLES

Table 6.1 SRAM PUF Results 88
Table 6.2 Arbiter PUF 4-Stage Distributed Results 93
Table 6.3 Arbiter PUF 9-Stage Distributed Results 96
Table 6.4 Arbiter PUF 19-Stage Distributed Results 98
Table 6.5 Arbiter PUF 128-Stage Compressed Results 100
1. INTRODUCTION

sub-rosa, adj. /ˌsəb-ˈrō-zə/: SECRETIVE, PRIVATE
sub-rosa, adv. /ˌsəb-ˈrō-zə/: in confidence, SECRETLY
- The Merriam-Webster Dictionary

The term sub-rosa is a Latin phrase that translates to “under the rose” [4]. For centuries, sub-rosa has been used to indicate that a conversation should be kept secret, or in confidence. The Scottish Government continues to hold sub-rosa meetings to this day, as mentioned on one of their websites [5]. History shows that secrecy has been important throughout human history, even back to Biblical times.

In today’s digital, connected world, secrecy is still desired, but often compromised. Our computers and cell phones are at risk constantly, with the threat of malware encrypting our data in an attempt to collect a ransom, as well as threats to databases with personal information [6]. Corporations are constantly fending off competitors trying to gain the upper hand through corporate espionage [7], and feeling the repercussions when the products they manufacture are compromised by attackers.

Another issue that corporations have to deal with is counterfeiting. Older products are often rebranded and resold as new products. Low-quality reproductions or redesigns are often
sold as genuine products, often failing prematurely or being disabled by the manufacturer, as in the case of FTDI’s serial-to-USB chips [8] and many inkjet manufacturers [9].

Hardware security concerns are receiving more attention lately, in the same way that software security concerns have been in the spotlight in recent decades. The media is taking notice, with an increase in stories in the news about hardware attacks and the damage caused by them. In a recent example, hackers found a way to trick Tesla’s Autopilot into switching lanes, even into oncoming traffic [10].

Another recent hardware attack, reported by Bloomberg, stated that a hardware infiltration was discovered on motherboards used in servers in multiple datacenters [11]. This shift in security threats has created momentum around hardware security, and those trying to protect their hardware.

1.1. Hardware Security Concerns in ASICs

Maintaining a secure and trusted environment with Application Specific Integrated Circuits (ASICs) is important to end-users because of the potential damage and loss that can result from an attacker replacing genuine ASICs with counterfeit ASICs. In order to protect this trusted environment, the entire supply chain of integrated circuit design, production and testing must be scrutinized.

The process begins with the software used to design and simulate the integrated circuits. Once the design is finalized, it is sent to the fabrication facility, where masks are created, which will be used to create each layer of the wafer containing multiple copies of the integrated circuit
Next, the die on the wafer are diced, then sent to be packaged in an assembly that can be used in the final circuit.

There are many opportunities in this process for a malicious actor to negatively affect the circuit, throughout the life of the circuit, including the design software, masks used to create the die, packaging of the die, and everything in between. This research gives new insight into how certain security circuits can help combat these hardware security concerns using circuits called Physically Unclonable Functions (PUFs).

1.2. Research Objectives

This research is a collaboration between the University of New Mexico (UNM) and Honeywell FM&T. Honeywell FM&T has funded this research to addresses hardware security concerns, with a focus on PUFs. PUF circuits have been implemented on both ASICs and FPGAs, using various designs, which will be covered in a later section of this document.

Literature searches have shown PUFs implemented on ASICs using Silicon (Si) substrates, however, no evidence of PUFs designed on Silicon Germanium (SiGe) Bipolar/CMOS (BiCMOS) could be found. The SiGe BiCMOS process has many excellent properties, and its use has increased in industry. These benefits will be covered in a later chapter.

The goal of this research was to design, build, and test well-known PUF designs on an ASIC, using Tower Jazz’s 180 nanometer Silicon Germanium (SiGe) Bipolar/CMOS (BiCMOS) process in order to evaluate the feasibility of using PUFs on SiGe BiCMOS. These PUF designs include Ring Oscillator (RO) PUF, SRAM, and Arbiter PUF, affectionately known as the RO-S-A PUF ASIC, or ROSA, as shown in Figure 1.1.
Additionally, the design includes circuit layouts in unit cells with similar functionality, but with different length and width dimensions. One set of circuits is implemented in a localized area on the die, referred to as a compressed layout. Another set of circuits is spread out across the die, referred to as a distributed layout.

The performance of the ROSA PUF circuits will be measured to determine which layout method improves the metrics from each of the PUF circuits. There is benefit to extensive temperature-voltage (TV) testing outside of nominal values when developing a new PUF. This is outside the scope of this research, since the PUF circuits used in the ROSA ASIC are well understood.

In fact, when choosing PUF circuits to implement for this research, one of the factors used in the selection process was how well a PUF design is understood by the PUF community. Often times, in academic competitions, cutting-edge designs were not stable enough to complete the competition, while more basic, well-understood designs were implemented on time, with many of those designs winning in several categories in those same competitions.
Figure 1.2 High-level schematic of the PUF circuits chosen in this project

Figure 1.2 shows a high-level schematic of the types of PUF circuits used in ROSA, including Ring Oscillator PUF, Arbiter PUF, and SRAM PUF. The functionality of these PUFs, along with the specific implementation of the ROSA PUFs will be covered in later chapters. These PUF circuits are well understood in research [13], and are a conservative choice to use when comparing other variables, such as different layouts and a new fabrication process.
2. BACKGROUND

“A person who never made a mistake never tried anything new.”
- Albert Einstein

2.1. Application Specific Integrated Circuits

Microprocessors in modern computers, phones, and tablets are designed to handle multiple operations, which are specified in software running on an operating system (OS). This flexibility allows a multitude of functionality to be executed on a single integrated circuit. This flexibility adds complexity, which in turn slows down program execution. Slower performance is a tradeoff that many integrated circuit designers are willing to accept in order to gain flexibility.

If speed is the main driving factor in the design, then the best option for integrated circuit design is to build a device without the flexibility of a microprocessor, called an Application Specific Integrated Circuit (ASIC), which is a special purpose processor [14]. What the ASIC loses in application flexibility, it gains in performance. Because all resources on the integrated circuit are specifically designed to be used by the ASIC, time is not lost due to resource sharing, as in the case of a microprocessor.

The ROSA ASIC designed and fabricated for this research was built with one purpose: to process PUF circuit data. There are no programmable circuits on the ROSA ASIC, only power and data lines. Because of this, the PUF circuits on ROSA run extremely fast. For example, one of the slower, 11-stage Ring Oscillator PUF circuits on ROSA used a 3.050 microsecond start-
stop time, then returned a value of 3,366 from the frequency counter in my test setup. As we can see in Figure 2.1, we can calculate the frequency of the circuit, which is running at speeds over a Gigahertz.

\[
\text{frequency} = f = \frac{\text{cycles}}{\text{start} - \text{stop time}} = \frac{3366 \text{ cycles}}{3.050 \mu\text{sec}} = 1.10361e^9 \text{ cycles/sec} = 1.10361 \text{ GHz}
\]

**Figure 2.3 Ring Oscillator Frequency Calculation**

Now that we understand the benefits of ASICs, let us review PUF circuits, including what types of PUF circuits are popular, in order to better understand the ROSA PUF integrated circuits.

### 2.2. Physically Unclonable Functions

One method of improving security in ASIC designs is to implement PUF circuits alongside the other circuitry in the device. The following sections outline the fundamentals of PUF circuits, as well as other important concepts that will give enough background to understand more complex PUF concepts. These other concepts include enrollment and verification, and strong and weak PUFs.

#### 2.2.1. PUF Fundamentals

PUFs are physical functions that take a challenge as input, and produce an output called a response. They are physical functions because the response from an ideal PUF circuit uses small changes in the materials of the semiconductor device to generate response bits that are different between die with the same exact circuitry, even die that come from the same wafer [15]. These
differences are exploited by PUF circuit designers [16], with a goal of generating response bits that have the same chance of being a one as they does a zero. The challenge and response relationship can be seen in Figure 2.2.

![PUF Challenge and Response](image)

**Figure 2.4 PUF Challenge and Response**

However, a PUF circuit that has the same probability of generating a one or a zero bit is known as an ideal PUF, and is not seen in practice. The reason for this could be because of bias that is created in the design, such as path length differences in the circuit, or even something as small as inconsistent doping in the substrate [17]. These subtle differences work to the PUF designer’s advantage because the circuit is unclonable, meaning that it is highly improbable that someone could purposely manufacture two PUF circuits that are identical.

Even with full knowledge of the PUF circuit design, the PUF circuit should not be able to be cloned. This unclonability means that PUF circuits are good hardware security solutions for helping manufacturers deter counterfeiting, to protect intellectual property (IP) [18]. They can be used in device authentication, and other applications where having a unique challenge and response are used in the system for security.

Besides being unique, the response must be repeatable on the same PUF circuit using the same challenge. The repeatability of the PUF circuit is one metric against which PUFs are
measured and compared against different designs. These metrics will be covered in a later chapter.

2.2.2. Enrollment and Verification

Once a PUF circuit is manufactured, the PUF can be interrogated, ideally at the fabrication facility. First, a challenge string is sent to the circuit, which will result in a response being sent back. The challenge and response from the same PUF circuit, are together known as the Challenge-Response Pair (CRP) [19]. The number of digits sent as the challenge depends on the type of PUF circuit, since some PUF circuits require multiple challenge bits, while others require only a single challenge bit. For every challenge we send to the PUF, we should get back a unique response.

The initial interrogation of the PUF circuit is called enrollment. Challenge strings are chosen, sent to the system, and then the CRPs that are sent back are recorded in a database. The CRP data is enrolled in the database at the beginning of the lifecycle of the integrated circuit to improve security, as shown in Figure 2.3. Chances are low that the circuit was compromised that soon after production.
Once the system is fielded, it has the potential for being tampered with. However, we can verify that the integrated circuit is the same device we expect to see. To verify the integrated circuit, we retrieve our original CRPs, which are stored in a database, and compare those values with the current CRPs we get back from the circuit to make sure that the responses match. This process of comparing the CRPs from the system in the field against the CRPs in the database is called verification. We are verifying that the system has not been tampered with.

The response from the PUF circuit can change, through a process called bit flipping. Bit flipping is when a value changes, or flips, from zero to one, or vice-versa. This can occur when the PUF circuit has instability. Instability can come from a poor PUF circuit design, as mentioned previously. Bit flipping can also occur due to changes in environment, such as fluctuations in temperature, humidity, and voltage.
It is important, before fielding a PUF system, to characterize the PUF circuits while running in extreme conditions. For example, temperature testing is typically performed at -40°C and +80°C, where nominal temperature is considered 25°C. Voltage testing is typically performed at +/-10% of nominal voltage. Figure 2.4, below, shows the 9 temperature-voltage (TV) corners that PUF circuits are typically tested at.

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Voltage 1</th>
<th>Voltage 2</th>
<th>Voltage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>+80°C</td>
<td>+2.97V</td>
<td>+80°C</td>
<td>+80°C</td>
</tr>
<tr>
<td></td>
<td>+3.3V</td>
<td>+3.63V</td>
<td>+3.63V</td>
</tr>
<tr>
<td>+25°C</td>
<td>+2.97V</td>
<td>+25°C</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>+3.3V</td>
<td>+3.63V</td>
<td>+3.63V</td>
</tr>
<tr>
<td>-40°C</td>
<td>+2.97V</td>
<td>-40°C</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td>+3.3V</td>
<td>+3.63V</td>
<td>+3.63V</td>
</tr>
</tbody>
</table>

Figure 2.6 TV Corners for Testing PUFs

As the PUF circuit ages, bit flips can occur. Aging studies are also important to perform before fielding a PUF, to understand any instability in the circuit. To improve stability, we can use error correction to compensate for this [20]. Since we know there will be at least some instability, we know that the numbers we found at enrollment will not match up completely. We can then say that if the value is greater than a certain threshold, such as 80%, where 20% of the bits did not match up with the enrollment values in the database, then we have a match. There is
a danger with too much instability, because then the percentage of acceptable bits must be lowered so much that PUF security can be more easily compromised. As the number of bits that must match are lowered, the chance of an attacker correctly guessing the CRPs is increased.

2.2.3. Strong and Weak PUFs

PUF circuits can be categorized as either “strong” or “weak”. These terms are not a reflection of how secure they are, but rather it describes the number of responses that can be generated from the PUF. PUFs that can generate a large number of CRPs are considered strong PUFs, while PUFs that generate a small number of CRPs, even down to a single CRP, are considered weak PUFs [21].

Strong PUFs have an advantage, because if a malicious actor obtains the response from a single challenge, then there are other CRPs that can be used. If a malicious actor does not know which CRP will be used, then they only have a $1/n$ chance of subverting the system, where $n$ is the number of CRPs.

One example of a strong PUF is an Arbiter PUF, which takes a single input for each stage. Additional arbiter stages increase the number of challenge inputs in the PUF circuit. SRAM PUFs, on the other hand, are an example of a weak PUF. This is because there is only a single response, with no challenge bits. Once the response of the circuit is known by a malicious actor, the security of the circuit can be more easily subverted.
2.3. Types of PUFs

Types of PUFs include Optical PUFs, Coating PUFs, Arbiter PUFs, Ring Oscillator PUFs and SRAM PUFs. The Optical PUF is considered the first PUF, and was introduced by Pappu in 2002 [22]. The Optical PUF uses interference patterns on a transparent optical medium. The Coating PUF uses isolating and conducting material for a coating on the chip that can be used to detect attacks.

The SRAM PUF uses SRAM blocks to generate data once the circuit is stabilized after power up. The Arbiter PUF compares digital path delays, using an arbiter circuit that chooses which of the two paths has the smaller delay. The Ring Oscillator PUF uses pairs of inverter chains, generating a single bit for each pair of ring oscillator that is compared against one another.

These different PUFs have strengths and weaknesses that can make them a good choice for a system, based on the system requirements. It is left to the system designers to decide which type of PUF will work best in their system. This research focuses on SRAM, Arbiter, and Ring Oscillator PUFs, so these PUF circuits will be explained in detail, below, as well as other, cutting-edge PUFs that have been introduced in recent publications.
2.3.1. SRAM PUF

SRAM PUFs are made up of the same type of SRAM cells used for memory applications. This is beneficial for applications where a SRAM is already part of the main integrated circuit design, since the SRAM array can also be used as a PUF circuit [23]. When a SRAM cell is powered up, the voltage in the cell settles to either a high-state or low-state, from an initial unstable state. Power supply ramp time has an effect on SRAM PUFs, as demonstrated in other research [24]. Whether or not the cell powers up to the low-state or high-state is influenced by the random threshold voltage mismatch in the two cross-coupled inverters that make up the SRAM cell, as shown in Figure 2.5.

![Figure 2.7 SRAM PUF](image)

With a well-designed SRAM cell, it is straight forward to create arrays of SRAM cells, each addressable by column and row. With this type of layout, the size of the SRAM cell can be matched with the available space on the integrated circuit, using space remaining after the main
design of an ASIC is completed. Another option is to repurpose a SRAM circuit used in the main design, which serves the purpose of Random Access Memory (RAM), but can also act as the PUF circuit.

2.3.2. Arbiter PUF

The Arbiter PUF is a type of timing-based PUF. A pair of multiplexers (MUXs), identically configured, make up a single arbiter MUX stage. Output from one arbiter MUX stage is the input to the next arbiter MUX stage [25].

After the final arbiter MUX stage, the outputs from each MUX are sent to the arbiter, which is an edge triggered flip-flop. Whichever output from the final arbiter MUX stage reaches the arbiter first sets its corresponding output line high, as shown in Figure 2.6. Once that line is set high, the other output line is logically forced low.

![Figure 2.8 Arbiter PUF](image)

Delays along the circuit path in the Arbiter PUF are from manufacturing process variations, which subtly modify the delay paths. As different challenge bits are used, they are
sent to each arbiter MUX stage, different responses are seen at the two Arbiter outputs. The two outputs from the arbiter circuit should always have opposite values. This is a good check to verify that the Arbiter PUF circuit is functioning correctly.

Arbiter PUF path length can be adjusted by increasing or decreasing the number of stages in the Arbiter PUF. For instance, a 10-stage Arbiter PUF can be created by connecting ten pairs of MUXs together, with a 10-stage shift register sending outputs to the MUXs in the arbiter.

2.3.3. Ring Oscillator PUF

The Ring Oscillator (RO) PUF is also a timing-based PUF. The RO PUF is one of the more common PUFs and is well understood [26], and therefore makes a good choice for research that focused on how the PUF is used and not specifically on the performance of the PUF itself. RO PUFs use identically laid out ring oscillator circuits, which oscillate at different frequencies, due to the aforementioned manufacturing process variations.

![Ring Oscillator PUF](image)

*Figure 2.9 Ring Oscillator PUF*

The RO frequencies are captured by a frequency counter, as shown in Figure 2.7. Some RO PUF circuits use on-die circuitry to compare the frequencies of pairs of RO circuits. These
comparisons generate a single bit, either a one or zero, based on which RO circuits being
compared was faster.

Other RO PUF circuit designs choose to measure the generated frequencies off of the die
by counting and comparing two pairs of RO circuits at a time to generate the binary data. Using
this method saves valuable on-die real estate, but requires very fast test equipment to be able to
count the frequencies coming out of the RO circuits. Once the frequencies are counted,
comparing their values is trivial.

2.3.4.Cutting-Edge PUFs

PUF circuit research is very active, based on the recent research papers published at
hardware security conferences around the world. Researchers are working towards PUF circuits
that are smaller, with improved bit error rates, and better reliability under extreme conditions.
Some of the new PUF research today includes Resistive Random Access Memory (RRAM)
PUFs, Carbon Nanotube (CNT) Field Effect Transistor (FET) PUFs, and Resonant-Tunneling
Diode (RTD) PUFs.

RRAMs, also known as ReRAMs and Memristors, have been used in recent PUF
research, due to their low, Native-Bit Error Rates (N-BER), the fact that they are CMOS-
compatible, and due to their small footprint in layouts [27]. RRAMs are also non-volatile and
binary, with two stable states, called the low-resistive state (LRS) and the high-resistive state
(HRS). One research group showed that even under extreme testing conditions, their PUF
performance was excellent, with a N-BER of ≈ 6.1E−6 CITATION Pan19 \(1033\) [28].
CNT-FETs-based PUFs are being used as security keys by another group [29]. Their PUFs have only been produced in small numbers, but the metrics from their testing so far has been promising. CNT-FETs-based PUFs could be implemented on CNT semiconductors, which is a direction that many semiconductor researchers believe the industry is headed [30].

RTDs are basic electrical structures that use quantum confinement, and are the basis of RTD PUFs [31]. Benefits of RTD PUFs include construction that is difficult to clone, can be manufactured on a wafer without additional manufacturing steps added, and have shown promising PUF metrics. RTD PUFs, like all of the other cutting-edge PUFs mentioned here, are relatively new, and are sure to improve over time as research continues.
3. ROSA PUF HARDWARE DESIGN

“Science is about knowing; engineering is about doing.”

- Henry Petroski

This section explains the design of the ROSA PUF hardware, including the ROSA PUF ASIC, as well as the other components in the test system used to communicate with the ROSA PUF test setup to collect data.

Figure 3.10 ROSA PUF System Diagram
The system diagram, in Figure 3.1, shows how all of the components of the ROSA PUF work together. On the left-hand side of the figure are the eleven types of PUF circuits that are implemented on the ROSA PUF. Components on the right-hand side are the PC, FPGA board, and Power PCB. Each of these components and subcomponents will be explained in greater detail, later in this chapter.

Software on the PC, written in C#, communicates to the Zedboard through the USB-to-Serial cable. The cable leads are terminated with standard male jumper wire connectors. These connectors plug into a custom PMOD breakout board, which patches the UART signal to the programmable logic (PL) on the Zynq FPGA on the Zedboard. The Verilog code, running on the PL of the FPGA, parses the command sent over UART, then sends the appropriate challenge string using the PMOD connection to the ROSA PUF ASIC that is mounted on the ROSA PUF breakout board. The FPGA also enables and disables the appropriate power signals coming out of the PMOD connector and in to the ROSA Power PCB.

Once the ROSA PUF generates a response to the challenge, it sends the response data back to the PC, using the aforementioned UART connection. The C# program stores the text file to the PC, stamping the text file with the current date and time. The data is then processed using both C# and Matlab.

3.1. **ROSA PUF ASIC**

The main goal of this research was to design, send to fabrication, and test multiple PUF circuits using the Tower Jazz 180 nanometer (nm) Silicon Germanium (SiGe) Bipolar / CMOS (BiCMOS) process.
Figure 3.11 PUF Die Blocks

Figure 3.2 shows all of the blocks on the ROSA PUF ASIC. The fabrication process used for this die was the SBC18H3, 180 nm technology, with 6 metal layers. The first two metal layers at the top are very thick and are used for distributing both power and ground. This distribution minimized IR drop and switching noise issues. This process also offers triple-well isolation and deep-trench isolation, which results in better latch-up performance.
There are two different power supplies required for this design. The first is the high voltage power supply, which needs to be at 3.3V, and is used for input/output (I/O). The second is the core power supply, which needs to be at 1.8V. The project requirements specified a design that fit within $5 \times 5 \text{mm}^2$. SPICE simulations and design rule checks (DRCs) were completed on each section of the ROSA PUF ASIC, as well as on the entire, completed die. The design was completed DRC clean, with no design rule errors.

Unique to the SBC18H3 Tower Jazz fabrication process are features such as high-performance SiGe heterojunction bipolar transistors (HBTs), capable of running at 240 and 280 GHz. Low-value and high-value unsilicided poly is available, along with N-well resistors, hyper-abrupt P-N junctions, MOS varactors, titanium nitride (TiN) thin film resistors, the choice of a 2.0 V 2.8 $fF/\mu m^2$ metal-insulator-metal (MIM) capacitor that uses layer M4, as well as inductors that use the thick (2.8$\mu m$) aluminum top layer.

However, the ROSA PUF ASIC did not require any of the aforementioned features. This is typical of PUF circuits, which are often designed using very basic cells. ASIC design is covered in the next few sections, including pad ring structure and detailed descriptions of each of the ROSA PUF circuits.

### 3.1.1. ROSA Pad Ring Structure

The Tower Jazz I/O standard cell library offers two different libraries for I/O pad structures. The first pad structure is io160u5V, with 160$\mu$m pitch. The second pad structure is io80u5V, with 80$\mu$m pitch. These two pad structures are shown in Figure 3.3. If the layout is pad-limited, then the io80u5V pad structure should be used. However, the ROSA PUF ASIC was
not pad limited, so the io160u5V pad structure was used. This larger pad patch is beneficial for wire bonding, making the wire bonding process both more reliable and easier.

![Figure 3.12 Two pad structures available, with 160 µm and 80 µm pitch](image)

Tower Jazz digital I/O usage guidelines were followed, including ensuring that the pads were used in a contiguous pad ring, using corner cells supplied by the vendor. ESD protection circuits are part of the io80u5V pad structure and integrate well with the corner cells, which use large protection diodes. Each of the pads have between 8 and 11 metal bus lines used for power, ground, and ESD rings, and are designed so that these lines connect when cells are placed adjacent to one another.
It is possible to use up to 8 nets, including VDD, VDDO, VDDP, VGG, VSS, VSSO, and VESD, which is an internal net. To reduce the number of electrical nets, many of these can be shorted inside the pad ring. The minimum requirement is three power supply connections, which are core power, I/O power, and a common ground.

Once all of the functional I/O pads were selected, additional pads were added, including power pads, four corner cells, and a single power-up sequencing pad. If the chip core power and I/O operate at different voltages, then a minimum supply nets VDD, VDDO, and VSS are required. The power requirements for this design were satisfied using the following pads:

- pvdcnn – VDD
- pvdnpo – VDDO (shorted to VDDP)
- pvscno – VSS (shorted to VSSO)
- pvscnns/pvscnnu – VSS (with power-sequencing protection enabled/disabled)

The order that the I/O and core supplies are powered up, as well as the time between power-up of each of these supplies, is referred to as power-up sequencing. If the I/O supply is powered up first, using VDDP/VDDO, then the output drivers could end up in an indeterminate state, which could cause system problems.

If the pvscnnu pad or the pvscnns pad is implemented, the following power-up sequence is recommended: 5.0V (VGG), then 3.3V (VDDP and VDDO), then 1.8V (VDD).

The supply library includes a feature that prevents power-up sequencing issues, but this feature was disabled, so that the power-up sequence could be observed. The power sequencing
feature also increases static power dissipation. Disabling the power sequencing protection was accomplished by using the pvscnnu pad in this design, as shown in the list of available pads, in Figure 3.4.

```
px00n  32KHz Xtal Oscillator
px01n  Medium Speed Xtal Oscillator (1 MHz to 25 MHz)
plb01n  5V Tolerant (level shifter to core) Bidirectional 120ohm/2mA
plb02n  5V Tolerant (level shifter to core) Bidirectional 80ohm/4mA
plb03n  5V Tolerant (level shifter to core) Bidirectional 50ohm/8mA
plb04n  5V Tolerant (level shifter to core) Bidirectional 32ohm/12mA
pld00n  5V Tolerant (level shifter to core) Input
pld01n  5V Tolerant (level shifter to core) Input w/Repeater
pld10n  5V Tolerant (level shifter to core) Input w/Hysteresis
plt01n  5V Tolerant (level shifter to core) 3-State 120ohm/2mA
plt02n  5V Tolerant (level shifter to core) 3-State 80ohm/4mA
plt03n  5V Tolerant (level shifter to core) 3-State 50ohm/8mA
plt04n  5V Tolerant (level shifter to core) 3-State 32ohm/12mA
ptgcor  Fadring Corner
pvbcnpo  Breaks VDD, VDDP, VDDO, VSSO
pvbcnpo  Breaks VDD
pvbdnpo  Breaks VDDO, VDDP
pvgcpo  Breaks VGG, VDDO, VDDP, VDD
pvgmnn  Breaks VGG
pvdnnn  Connects VDD
pvdnnp  Unconnected / Diode-Protected
pvdnno  Connects VDDO
pvdnpn  Connects VDDP
pvdnpo  Connects VDDP VDDO
pvgnnn  Connects VGG (ESD/Clamping Bus)
pvgnno  Connects VDDO VGG
pvgnpo  Connects VDDP VDDO VGG
pvscnn  Connects VSS
pvscnns  Alternate Core VSS Pad with Power Sequencing Protection
pvscnnu  Alternate Core VSS Pad with Pull-up to Disable Power-sequence
pvssno  Connects VSS, VSSO
pvssnn  Unconnected / Diode-Protected
pvssno  Connects VSSO
```

**Figure 3.13 List of pads available in library**

The list of available pads shows several lines that are highlighted. These lines are the I/O pads that were used in this design. The layout of these I/O pads can be seen in Figure 3.5.
The initial arrangement for the pad ring on the ROSA ASIC is shown in Figure 3.6. The outside dimensions of the die are 4.9mm, which keeps the design within the specified 5.0mm
requirement. There are 104 pads total, with 26 pads on each of the 4 sides of the die. Figure 3.6 also shows the breakdown of the number of power and I/O pads in this design, which are 12 core VDD pads, 8 high voltage VDDO power pads for the I/O, 16 VSS ground pads, and 68 I/O pads. The area inside of the pad ring is 4.47 mm x 4.47 mm.

3.1.2. ROSA SRAM PUF Design

This section will cover the detailed design of the SRAM circuitry on the ROSA PUF ASIC. Figure 3.7 shows the schematic for the SRAM unit cell.

![Separate SRAM Power (V_{DD-SRAM})](image)

**Figure 3.16 SRAM Schematic – Unit cell**

The schematic for the SRAM unit cell shows column and row select lines that feed into AND gates. The critical part of the SRAM unit cell circuit is the section with two back-to-back
inverters that generate the random bit. The output from these inverters goes to a pair of tristate inverters. When the row select and column select lines for a particular unit cell are selected high, the two tristate buffers are enabled, which enables data to be read from the Bit and Bit_Bar lines.

Care was taken to design the circuit so that no systematic bias was introduced, including the addition of the Bit-Bar line, which balances the circuit. This ensures that the output of the SRAM unit cell is completely dependent on very subtle process variations. For comparison, typical SRAM unit cell designs are not concerned with the same symmetry required for SRAM cells used in PUF circuits. This SRAM unit cell layout was carefully hand drawn, instead of using the standard SRAM compiler typically used to design SRAM circuits.

At startup, the SRAM PUF generates data in the back-to-back inverters. This is initiated as the VDD power supply rises. Multiple tests of the SRAM PUF require VDD to fall to zero, then rise again to VDD to repeat the test, which could damage other circuitry on the PUF die. Core VDD rises very slowly, on the order of several millisecond, which is not optimal for the SRAM PUF. For these reasons, we decided to use a separate power supply for the SRAM PUF unit cells, completely apart from the core VDD.

Separating the core VDD and the SRAM PUF VDD enables more reliable testing for the SRAM PUF and protects the other PUF circuits on the die. The power supply for the SRAM PUF is connected to an “Unconnected Pad”, which only has ESD protection devices.

In order to select row and column, a shift register must be used. Figure 3.8 shows the schematic for the SRAM shift register used for both row select and column select.
The schematic for the shift register shows Reset and Clock signals going into the circuit, which is a serial-in, parallel-out (SIPO) shift register (SR). The reset line clears the D flip-flops that make up the shift register. After each clock pulse, the shift register disables the current select line, then enables the next select line. Choosing a certain select line is merely a process of clocking the circuit $x$ number of times, in a shift register with $x$ bits.

Figure 3.9 shows the schematic for a 2x2 SRAM PUF. Designing and simulating a smaller circuit, such as a 2x2 SRAM PUF is important to do before creating the larger, final circuit.
The schematic for the SRAM 2x2 PUF shows the unit cells arranged in a grid, with each of the signal lines connecting to the bus in each row or column, as required. The row and column select bus lines each go to their corresponding shift registers, called row select and column select, respectively.

The layout for the SRAM unit cell, shown in Figure 3.10, shows the power, ground, and signal lines; each are laid out so that the unit cells can be arranged in a grid, taking up the

Figure 3.18 SRAM Schematic – 2x2 PUF
smallest area possible, using available space on the die as efficiently as possible. The size of the SRAM PUF unit cell measure 20.46 µm x 20.46 µm.

**Figure 3.19 SRAM Layout – Unit cell**

The layout for the SRAM 2x2 PUF is shown in Figure 3.11. It was designed so that the power and data lines in both the horizontal and vertical directions mate up perfectly with all of the neighboring cells in the SRAM portion of the circuit.
The floorplan for the completed SRAM on the final die can be seen in Figure 3.12. As you can see, the SRAM circuitry uses up a large percentage of available space on the die. The array of SRAM cells number 150 by 125, which results in 18,750 independently addressable PUF bits. The SRAM PUF circuitry measures 2.6 mm x 3.1 mm on the die.
The test procedure begins with column reset (Col_Reset) and row reset (Row_Reset) being set high, while all other lines begin at the low state. The target pulse width is 1 microsecond, which is also the targeted time between pulses. After a 1 µs wait from the start of the test procedure, both Col_Reset and Row_Reset are pulled low for 1 µs. Next, SRAM_VDD
is enabled, followed by a pulse to the Row_CLK line that sets the shift register to the first row. There are 150 columns, so we next pulse Col_CLK, followed by a pause, then repeat 150 times. This completes the readout of a single row. At this point, Row_CLK is triggered to advance to the next row, and Col_Reset is brought low for 1 µs to reset the column position. This process continues until all 125 rows are read out. This timing diagram for the SRAM PUF can be seen in Figure 3.13, below.

![SRAM PUF Timing Diagram](image)

**Figure 3.22** SRAM PUF Timing Diagram

To repeat the SRAM PUF test, SRAM_VDD is turned off, then started up again once voltage has settled down to zero, which took several seconds due to filter capacitors in the power circuit. The design requirements called out for a SRAM power source that could deliver at least 50 mA of current at +1.8V. The rise time needed to be within 10 ns, without any voltage drop.

It is helpful to simulate the fundamental components in a circuit, such as the SRAM unit cell, to help ensure that these fundamental components can be trusted to function correctly once inserted into a larger section of circuitry. Figure 3.14 shows the T-SPICE simulation for the SRAM unit cell.
The signals t1 through t6 are the probes on each of the transistors in the unit cell. The lines rowsel and colsel are the row select and column select, respectively. The simulation shows that there is no change when row select goes high. However, as soon as the column select line goes high, the circuit is completely activated. At this point, we see that bll, which is “bit line low”, goes high, while blh, which is “bit line high”, goes low, as expected.

Figure 3.23 SRAM Simulation – Unit Cell

Figure 3.15 shows the T-SPICE simulation for the SRAM 2x2 PUF, which uses four of the aforementioned SRAM unit cells. As specified in the timing diagram, column reset and row reset lines are pulled low to reset the shift register. Next, row_clock is pulsed to choose the first row, followed by multiple clock pulses to read out the values at all of the columns in the first row. As you can see, the bit lines, bll and blh, go high and low, respectively, indicating a successful read of the SRAM 2x2 PUF circuit.
3.1.3. ROSA Arbiter PUF Design

The PUF ASIC includes four different types of Arbiter PUFs. There are three Arbiter PUFs that are laid out on the die in a linear manner, which we refer to as distributed. These include 4-stage, 9-stage, and 19-stage Distributed Arbiter PUFs. There is also an Arbiter PUF that is laid out in a square shape on the die, which we refer to as compressed. This is a block of 50 independent, 128-stage Compressed Arbiter PUFs, which takes 6400 challenge bits as inputs, and returns 50 PUF bits as outputs.

The number of stages determines the path length for signals that start at the input trigger. The trigger input is split at the first stage and sent into each of the MUXs that make up the first stage. The outputs from each MUX become the inputs for the next stage of the arbiter. This continues until the last stage, which sends the output from each MUX to a latch circuit. If the
first MUX is faster, then the latch sets “Out1” to be high and “Out2” to be low, otherwise, if the second MUX is faster, then the latch sets “Out1” to be low and “Out2” to be high.

![Programmable Path Diagram]

**Figure 3.25** Schematic of an arbiter PUF with 3-bit challenge bits

The MUXs in the Arbiter PUF are influenced by a shift register with the same number of flip-flops as there are arbiter stages, as shown in Figure 3.16. This allows for each stage in the arbiter to be individually biased by the bits in the shift register. These bits make up the challenge, which produces a single bit, at both “Out1” and “Out2”.

The shift register is loaded at the “Data_in” line, and triggered in the flip-flop by the rising edge of the “CLK_in” line. Multiple challenges can be sent to the same circuit and produce a different output bit for each challenge. There is only one copy of each distributed Arbiter PUF, so the outputs are read directly from Out1 and Out2, since there is no shift register. In order to validate the design, layout was completed for a 5-Stage Arbiter PUF, as shown in Figure 3.17.
Figure 3.26 Arbiter PUF Layout – 5-Stage

Figure 3.27 Arbiter PUF Layout – 5-Stage

Figure 3.18 shows an example of the compressed and distributed Arbiter PUF layouts. In addition to compressed and distributed versions of the Arbiter PUF layouts, copies of the Arbiter PUF circuit were created with a different number of Arbiter stages. There was a single version of compressed Arbiter designed, that has 128 stages. However, there are 50 independent copies of the 128-stage Arbiter on the die. The distributed versions of the Arbiter were designed with 4, 9,
and 19 stages. There are long interconnects in the path of the distributed Arbiter circuits so that we could observe the impact of averaging effects and local variations. The floorplan for the Arbiter can be seen in Figure 3.19.

![Arbiter PUF Floorplan](image)

**Figure 3.28 Arbiter PUF Floorplan**
The compressed 128-bit Arbiter PUF timing diagram, in Figure 3.20, shows that all lines start low, except for Load, which starts high. The timing for Arbiter is similar to SRAM, with a 1 \(\mu\)s pulse width and 1 \(\mu\)s between transitions, including between pulses and across all signals. The compressed 128-stage Arbiter PUF has a shift register that reads in data from Data_In for the challenge for each of the 50 copies of compressed Arbiter. Data is read in for each pulse of CLK_In at the falling edge. There are 128 challenge bits for each Arbiter copy, so 6,400 bits total must be clocked in. The Trig signal goes high after data is read in. The Load signal is pulled low before the first CLK_Out pulse, then goes high before the second CLK_Out pulse. CLK_Out continues pulsing until all 100 responses are read out, which consists of 2 bit values (out1 and out2) for each of the 50 compressed Arbiter PUF circuits. Finally, the Trig signal goes low.
The distributed Arbiter PUFs (4-, 9-, and 19-stage) use a similar timing diagram as the 128-stage Arbiter PUF, with the exception of only having a single value output to Out1 and Out2, which are complements of each other. As mentioned earlier, there are no shift registers on the distributed Arbiter PUF circuits, so values are output soon after Trig goes high, then should be set low once Out1 and Out2 are read. Power cycling between tests is similar to the SRAM recommendations mentioned previously.

<table>
<thead>
<tr>
<th>Arbiter Challenge</th>
<th>Distributed 4-Stage</th>
<th>Distributed 9-Stage</th>
<th>Distributed 19-Stage</th>
<th>Compressed 128-Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0000000000</td>
<td>0000000000000000000</td>
<td>000000000000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>0000000001</td>
<td>0000000000000000001</td>
<td>000000000000000000000001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>0000000010</td>
<td>0000000000000000010</td>
<td>000000000000000000000001</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>0000000111</td>
<td>0000000000000000111</td>
<td>000000000000000000000001</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>00000100</td>
<td>0000000000000100</td>
<td>0000000000000000000100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>00000101</td>
<td>0000000000000101</td>
<td>0000000000000000000101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>00000110</td>
<td>0000000000000110</td>
<td>0000000000000000000110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>00000111</td>
<td>0000000000000111</td>
<td>0000000000000000000111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>101010101010100101</td>
<td>101010101010100101</td>
<td>101010101010100101101</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>010101010101010100</td>
<td>010101010101010100</td>
<td>010101010101010101010</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>111111010111111111</td>
<td>1111111111111111111</td>
<td>111111111111111111110</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>111111111111111111</td>
<td>1111111111111111111</td>
<td>111111111111111111110</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>111111111111111111</td>
<td>1111111111111111111</td>
<td>111111111111111111110</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>111111111111111111</td>
<td>1111111111111111111</td>
<td>111111111111111111110</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>111111111111111111</td>
<td>1111111111111111111</td>
<td>111111111111111111110</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>111111111111111111</td>
<td>1111111111111111111</td>
<td>111111111111111111111</td>
</tr>
</tbody>
</table>

**Figure 3.30 Arbiter PUF Test Parameters**

The challenge inputs for each type of Arbiter PUF are shown in Figure 3.21. The 4-stage Arbiter PUF used every 4-bit binary combination as challenge values. Testing every possible challenge value for the 9-stage Arbiter PUF would have required 512 different challenge strings.
Similarly, testing every possible challenge value for the 19-stage Arbiter PUF would have required 524,288 different challenge strings.

Obviously, it is unreasonable to exhaustively test the Arbiter PUFs with large numbers of possible challenge strings. In order to keep the challenge strings similar, zeros and ones were added to the smaller and larger binary values, respectively. Challenge strings #8 and #9 in the test parameter list show a slight deviation in challenge strings, by using alternating ones and zeros, with one set of strings starting with a zero and the other set of strings starting with a one.

![Figure 3.31 Arbiter PUF 5-Stage with 10fF Capacitor on Output 2a](image)

Figure 3.31 Arbiter PUF 5-Stage with 10fF Capacitor on Output 2a

Figure 3.22 shows the Arbiter PUF, 5-Stage, with a 10fF capacitor on output line 2a. The capacitive values were added to simulate the bias in a manufactured circuit due to process variations. In this simulation, we see that the challenge value ‘00100’ is clocked in, resulting in Out1 being low, while the complementary Out2 value stays high. In the second half of the
simulation, a new challenge value of ‘00011’ is clocked in, resulting in Out1 staying high, while Out2 goes low.

Figure 3.32 Arbiter PUF 5-Stage with 10fF Capacitor on Output 2b

Figure 3.23 is the same SPICE simulation, but with the capacitive value moved to the other output line, 2b. We can see that the entire simulation runs the same as the last simulation, with the exception of the output values being the opposite of the first simulation, due to the change in the simulated capacitive value. These simulations show that the circuit behaves as expected, and increases confidence that the final Arbiter PUF circuits will be as well-behaved as the simulations predict.

3.1.4. ROSA Ring Oscillator PUF Design

The ROSA PUF ASIC includes six different types of RO PUFs. There are two RO PUFs that are laid out on the die in a linear manner, which we refer to as distributed. These include 7-
stage and 11-stage Distributed RO PUFs. There are also four RO PUFs that are laid out in a square shape on the die, which we refer to as compressed. The distributed designs include a 7-Stage, 9-Stage, 11-Stage, and a 13-Stage RO PUF.

Figure 3.24 shows a 5-Stage RO PUF, with a 3-bit counter and a 3-bit shift register. The 20-bit counter and 20-bit shift register in the final design operate in the same manner as the 3-bit versions shown in the schematic, which is shown for simplicity.

![Figure 3.24 Schematic for 5-Stage RO PUF with Counter and Shift Register](image)

The ring oscillator portion of the RO PUF begins with an Enable signal, which feeds in to a NAND gate. The NAND gate contributes to the oscillation of the ring oscillator, and is counted as a stage in this 5-stage RO PUF. The output of the NAND gate feeds in to an inverter, which feeds in to the next inverter, continuing until the final inverter in the chain is reached. At that point, the signal is split between a feedback line that is an input in the NAND at the beginning of the inverter chain, as well as feeding into one of the inputs of the NAND for the counter circuit.
Once the ring oscillator circuit begins oscillating, the counter can begin counting the pulses once the Start/Stop line is enabled. Enabling Start/Stop allows the counter circuit to begin capturing the pulses coming from the ring oscillator. Due to the high speed of the ring oscillator, Start/Stop must not be enabled for long, due to the limitations of the 20-bit counter in the final design. The counter can be reset using the Reset line. Once counting is finished, the count values can be loaded into the shift register, which has the same number of bits available as the counter. Values in the shift register can be read out using the Clock_out line of the shift register. In order to validate the design, layout was completed for a 11-Stage RO PUF, as shown in Figure 3.25.

![11-stage Ring Oscillator](image)

Figure 3.34 RO PUF Layout – 11-Stage

Figure 3.26 shows an example of the compressed and distributed RO PUF layouts. The floorplan for all of the RO PUFs on the die can be seen in Figure 3.27.
Figure 3.35 RO PUF Layout – Compressed and Distributed

Figure 3.36 RO PUF Floorplan
Figure 3.28 shows the timing diagram for the RO PUF. All lines start low, except for Reset, which starts high. The timing for RO is similar to SRAM, with a 1 µs pulse width and 1 µs between transitions, including between pulses and across all signals. The first step is to pull the Enable line high, followed by pulling Reset low for 1 µs. Next, Start/Stop is enabled long enough to receive enough counts to show variation in the count values, but not so long as to overflow the counter. Based on a 2.6GHz maximum oscillation frequency for the 7-Stage RO, feeding into a 20-bit counter, the Start/Stop pulse should be no longer than 400 µs.

![RO PUF Timing Diagram](image)

**Figure 3.37 RO PUF Timing Diagram**

The Load signal should be pulled low before the first CLK_Out rising edge, then set high after the falling edge of the first CLK_Out pulse. Data is available on the Data_Out line beginning with the falling edge of each CLK_Out pulse. Each RO PUF circuit has 65 copies,
with 20-bit counters and shift registers. Therefore, CLK_Out must be clocked 65x20=1,300 times.

![Figure 3.38 RO PUF 7-Stage T-Spice Simulation](image)

Figure 3.38 RO PUF 7-Stage T-Spice Simulation

Figure 3.29 shows the RO PUF, 7-Stage simulation. In this simulation, we see the enable line going high, followed by the reset line getting reset low. Reset goes high again, followed by start/stop going high to start the counter. Oscillations are registered, as seen on bit lines b1 through b20.

Start/stop is then pulled low to stop counting, then load is pulled low first, then set high again around the first clk_out pulse. The clk_out line continues to cycle high and then low until all of the data is read out of the shift register, which is shown on the puf_data line. These simulations show that the circuit behaves as expected, and increases confidence that the final RO PUF circuits will be as well-behaved as the simulations predict.
The complete floorplan of the ROSA PUF ASIC can be seen in Figure 3.30. PUF circuits were arranged so that the entire die could be used, while still including all of the types of PUF circuits required for these experiments.
Figure 3.31 shows the final layout of the ROSA PUF ASIC. This completed layout shows all of the aforementioned PUF circuitry, as well as the completed pad ring structures.

Figure 3.40 ROSA PUF Final Layout
3.1.5 Fabricated ROSA PUF ASIC

Once the die were fabricated, optical inspections were performed on some of the die to evaluate their construction. The packaged die is shown below, in Figure 3.32. In this image, the bond wires and die attach are clearly visible.

![Packaged PUF Die](image)

**Figure 3.41** Packaged PUF Die
Some of the die were delivered as bare die, without any packaging. This allowed for better inspection of the bond pads, without any interference from the wire bonds. Measurements of the bond pads are shown below, including the entire bond pad area, which is covered by the passivation layer, as well as the exposed portion of the bond pad which was not covered by the passivation layer, as shown in Figure 3.33.

**Figure 3.42 Measurements of Bond Pads on Bare Die**

Other features, including the leveling blocks, can be seen in Figure 3.34. The fab requested that additional leveling blocks be added, in order to help preserve the structure of the die during the chemical mechanical polishing (CMP) process during fabrication.
Figure 3.43 Close-Up of Top of Die, including Leveling Blocks

Figure 3.35 shows a close-up of the bond pads of a packaged die, including a bond pad in the top-center of the image that has damage, possibly from the wire bonding process.
3.2. **ROSA PUF Test Setup**

The experimental setup consists of the ZedBoard, which is the experiment board that the FPGA runs on, and the computer that will be used for processing the data generated by the Exhaustive PUF System.

3.2.1. **Printed Circuit Boards**
There are three custom printed circuit boards (PCBs) connected to the Zedboard, as shown in Figure 3.36. These boards include the main PUF Breakout PCB, the PUF Power PCB, and the PUF UART PCB. The design and functionality of each of these boards will be detailed in this section.

**Figure 3.45 Complete PUF Test Setup**

Breaking out the PUF ASIC into headers for testing was made possible with the main PUF breakout PCB. The schematic for the PUF breakout PCB is shown in Figure 3.37. The schematic shows the Pin Grid Array (PGA) ZIF socket component at the center, which only uses 104 of the available 224 pins. Signals for the unconnected pins are shown in the center of the PGA component. Each side of the die has 26 pins, so the breakout PCB copied this layout, with each side of the package routed to 26-pin headers on all four sides of the ZIF socket. Between the socket and the headers are placeholders for capacitors. Only the power lines require
capacitors, however, pads were added to each line to give access to probes, as well as to remedy any potential packaging issues where pins are not in the expected location.

Figure 3.46 Schematic for Main PUF Breakout PCB
Figure 3.47 Layout for Main PUF Breakout PCB

Figure 3.38 shows the layout for the main PUF breakout PCB. All routing was run by hand, and was completed using two layers. This layout does not include the grounded copper pour.
Figure 3.39 shows the same layout for the main PUF breakout PCB, with copper pour rendered. Care was taken in the routing during board layout to minimize areas that are not surrounded by the grounded copper pour.
Figure 3.40 shows the top side of the bare main PUF breakout PCB. The capacitor pads closest to the ZIF socket are connected to the power lines. The other pads are used as test points.
Figure 3.41 shows the bottom side of the main PUF breakout PCB. Lines routed on the bottom layer were used when space on the top side did not allow for successful routing, due to
the large number of power and I/O lines. Vias were placed liberally throughout the design to improve electromagnetic interference (EMI) characteristics [32].

![Figure 3.51 PGA Socket Used For ROSA PUF ASIC](image)

The PUF IC was packaged using a custom, 104-pin PGA package. The use of this package required a custom ZIF socket, shown in Figure 3.42. The ZIF socket was convenient, because it allowed for the packaged PUF ICs to be swapped in and out throughout the testing process, including testing each die in the set, for each type of PUF circuit on the PUF IC.
The final, populated main PUF Breakout PCB is shown in Figure 3.43. When inputs are not in use, they must be connected to ground. To make this process simpler, ground points were added along the outer headers for each PUF IC pin. The ground headers were placed 0.1” away from the other set of headers. With this in place, standard 0.1” jumpers could be used to connect each input from the PUF IC to its corresponding ground pin on the ground header.
The PUF Power PCB schematic is shown in Figure 3.44. The Power PCB takes power input from a DC barrel jack, which comes from a wall-type power supply that has an output of 9VDC. This power is then sent to three separate voltage regulators. One regulator takes the 9VDC input and outputs 3.3VDC. The two other voltage regulators both have outputs of 1.8VDC. The 3.3V output is used for the pad ring supply, while one 1.8V output is used for Core voltage, and the other 1.8V output is used for SRAM VDD.

There are also three logic inputs on the board that are used to enable each of the regulators. The enable lines for the voltage regulators are enabled with logic low, and disabled with logic high. Once enabled, the output voltage is available at the three voltage output pins.
The layout for the Power PCB, in Figure 3.45, shows mounting holes that were added to the PCB design so that standoffs could be used. Standoffs raise the PCB off the testing surface, which can help prevent accidental shorts on the bottom of the PCB while in operation.
Figure 3.55 Layout for Power PCB with Copper Pour

Figure 3.46 shows the same layout for the Power PCB, with the addition of a grounded copper pour layer. Care was taken to avoid sharp corners between components on the PCB.
Figure 3.47 shows the top side of the bare PUF Power PCB. The DC power jack, DCJ0202, was placed at the edge of the PCB to allow for easy insertion and removal.
Figure 3.48 shows the bottom side of the bare PUF Power PCB. As you can see in the image, there were only two signals that needed to be routed on the bottom side of the PCB.
Figure 3.49 shows the Power PCB after it was populated with components, including 9 capacitors, 3 voltage regulators, a DC power jack, and a double-row 0.1” header. Standoffs were added to raise the PCB off of the testing surface.
Figure 3.59 Schematic for PMOD-to-Terminal PCB

Figure 3.50 shows the schematic for the PMOD-to-Terminal PCB, which was used to have reliable, bare-wire connections for signals used by the FPGA. Access to the PMOD is available through the FPGA’s Programmable Logic (PL), which is very beneficial for applications requiring general purpose input output (GPIO) connections.

Figure 3.51 shows the layout for the PMOD-to-Terminal PCB without grounded copper pour on the left, and with grounded copper pour on the right. Figure 3.52 shows the top of the PMOD-to-Terminal PCB on the left, and the bottom side of the PCB on the right.
Figure 3.60 Layouts of PMOD-to-Terminal PCB - With and Without Copper Pour
Figure 3.61 PMOD-to-Terminal PCB - Top and Bottom
The main purpose of the PUF PMOD-to-Terminal breakout PCB, shown in Figure 3.53, is to connect the male pins from the USB-to-Serial cable, which is connected to the PC, to the PMOD lines on the Zedboard in order to implement a serial port that can be used for communication between the PC and the state machine running on the FPGA of the Zedboard. The board has a male header that mates up to the female header on the PMOD socket of the ZedBoard. The PCB was designed to be thin enough to have multiple PMOD PCBs connected at the same time on the ZedBoard. Connections between each of the PCBs was done using standard male-to-male, male-to-female, and female-to-female style jumper wires, as required.

Figure 3.62 PMOD-to-Terminal PCB in Use
3.2.2. ZedBoard

The ZedBoard, shown in Figure 3.54, is an evaluation and development board that comes from a collaboration between Xilinx, Avnet, and Digilent.

![ZedBoard Image]

**Figure 3.63 ZedBoard – Top**

The ZedBoard has five PMOD connectors, two of which were used for this research. Switches and buttons were used as well for initial Verilog development and testing.
Figure 3.64 ZedBoard - Bottom

The bottom of the ZedBoard, where the SD card connector is located, is shown in Figure 3.55. The ZedBoard is a great value, with a starting price of $475, and includes a 4 GB SD card. Other features of the ZedBoard include 512 MB DDR3, 256 Mb Quad-SPI flash, onboard USB-JTAG, 10/100/1000 Ethernet, USB OTG 2.0, USB-UART, PS and PL I/O expansion via the various ports (PMOD, FMC, XADC), multiple displays (1080p HDMI, 8-bit VGA, 128x32 OLED), and an I2S audio codec.
The ZedBoard uses a heatsink to keep the FPGA cool, which can be seen in Figure 3.56. The FPGA, below the heatsink, is a Xilinx Zynq XC7Z020 SoC Series device. The 7020 version has dual-core ARM A9 processors that can run at 667 MHz. This device also has 53,200 LUTs, 106,400 flip-flops, 4.9 Mb of block RAM, and 220 DSP slices. This board is targeted at the education market as an affordable option for students who want to learn embedded programming. Even though the ZedBoard is affordable, it still has plenty of processing power on the Zynq chip, with numerous peripherals that make for a great foundation for research experiments, such as the ROSA PUF ASIC.
The PC connects to the FPGA via one of the PMOD connectors. There is a built-in UART connection on the ZedBoard, used by the Processing System (PS), which can be seen in Figure 3.57, at the top of the board, near the JTAG connector.

### 3.2.3. Personal Computer

Most modern personal computers (PCs) are capable of running the software that connects to the FPGA over UART, which runs on Windows. An available USB port for UART communications is also required.
4.PUF SOFTWARE

“Be a yardstick of quality. Some people aren’t used to an environment where excellence is expected.”
- Steve Jobs

This section details the software used for system interfacing, as well as for data analysis.
This chapter details software development platform choice, as well as changes that were made, due to unreliable performance with serial port communication. Matlab was used heavily for data analysis.

4.1.Interface Software

Software engineering began with Java for the interface software used to collect data. Java’s serial port libraries were very limited at the time of this research, and had poor performance, especially compared to the performance I had seen in the past with C# serial port code. The data collection software is one of the crucial elements of this test setup, so any reliability issues are not acceptable.

A new interface software was created using C# and had much improved data collection, without the dropped bits experienced with Java. With the Java serial port library, options for serial port communication were very limited. The freshly updated C# serial port libraries had more options than were needed for this project, making software development much easier.
The PUF Data Application, shown in Figure 4.1, which was written in C#, is shown in the state when the program is freshly opened. Available communication ports are searched, then added to the “Ports” list. All other drop-down menus include typical values needed for serial port communication, with the typical values set as default.

The operation of the software is fairly straightforward. Once all the parameters are set, the “Start Test” button is pressed, which runs the tests for the set number of iterations. Text files
are written, capturing the settings used for each test, as well as the raw data used for later analysis.

![Figure 4.68 C# PUF Data Application - PUF Types](image)

Next, the “PUF Type” drop-down list that shows all of the available PUF experiments that can run from this software is shown in Figure 4.2.

### 4.2. Data Processing

These experiments generated large amounts of data. In order to process the data, both C# and Matlab were used, as detailed in the following sections. The software used for data processing evolved as the project progressed.

#### 4.2.1. C#

The C# development environment used was Microsoft Visual Studio. Visual Studio is a very mature product, with many features that are useful for development, debugging, and distribution. Data processing started with C#, however, after experimenting with Matlab, it was
obvious that the toolboxes and functions available in Matlab would make data processing much more straight forward. In addition, Matlab software is tested and qualified by quality engineers at MathWorks, to give customers confidence in the analysis results.

4.2.2. Matlab

Matlab software was first commercially release by MathWorks in 1984. Before that, it was developed at the University of New Mexico, under Professor Cleve Moler, the chairman of the Computer Science department, in the late 1970s. Since then, Matlab has grown to include leading edge tool boxes, including advanced neural network and machine vision tool boxes, as well as great improvements in the graphical programming environment called Simulink.

For this research, Matlab was used heavily for performing calculations on data, as well as plotting the results. The transition from C# to Matlab for data analysis resulted in time savings and increased confidence, since custom C# data analysis code needed to be tested thoroughly before the results could be trusted. Matlab was also used by a colleague at Honeywell FM&T, who followed the NIST standards outlined in the next chapter, to create functions that would process binary data, including PUF data.
5. PUF METRICS

“I’ve missed more than 9,000 shots in my career. I’ve lost almost 300 games. 26 times, I’ve been trusted to take the game winning shot and missed. I’ve failed over and over and over again in life. And that is why I succeed.”

- Michael Jordan

This chapter covers the metrics used to quantify the performance of each PUF circuit. The metrics used for this research include the NIST Tests, which consist of the Frequency (Monobit) Test, as well as the Frequency Test within a Block Test. Inter-Die Hamming Distance and Intra-Die Hamming Distance were also used, and will be explained below, along with the other metric methods.

NIST (National Institute of Standards and Technology) has a Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications that uses standard algorithms to evaluate a PUF circuit’s response. These standards come from the NIST Special Publication (SP) 800-22, which gives background information on random number testing, as well as detailed explanations of the algorithms used for each type of test [33].

5.1. NIST – Frequency (Monobit) Test

The NIST Frequency (Monobit) Test is used to evaluate a binary string to determine whether or not the string should be considered random. NIST Special Publication 800-22, mentioned earlier, explains this test in great detail. In order to determine randomness, the
algorithm tests for the proportion of zeros and ones for the entire bit sequence. Specifically, it checks how close the fraction of ones is to 0.5.

The calculated value used as a final metric for this test is called the P-value, which is frequently called the “tail probability”. The calculations for the P-value are shown below, in Figure 5.1.

\[
\varepsilon_{conv}(x) = \begin{cases} 
-1, & x = 0 \\
1, & x = 1
\end{cases} 
\]  
(1)

\[
S_n = \sum_{i=1}^{n}(\varepsilon_{conv})
\]  
(2)

\[
S_{obs} = \frac{|S_n|}{\sqrt{n}}
\]  
(3)

\[
P - value = \text{erfc}\left(\frac{S_{obs}}{\sqrt{2}}\right) = \frac{2}{\sqrt{\pi}} \int_{S_{obs}/\sqrt{2}}^{\infty} e^{-t^2} dt
\]  
(4)

**Figure 5.69 NIST Frequency (Monobit) Test Calculation**

The P-value calculation starts with a conversion of the bit string values, \(\varepsilon\), as shown in Figure 5.1, Equation 1. Next, all bits in sequence \(\varepsilon\) are summed, as shown in Equation 2. The test statistic is calculated in Equation 3, by dividing the absolute value of the sum by the square root of the sequence length. The P-value is calculated as shown in Equation 4.

NIST recommends using a threshold of 0.01. If the computed P-value is less than the threshold value, then the sequence is considered non-random. NIST also recommends \(n\) to be greater than or equal to 100. An example calculation of this test is shown in Figure 5.2.
**Example:** \( e=1100110101, \; n=10 \)

\( S_n = 1+1+(-1)+(-1)+1+1+(-1)+1+(-1)+1 = 2 \)

\[ S_{obs} = \frac{|S_n|}{\sqrt{n}} = \frac{|2|}{\sqrt{10}} = 0.632455532 \]

\[ p-value = \text{erfc} \left( \frac{0.632455532}{\sqrt{2}} \right) = 0.527089 \]

*Figure 5.70 NIST Frequency Monobit Example*

The example begins with a 10-element bit string. The sum of all bits in the sequence is calculated to be 2. The sum, and number of elements are used to calculate the test statistic, which evaluates to approximately 0.63. This value is used in the final calculation of the P-value, which evaluates to 0.527089. Based on a threshold of 0.01 (1%), this sequence is considered random.

\( S_n \) values that are large and negative indicate that there are too many zeros. \( S_n \) values that are large and positive indicate that there are too many ones. NIST offers software to calculate these values. However, the software only runs on a Linux environment, so Windows users familiar with computational programs, like Matlab, may choose to implement their own analysis routines.

**5.2. NIST – Frequency within a Block Test**

NIST uses a test similar to the previous test, with a focus on smaller blocks of data, called the Frequency within a Block Test. Used to evaluate a binary string for randomness, this test
takes a binary string as input, and tests for the proportions of ones with M-bit blocks. This test is specifically searching for significant runs of zeros or ones.

This test is well defined in the NIST 800-22 special publication, including all algorithms and functions used for calculations, including the Incomplete Gamma Function, shown in Figure 5.3, which is used for this test.

\[
\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} \, dt
\]

\textbf{Incomplete Gamma Function}

\[
P(a, x) \equiv \frac{\gamma(a, x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} \, dt
\]

where \( P(a, 0) = 0 \) and \( P(a, \infty) = 1 \).

\textbf{Incomplete Gamma Function}

\[
Q(a, x) \equiv 1 - P(a, x) \equiv \frac{\Gamma(a, x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} t^{a-1} \, dt
\]

where \( Q(a, 0) = 1 \) and \( Q(a, \infty) = 0 \).

\textit{Figure 5.71 Incomplete Gamma Function}
The calculations for the Frequency within a Block Test can be seen in Figure 5.4.

Equation 5 starts by with N, which is the number of overlapping blocks, and is calculated by dividing the number of bits in the string, n, by the block length, M. Equation 6 shows the calculation for the lower-limit bounds for the integral. Equation 7 shows the calculation for the gamma function. Finally, Equation 8 shows the calculation of the P-value.

Like the previous test, the decision threshold for this test is 0.01 (1%). If the P-value is less than the threshold value, then the binary sequence is considered non-random. Recommended values for this test by NIST included n being greater than or equal to 100, M being greater than or equal to 20, M being greater than 0.1n, and N being less than 100.
5.3. Hamming Distance

Hamming distance (HD) is a measure of the number of positions that are different between two binary numbers. An example of a basic hamming distance calculation can be seen in Figure 5.5.

- **Example**
  - The decimal value for 8 in binary is 1000
  - The decimal value for 3 in binary is 0011
  - Bits 3 (1 and 0) are different, so we increment our HD value to 1
  - Bits 2 (0 and 0) are not different, so our HD value stays at 1
  - Bits 1 (0 and 1) are different, so we increment our HD value to 2
  - Bits 0 (0 and 1) are different, so we increment our HD value to 3
  - Therefore, the hamming distance of 8 and 3 is 3

**Figure 5.73 Hamming Distance Example**

The hamming distance example starts with two decimal values, 8 and 3, whose binary equivalent values are 1000 and 0011, respectively. There are four bit positions in these strings, which can be represented as b3, b2, b1, and b0, which is the least significant bit (LSB). The hamming distance calculation starts with b3, the most significant bit (MSB). Since the MSB values of these bits are 1 and 0, we increment the hamming distance value to 1. The bits in position b2 are the same, so there is no change to the HD value. The final bits, b1 and b0 are both
different from each other, so the HD value is incremented for each case, bringing the total hamming distance value to 3.

5.4. Intra-Die Hamming Distance

Intra-die hamming distance is the measure of the pairwise distance between two binary strings generated by the same PUF, using the same challenge. The calculation for intra-die hamming distance can be seen in Figure 5.6.

$$Intra\ HD = \frac{1}{k} \sum_{i=1}^{k} \frac{HD(R_{i,1}, R_{i,2})}{n} \times 100\%$$ \hspace{1cm} (9)

Figure 5.74 Intra-Die Hamming Distance Calculation

Variables used in the intra-die HD calculation include $k$, which represents the number of die, $R_{i,1}$, which represents the response from die ‘i’ for the first challenge, $R_{i,2}$, which represents the response from die ‘i’ for the second challenge, and $n$, which represents the number of bits in the PUF response. Since we want the PUF to reliably output the same bit string, the ideal distance between each string is 0.

5.5. Inter-Die Hamming Distance

Inter-die hamming distance is the measure of the pairwise distance between two binary strings generated by different PUF die, using the same challenge. The calculation for inter-die hamming distance can be seen in Figure 5.7.
\[ \text{Inter } HD = \frac{2}{k(k-1)} \sum_{i}^{k-1} \sum_{j=i+1}^{k} \frac{\text{HD}(R_i,R_j)}{n} \times 100\% \quad (10) \]

**Figure 5.75 Inter-Die Hamming Distance Calculation**

Variables used in the inter-die HD calculation include \( k \), which represents the number of die, \( R_i \), which represents the response from die ‘i’ for the challenge, \( R_j \), which represents the response from die ‘j’ for the same challenge that die ‘i’ received, and \( n \), which represents the number of bits in the PUF response. Since we want the PUF to output a different bit string, the ideal distance between each string is 50%.

**5.6. Bit Flipping**

Bit flips, defined earlier, happen when a response that is typically a zero or one, flips to the opposite value, becoming a one or zero. This metric is important when comparing PUFs because it shows a PUF circuit’s reliability. Often times, multiple tests must be performed on a PUF to gather enough data to show whether or not bit values stabilize as testing iterations increase. The results of these experiments will be shown in the next chapter.
6. RESULTS

“The most exciting phrase to hear in science, the one that heralds new discoveries, is not Eureka!’ but ‘That’s funny…””

- Isaac Asimov

This section details the results from experiments run on all of the PUF circuits on the die, including SRAM PUF, multiple Arbiter PUFs, and multiple RO PUFs. There were 70 packaged die used in these experiments.

6.1. SRAM PUF Results

This section contains results for the SRAM PUF tests, including NIST Frequency (Monobit), NIST Frequency Test within a Block, Intra-Die Hamming Distance, Inter-Die Hamming Distance, and a table summarizing all of the results.

The results from the SRAM PUF can be seen below, in Table 6.1. The results include the NIST Frequency (Monobit) Test percentage of passing, average P-value for that test, the NIST Frequency within a Block Test percentage of passing, the average P-value for that test, and the mean intra-die hamming distance and inter-die hamming distance.

<table>
<thead>
<tr>
<th>NIST Freq. (Monobit)</th>
<th>Average P-Value Test #1</th>
<th>NIST Freq. within a Block</th>
<th>Average P-Value Test #2</th>
<th>Mean Intra-Die HD</th>
<th>Mean Inter-Die HD</th>
</tr>
</thead>
</table>
Table 6.1 SRAM PUF Results

The passing rate of the first NIST test, called the Frequency (Monobit) test, was very high at 99.4%. The associated average p-value for this test was 0.5955, which is much closer to the maximum value of 1.0 than the threshold value of 0.01.

Figure 6.76 SRAM PUF NIST Frequency (Monobit) Test
Figure 6.1 shows the counts per bin for the calculated p-values for the Frequency (Monobit) test. The curve that was fit to the data shows that the p-values favored the upper end of the range, close to the ideal value of 1.0. P-values at the other end of the range were generally above the 0.01 threshold, with the exception of 0.6% of data sets tested.

The passing rate of the second NIST test, called the Frequency within a Block test, was 99.5%, which is even higher than the first test. The average p-value for this test was 0.6732, which is, again, even higher than the first test. Figure 6.2 shows the counts per bin for the calculated p-values for the Frequency within a Block test. Again, most p-values favored the upper end of the range, with all p-values staying above the threshold, with the exception of 0.5% of data sets tested.
Figure 6.77 SRAM PUF NIST Frequency Test within a Block
From Table 6.1, we see that the mean intra-die hamming distance for the SRAM PUF was 3.4%. We can see how each die tested performed in Figure 6.3, which shows most values below 0.05 (5%), and even several approaching the ideal value of 0.

Figure 6.78 SRAM PUF Intra-Die Hamming Distance
SRAM PUF mean inter-die hamming distance was 48.1%, which is very close to the ideal value of 50%. Figure 6.4 shows the counts for the SRAM PUF, with an excellent bell curve distribution centered close to the target 50%. Outlier values were at 43% on the low end, and 58% on the high end.
6.2. Arbiter PUF Results

This section details the results for all of the Arbiter PUF circuits, including the compressed 128-Stage Arbiter PUF, as well as the distributed 4-Stage, 9-Stage, and 19-Stage Arbiter PUFs.

6.2.1. Distributed 4-Stage Arbiter PUF Results

The results from the distributed 4-stage Arbiter PUF can be seen below, in Table 6.2. The first NIST test for Frequency (Monobit) had a respectable 91.1% passing rate. The second NIST test for Frequency within a Block had great results, with a 99.5% passing rate. Again, those data sets with P-values below the threshold did not pass these tests, and are not considered random. The mean intra-die hamming distance was 22.6% for the 4-stage Arbiter PUF, while the mean inter-die hamming distance was 50.7%.

<table>
<thead>
<tr>
<th>NIST Freq. (Monobit)</th>
<th>NIST Freq. within a Block</th>
<th>Mean Intra-Die HD</th>
<th>Mean Inter-Die HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>91.1%</td>
<td>99.5%</td>
<td>22.6%</td>
<td>50.7%</td>
</tr>
</tbody>
</table>

Table 6.2 Arbiter PUF 4-Stage Distributed Results
Figure 6.80 Distributed Arbiter 4-Stage Bit Flipping - Averages

Figure 6.5 shows the average response bit values for each of the 16 challenge strings. These tests were performed repeatedly, for 10,000 iterations. We can see that the averages approach a stable value near iteration 1,000.

This data shows that most of the responses gravitate towards either zero or one, with several others favoring one end of the range. However, there are two average values that start to approach the 0.5 bit value mark, with a near perfect split between resulting in a zero or one, even out to 10,000 iterations.
Figure 6.81 Distributed Arbiter 4-Stage - Number of Bit Flips Total

Figure 6.6 shows an analysis unique to the 4-stage Arbiter, which uses cumulative values for each bit flip to show the data in a different way. In this analysis, those responses with fewer bit flips after 10,000 iterations, grouped at the bottom of the chart, are the most stable. Those responses towards the top of the chart lack the stability to be used in a system requiring response consistency.
6.2.2. Distributed 9-Stage Arbiter PUF Results

The results from the distributed 9-stage Arbiter PUF can be seen below, in Table 6.3. The first NIST test for Frequency (Monobit) had a decent 97.1% passing rate. The second NIST test for Frequency within a Block had excellent results, with a near perfect 99.9% passing rate. Again, those data sets with P-values below the threshold did not pass these tests, and are not considered random. The mean intra-die hamming distance was 36.7% for the 9-stage Arbiter PUF, while the mean inter-die hamming distance was an excellent 49.9%.

<table>
<thead>
<tr>
<th>NIST Freq. (Monobit)</th>
<th>NIST Freq. within a Block</th>
<th>Mean Intra-Die HD</th>
<th>Mean Inter-Die HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>97.1%</td>
<td>99.9%</td>
<td>36.7%</td>
<td>49.9%</td>
</tr>
</tbody>
</table>

Table 6.3 Arbiter PUF 9-Stage Distributed Results
Figure 6.82 Distributed Arbiter 9-Stage Bit Flipping - Averages

Figure 6.7 shows the average response bit values for each of the 16 challenge strings. These tests were performed repeatedly, for 10,000 iterations. We can see that the averages approach a stable value near iteration 3,000. In this case, the data shows that bit flipping is increased compared to the 4-stage Arbiter, which had most of the responses gravitate towards either zero or one. Here, many of the average values start to approach the 0.5 bit value mark, with some of the average values gravitating towards zero, with very few at the top of the chart towards one.
6.2.3. Distributed 19-Stage Arbiter PUF Results

The results from the distributed 19-stage Arbiter PUF can be seen below, in Table 6.4. The first NIST test for Frequency (Monobit) had an excellent 99.8% passing rate. The second NIST test for Frequency within a Block had excellent results, with a near perfect 99.9% passing rate. The mean intra-die hamming distance was a respectable 16.2% for the 19-stage Arbiter PUF, while the mean inter-die hamming distance was an excellent 49.7%.

<table>
<thead>
<tr>
<th></th>
<th>NIST Freq. (Monobit)</th>
<th>NIST Freq. within a Block</th>
<th>Mean Intra-Die HD</th>
<th>Mean Inter-Die HD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>99.8%</td>
<td>99.9%</td>
<td>16.2%</td>
<td>49.7%</td>
</tr>
</tbody>
</table>

Table 6.4 Arbiter PUF 19-Stage Distributed Results
Figure 6.83 Distributed Arbiter 19-Stage Bit Flipping - Averages

Figure 6.8 shows the average response bit values for each of the 16 challenge strings. These tests were performed repeatedly, for 10,000 iterations. We can see that the averages approach a stable value near iteration 2,000, with the exception of three of the average values near the 0.5 bit value mark. This data for the 19-stage Arbiter is similar to the 4-stage Arbiter, with most of the responses gravitating towards either zero or one.
6.2.4. Compressed 128-Stage Arbiter PUF Results

The results from the compressed 128-stage Arbiter PUF can be seen below, in Table 6.5. The first NIST test for Frequency (Monobit) had an 83.3% passing rate. The second NIST test for Frequency within a Block had a slightly higher 87.0% passing rate. The mean intra-die hamming distance was a good, at 2.6% for the 128-stage Arbiter PUF, while the mean inter-die hamming distance was 47.5%.

<table>
<thead>
<tr>
<th>NIST Freq. (Monobit)</th>
<th>NIST Freq. within a Block</th>
<th>Mean Intra-Die HD</th>
<th>Mean Inter-Die HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>83.3%</td>
<td>87.0%</td>
<td>2.6%</td>
<td>47.5%</td>
</tr>
</tbody>
</table>

Table 6.5 Arbiter PUF 128-Stage Compressed Results
6.3. Ring Oscillator PUF Results

The ROSA PUF ASIC has seven different Ring Oscillator PUF types, including compressed 7-stage, 9-stage, 11-stage, and 13-stage RO PUFs. Distributed 7-stage and 11-stage RO PUFs complete the list of available RO PUFs. In order to perform a better comparison, the 7-stage and 13-stage RO PUFs, in both compressed and distributed form, were going to be used to contribute data to answer the question about whether compressed or distributed layouts resulted in better PUF circuits.

Data collection started with the 7-stage Compressed RO PUF, resulting in data sets with reasonable values. Data collection continued with the 7-stage Distributed, or Spread, RO PUF. Unlike the 7-stage Compressed RO PUF, the 7-stage Distributed RO PUF was outputting nearly identical values, no matter what start/stop time was used. Whether the start/stop time was set to the fastest pulse that the FPGA could produce (sub-microsecond), or increased to several seconds, the values did not change. By not being able to increase start/stop times, many of the responses from the RO PUF were the same, since the counter was designed to capture much longer start/stop times, and did not have the resolution for these small values (often times all below 1023). This meant that the data could not be trusted, or used for analysis.

Revisiting the 7-stage Compressed RO PUF tested before, I found that the responses from that PUF changed, corresponding to changes in start/stop times. With a very short start/stop time, the values would be very small, typically below 100. With a very long start/stop time, the 20-bit
counter was maxed out, resulting in values near 1,048,575, which is the largest value that can be represented with 20 bits.

Unwilling to accept defeat, I began capturing waveforms using the best oscilloscopes and digitizers I could get my hands on at Honeywell FM&T. I pored over the FPGA Verilog, the UART code, and even the C# code used for data capture. Each of these components were reviewed, line by line, and refactored anywhere that there was any possibility for ambiguity, or potential for miscalculation. Despite these efforts, I was getting the same results.

At this point, I moved on to the 11-stage RO PUF, with the intention of revisiting the 7-stage RO PUF once this data was collected. As I began to collect both distributed and compressed 11-stage RO PUF data, I was shocked to find that I had the opposite problem. The problem before had been not able to change start/stop times, resulting in small values. This time, I was overflowing the counter, with most of the values being 1,048,575, the maximum value for the 20-bit counter.

Without good data from the 7-stage or 11-stage RO PUFs, there could be no comparison between distributed and compressed. Not wanting to leave readers without any data, a 3D surface plot of 7-stage Distributed RO PUF counter values, taken over 1,000 test iterations. As you can see in Figure 6.9, the first half of the 65 ring oscillators have very low values in each iteration, with an exponential increase in count values from the last ring oscillators. Figure 6.10 shows a subset of this data, taken over 100 test iterations.
Figure 6.84 RO PUF - 7-Stage Distributed - 3D Surface Plot

Figure 6.85 RO PUF - 7-Stage Distributed - Heat Map
7. CONCLUSION

“You are never too old to set another goal or to dream a new dream.”

- C.S. Lewis

In conclusion, both SRAM PUF and Arbiter PUFs operated as expected. The RO PUF issues could be a result of issues in the data capture software, hardware or software UART, PCBs, die packaging, or even the ASIC design. Because of the extensive simulations and care taken in the design, the ASIC is not likely the component causing problems in the system. Circuit board implementation was straightforward, but could still be a problem source. Verilog and PC software are another potential source for why the RO PUF is not operating as expected.

Arbiter PUF data showed strong performance in the NIST and hamming distance tests. Figure 7.1 shows the metrics for all four Arbiter PUFs. Beginning with the NIST tests, the distributed Arbiter PUFs performed much better than the compressed Arbiter PUF, with some values approaching 100%.
<table>
<thead>
<tr>
<th>Arbiters</th>
<th>NIST Freq. (Monobit)</th>
<th>NIST Freq. within a Block</th>
<th>Mean Intra-Die HD</th>
<th>Mean Inter-Die HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-Stage Distributed</td>
<td>91.1%</td>
<td>99.5%</td>
<td>22.6%</td>
<td>50.7%</td>
</tr>
<tr>
<td>9-Stage Distributed</td>
<td>97.1%</td>
<td>99.9%</td>
<td>36.7%</td>
<td>49.9%</td>
</tr>
<tr>
<td>19-Stage Distributed</td>
<td>99.8%</td>
<td>99.9%</td>
<td>16.2%</td>
<td>49.7%</td>
</tr>
<tr>
<td>128-Stage Compressed</td>
<td>83.3%</td>
<td>87.0%</td>
<td>2.6%</td>
<td>47.5%</td>
</tr>
</tbody>
</table>

**Figure 7.86** Arbiter PUF Distributed and Compressed Layout Metrics

Mean intra-die hamming distance was surprisingly better for the compressed Arbiter PUF, with the 9-stage distributed Arbiter PUF showing poor results in this test. Mean inter-die hamming distance, however, showed a clear advantage with the distributed layout over the compressed Arbiter layout.

Three out of the four metrics used in this research point towards the distributed Arbiter layout as being superior to the compressed layout. These results are yet another data point in this research area. Hopefully, this research is helpful to other scientists and engineers who are working towards advancing the field of Physically Unclonable Functions.
Notice:

- This work is funded by the Department of Energy’s Kansas City National Security Campus, operated by Honeywell Federal Manufacturing & Technologies, LLC under contract number DE-NA0002839.
REFERENCES


