6-5-2020

Delay-based Physical Unclonable Function Implementation

Abigail Aguirre

Michael Hall

Timothy Lim

Jonathan Trinh

SANTA CLARA UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Date: June 5, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Abigail Aguirre
Michael Hall
Timothy Lim
Jonathan Trinh

ENTITLED

Delay-based Physical Unclonable Function Implementation

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN ELECTRICAL ENGINEERING

Fatemeh Tehranipoor

Thesis Advisor

Shoba Krishnan (Jun 11, 2020 12:57 PDT)

Department Chair

Delay-based Physical Unclonable Function Implementation

by

Abigail Aguirre
Michael Hall
Timothy Lim
Jonathan Trinh

Senior Design Project Report

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Electrical Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 5, 2020

Delay-based Physical Unclonable Function Implementation

Abigail Aguirre
Michael Hall
Timothy Lim
Jonathan Trinh

Department of Electrical and Computer Engineering
Santa Clara University
June 5, 2020

ABSTRACT

As we venture further into the 21st century, it becomes much clearer that hardware security is at the forefront of many challenges that we face today in ensuring that data is protected. "Keys" (a sequence of bits) can be used to unlock pieces of data and is a concept that is pervasive throughout cryptography, but storage in memory makes this sole method nonviable. To make the approach more practical, one can dynamically generate a key through a Physical Unclonable Function (PUF). PUFs are circuit primitives that use intrinsic variations of microchips created during the manufacturing process to generate a unique "fingerprint" for each chip. We simulated several different PUF designs on a Field Programmable Gate Array (FPGA) board to determine how changes to a starting design can affect the reliability, randomness, and uniqueness of these IDs. We propose two schemes, a parallel and a serial scheme for a ring oscillator (RO) based PUF. The parallel scheme is a useful benchmark for other designs, and the serial scheme uses much less hardware than other RO PUF designs. The serial scheme is not as random, reliable, or unique as the parallel scheme, but it creates input-output pairs with much less area.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Hardware security provides protection against attacks on the physical side of technology. Consequently, hardware security primitives are strong circuit building blocks that prevent hardware attacks. Designers need lightweight and cost-effective methods to protect and verify their circuit's integrity. Physical Unclonable Functions (PUFs) are a robust solution to combat the aforementioned issues. The fundamental concept that PUFs rely on happens at the manufacturing step. In manufacturing, it is impossible for all processes to be applied uniformly across the entire wafer, which leads to some small random variations in all Integrated Circuits (ICs). PUFs use these random attributes introduced during manufacturing to generate unique IDs and keys for authentication. Since process variation is unavoidable and uncontrollable, a PUF implements a function unknown to both the designer and the attacker, and this function differs between chips. While two chips that come off the End of Line (EoL) may be functionally equivalent, they will have physical characteristics that are different (i.e. Load/Line Capacitance, parasitic, etc.) as a result of process variations. These physical characteristics can be used to uniquely characterize each IC that comes off the line. PUF input-output pairs are referred to as challenge-response pairs (CRPs).

A ring oscillator is an odd number of inverter gates connected serially in a ring. This configuration causes each node in the circuit to oscillate between a logic high and logic low. The frequency of oscillation is dependent on the delay of the inverters that make up the ring oscillator. Since every ring oscillator has a unique delay, it can be used to uniquely identify a part. A ring oscillator PUF (RO PUF) is a type of delay-based PUF that, with n bits, has two $2^{\frac{n}{2}}$-to-1 multiplexer (Mux) with $2^{\frac{n}{2}}$ ring oscillators per mux. A counter is used at each of the mux outputs and a race arbiter outputs a logic high or a logic low, depending on which ring oscillator is faster.

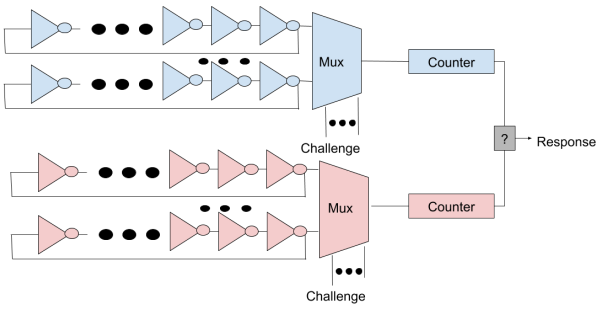


Figure 1.1: A Traditional RO PUF Schematic

There are three main performance metrics that are used to evaluate a PUF: reliability, uniqueness and randomness. Reliability is the measure of how repeatable the PUF can produce consistent CRPs under various conditions. This is measured by subtracting the average number of bit differences for all CRPs from 100%. In the equation below, n is the number of bits, m is the total number of times the 2^n CRPs were tested. T_m is the time of the final response, R_{T1} is the response of the first run, R_{T2} is the response of the second run, and R_t is response of run t.

$$Reliability = (1 - \frac{1}{m} \sum_{t=T2}^{Tm} \frac{HD(R_{T1}, R_t)}{n}) \times 100\% \tag{1.1}$$

Ideally, each PUF will produce a unique set of CRPs which is independent from all other PUFs. In other words, two PUFs given the same challenge should produce a different response. This is measured using the inter-Hamming distance of the design. The inter-Hamming distance is the number of bit differences between two PUFs (the same design on two different parts), which is ideally $\frac{n}{2}$. In the equation below, k is the number of different parts tested, R_i and R_j are the responses of two parts given the same challenge, and n is the number of bits.

$$InterHD = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=2}^{k} \frac{HD(R_i, R_j)}{n} \times 100\% \tag{1.2}$$

Randomness is a measure of how unpredictable the output is. A PUF that has response bits of mostly 1s is not as random as a PUF whose response has an even distribution of 1s and 0s. This is measured using the Shannon Entropy of the CRP set. In the equation below, N is the number of bits per response and p_i is the probability of a bit being a certain outcome.

$$Entropy = -\sum_{i=0}^{N-1} p_i \log_2 p_i \tag{1.3}$$

1.2 Motivation

As technology is increasingly accelerating at a high rate, security is becoming a more common issue with attacks happening at all layers of the software-hardware stack. While many are concerned about software security, this leaves hardware security in a state that is much to be desired. Even if the software security implemented for a specific application is "robust," it is only secure as the hardware on which it sits. Hardware Trojans (or manipulation of circuitry in an integrated circuit) are methods for attackers to gather sensitive information/data as attacks on the "lower portion of the stack" (closer to the hardware) have become more and more ubiquitous. Recent examples include Meltdown and Spectre, two hardware vulnerabilities on Intel's processors. While these attacks are not necessarily at the bottom-most level of the software-hardware stack, they demonstrate that every point in the stack is vulnerable and that a system is only a secure as its weakest point.

In recent years, chip manufacturing has been moved outside the United States, while Intellectual Property (IP) designing has largely remained inside the US. ICs must go through a long and arduous journey from the silicon foundry to the products we take for granted in our everyday life. During shipping ,parts have the potential to fall into the wrong hands, and at the same time, it's also possible that the counterfeit ICs are used instead of legitimate ICs. The fake chips that arise from counterfeiting practices are harmful to both the industry and the consumer - the original designer loses money and credibility as the consumer attempts to use a chip with higher failure rates and a shorter lifespan. Some counterfeit parts even make their way into applications like nuclear submarines or the braking systems in high speed trains. When a designer sends their schematic overseas, they should be cautious with their design. Did the foundry overproduce chips to reverse engineer and sell the design to competitors? Did they add in hardware Trojans to leak/destroy protected data? How can a designer ensure the chips are authentic? And if another individual purchase one of the ICs, how do they know they are receiving a genuine one and not a counterfeit chip? Designers need some mechanism to guarantee that their design remains authentic after production and unusable to counterfeiters.

These questions pose a need to both verify that the parts used are legitimate before installation, and 'lock' the part so that intellectual property (IP) is not stolen. A PUF is able to effectively accomplish these tasks since it can, ideally, create a unique identifier for authentication that can be verified before use or installation. A PUF's output cannot be cloned because a specific PUF's output cannot be determined before the chip is manufactured.

Chapter 2

Statement of Problem and Objectives

2.1 Statement of Problem

Counterfeiting and IP piracy has increased since silicon production and product manufacturing has mostly shifted abroad. Counterfeit parts are typically show problems well after the part has been installed, where it can be laborious to replace and dangerous to not. A method is needed to combat this maliciousness and protect both the individuals who may accidentally use counterfeit products as well as the intellectual property itself.

An RO PUF is effective at creating a random function that is both unique and reliable, and so it is ideal for the purposes of authentication. The idea of an RO PUF has been explored before, however there are weaknesses with certain challenges. This shortcoming means a bit selection algorithm must be used to select the most secure CRPs. This is inefficient as it can severely limit the input space for a PUF.

2.2 Objectives

We aimed to design, simulate, compare and post-process the data of a delay-based PUF on a Field-Programmable Gate Array (FPGA) as a proof of concept test for a scrambler block implemented in the challenge bus to eliminate the "bad" challenges.

Since this was a proof of concept, the PUF was only 8 bits wide. Two schemes, one serial and one parallel, will be tested in order to compare their performance to each other. Both RO PUF schemes were designed with Verilog and implemented on the FPGA using Vivado, as well as parameterized according to the aforementioned three main PUF metrics: reliability, uniqueness, and randomness.

This project did not include the physical manufacturing of the circuit, since IC tape-out is very expensive, and as students, we are not at liberty to make those kinds of purchases. Furthermore, it is not feasible to manufacture the system as the PUF in an IC is a sub-block that is meant to protect other IP, of which we have none.

Since the PUF was simulated on an FPGA, this project did not include voltage variation as increasing the supply voltage (typically a 5V USB input) does not stress the design, but instead only stresses the on-board voltage regulator,

which we did not design.

Initially, there were plans to test the performance of the PUF under different environmental conditions by using a Thermostream to change the ambient operating temperature, however the COVID-19 pandemic and observation of shelter-in-place restrictions made these tests impossible. Furthermore, the design on the FPGA was to be stress tested by running it at a high temperature (around 85˚C) for extended periods of time, but similarly, this was not possible due to the COVID-19 restrictions.

Chapter 3

Project Plan and Methodology

3.1 Project Plan

1. Fall Quarter

    - We began our research on Physical Unclonable Functions by reading through the current literature.
    - We looked at many schemes from which to draw inspiration. From these, we designed two new schemes to increase randomness and improve common weaknesses in RO PUFs.
    - We designed blocks in Verilog like ring oscillators, multiplexers, arbiters, shift registers, etc. To test these, we wrote and used SystemVerilog testbenches. These were verified using VCS.
    - We worked on integrating subsystems in Verilog using the above blocks.
    - All the verilog blocks and integrated systems were verified and synthesized using Vivado.

2. Winter Quarter

    - We designed our testing infrastructure.
    - We began implementation and initial placement on our FPGA (Xilinx Spartan 7 on the Arty S7 board). We debugged many of our issues in this step since we could not simulate the physical differences.
    - We wrote Python scripts for automating the data collection.
    - We began data collection for our parallel scheme.
    - We wrote more scripts to provide quantitative analysis on the collected data (the responses) via the three metrics of reliability, uniqueness, and randomness.

3. Spring Quarter

    - We designed our scrambler block for the serial scheme.
    - We integrated our serial scheme into the testing infrastructure.

- We made design adjustments and continued processing data.

3.2 Methodology

Our goal was to compare metrics for both schemes, and draw conclusions from that. We measured three core PUF metrics for both schemes: reliability, uniqueness, and randomness, and compared the performance of the two schemes.

In order to collect data, we created a list of all 256 possible 8-bit challenges, and passed them to the PUF sequentially. We read out one 'set' of 256 responses, and stored them in a text file to be analyzed later using Python scripts.

Our parallel scheme was to be used mostly as a benchmark. Since each output bit uses independent hardware, we expected this scheme to be highly unique and random. We expected the serial scheme to not be as random or unique, but still desirable because of its much lower area overhead.

3.2.1 Tools Used

Software Tools

We used a number of software tools to design and test our PUF designs.

- Synopsys VCS - We used VCS to write the initial Register Transfer Level Verilog Code and run early stage simulations. To verify each block, VCS has a digital waveform viewer during the simulation step that we used to make sure signals toggle properly under various conditions at specific times.
- Xilinx Vivado Design Suite 2019 - We used Vivado to run more thorough simulations as well as refine and place on our FPGA. It is an excellent tool that comes with specific placement capabilities that allow us to simulate the creation of different chips in manufacturing through placing designs in different slices.

Hardware Tools

To physically manifest the process variations, we used various hardware products:

- Digilent Arty S7 board with Xilinx Spartan 7 FPGA - Pat McGuire from Xilinx was kind enough to supply us with three of these boards to simulate the circuit itself placed on silicon die.

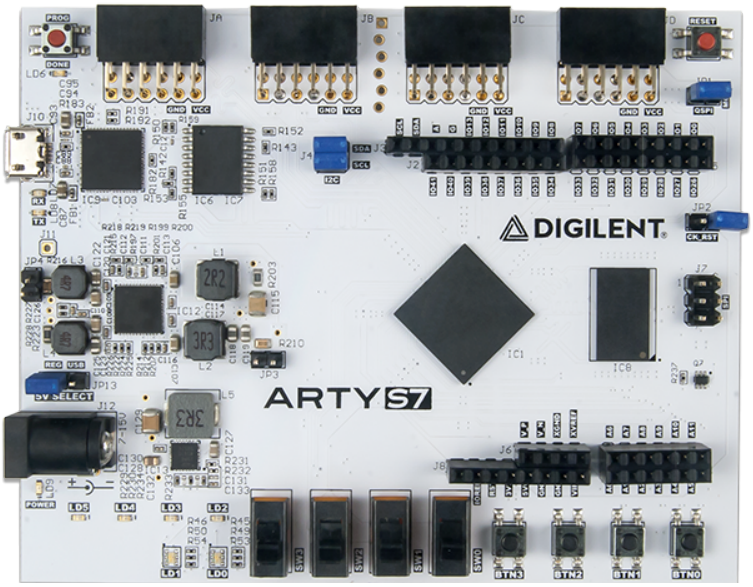


Figure 3.1: Digilent Arty S7 Board

- Raspberry Pi 4 - We used a Raspberry Pi to automate the sending of different challenges and monitor/record the responses.

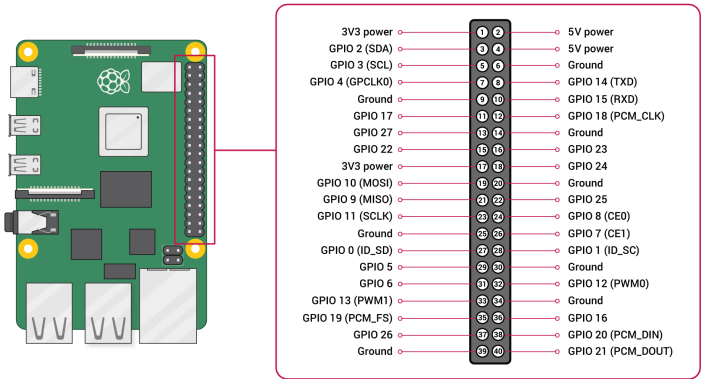


Figure 3.2: Raspberry Pi 4 with GPIO Pinout

Both of these systems came up a GPIO interface that allowed us to use single male-to-male wires to connect them to each other.

Chapter 4

Project Outcomes

As mentioned previously, we designed two schemes of a Ring-Oscillator Physical Unclonable Function that takes 8 bits as input and gives an 8-bit output.

4.1 Parallel Scheme

The parallel scheme is the simplest way to achieve more output bits, at the expense of hardware. Our proposed design needs 32 ROs for a single response bit, which even at 8 bits wide is quite area-expensive. Figure 4.1 shows a sub-block that is very similar to a typical Ring-Oscillator PUF. By placing 8 of these circuits in parallel, we now need 256 ROs, and the corresponding counters and MUXes to go with them. This circuit was used mostly as a benchmark for the serial scheme, since ideally there should be no correlation between partitioned ROs as described. This could change in layout, if these blocks were placed very close to each other, but can be corrected by a good layout designer - it is safe to assume this design creates responses with high entropy. This parallel scheme also has the advantage of speed. Our parallel scheme only needs each counter module to run once to generate an output. The number of ring oscillators per MUX scales with the size of our input challenge, and to generate a response longer than one bit, several of these schemes are placed in parallel and fed the same challenge. Figure 4.2 shows the overall parallel block diagram.

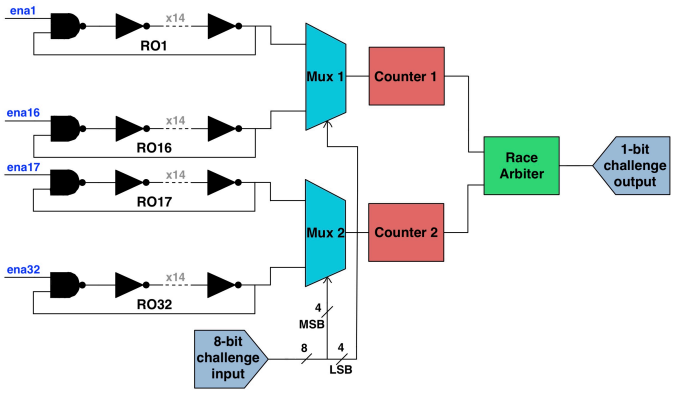


Figure 4.1: Parallel Sub-Block Diagram: 8-bit challenge resulting in a 1-bit response

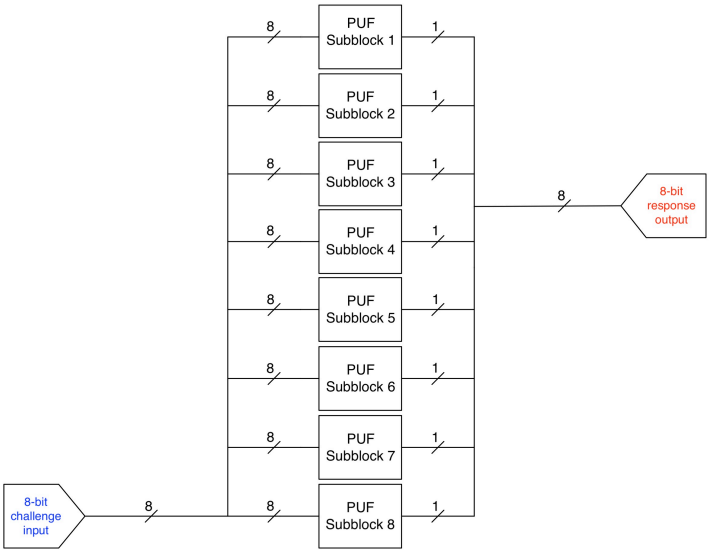


Figure 4.2: Overall Parallel Block Diagram: 8-bit challenge resulting in a 8-bit response by collating sub-blocks

In short, this scheme should be very random and unique, but uses a lot of area to do so. As an added bonus, this scheme also generates its responses very quickly.

4.2 Serial Scheme

In general, the main drawback of a ring oscillator based PUF is its high area overhead for a single output bit. Figure 4.3 shows our proposed serial design; the original challenge feeds into a scrambler, which will serially output eight different challenges into the same group of ROs, producing 8 output bits for a little area. Current serial PUF implementations often use a counter instead of a scrambler. While this does generate multiple output bits, the counter is linear, and so creates highly correlated 'adjacent' outputs. Since 0 and 1 are 'adjacent' in the counter, along with 1 and 2, 2 and 3, and so on, the responses generated by adjacent or closely adjacent challenges will share many response bits. The challenge of all 0's shares seven response bits in common with the challenge of all 1's and any two adjacent input challenges will always share seven output bits. Our first proposed suggestion is to replace this counter with a linear feedback shift register (LFSR), a common circuit used for pseudo random number generation. The LFSR works much like a counter, except it runs through the numbers in a pseudo-random but predictable order. The adjacency problem with the counter appears to have been solved - except for the fact that the LFSR is still a linear circuit. The adjacencies were not removed but just shifted. If our LFSR always goes from, for example, '54' to '205', then the responses for the challenges '54' and '205' will share seven bits.

We designed a scrambler circuit which uses some nonlinear component to eliminate this issue. Our scrambler circuit contains an extra 8-bit register in the LFSR to store the original challenge. The first output of the scrambler will always be the original challenge, and then every subsequent scrambler output will be the bitwise XOR of the original

challenge with the next LFSR value. Now, if our original challenge is '54', the output will be 255 XOR'ed with 54, which here is 251. This extra component removes the predictable adjacencies and will lead to less correlation. ***Our proposed serial scheme has a low area cost, and also uses less power. It incorporates a scrambler block which further increases unpredictability before signals are actually placed onto the MUX select lines.***

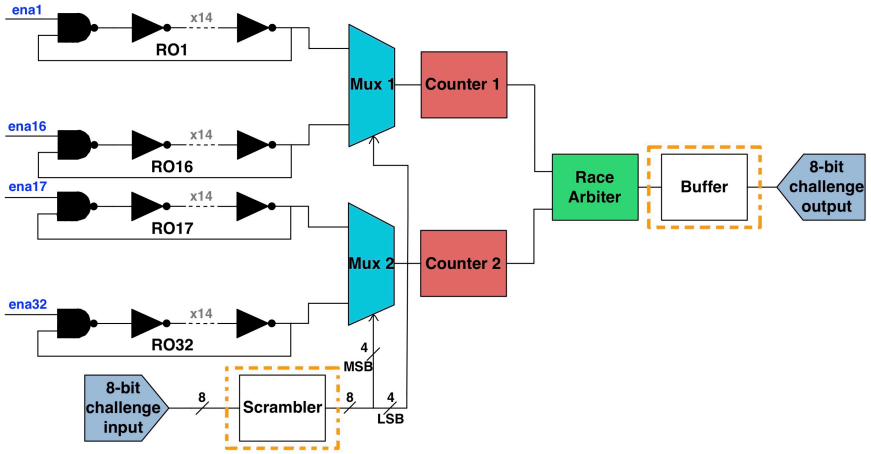


Figure 4.3: Serial Scheme: Notice that removing yellow-outlined modules is the same as a parallel sub-block

In short, this design will not be as random or unique as the parallel scheme, but requires significantly less area.

4.3 Data Collection Infrastructure

Our actual hardware setup is picture below in Figure 4.4. We used male to male wires to connect both systems to each other via GPIO pins. The GPIO interface allows the Raspberry Pi to send bits into the FPGA via eight sending pins and receive the responses via a separate set of eight receiving pins.

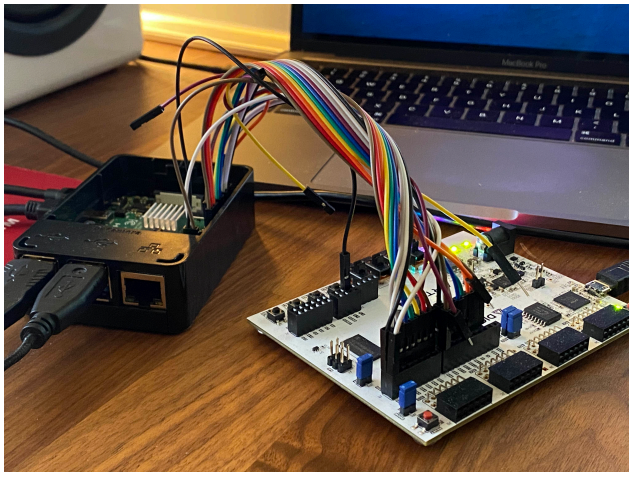


Figure 4.4: The Experimental Setup to Automate Data Collection

A text file was created of all possible challenges with 8 bits. In this file, we enumerated all 256 possibilities starting with all 0s and ending with all 1s, separated by newline. The file was then parsed line-by-line to isolate each challenge to send to the Arty S7. The Raspberry Pi and Arty S7 data collection configuration executes as follows:

1. The Raspberry Pi will initialize the eight challenge pins for the Arty S7
2. The Raspberry Pi will set the 'reset' pin high momentarily to reset the PUF
3. When the race arbitration is completed, the Arty S7 will output the response bits and pull the 'done' signal high.
4. The Raspberry Pi will record the response to a CSV file
5. Repeat steps 1 through 4 for the next challenge until all 256 challenges have been sent
6. Repeat steps 1 through 5 for the desired number of data sets.

Figure 4.5 below shows this cycle in a state-diagram form.

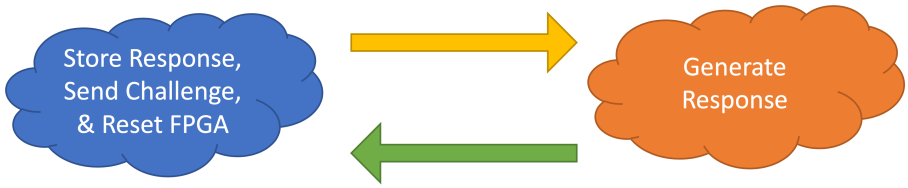


Figure 4.5: State Diagram Dictating Control Logic of Raspberry Pi

This interface was used instead of a serial communication protocol such as I^2C in order to maximize data collection with a limited period if time. Instead of sending each bit one after the other, this sends and receives them all once allowing us to run tests faster. One 1000 data set run with a 23-bit counter took around four hours to complete with our setup allowing us to gather ample data to analyze. While it does require more wires, there were more than enough GPIO pins on both the Raspberry Pi and the Arty S7 to accommodate this, making it a worthy trade-off.

Chapter 5

Final Design Results

5.1 Results and Validations

This section demonstrate the results of two different schemes (serial and parallel). We also validate our result using 3 different functional metrics.

5.1.1 Evaluation Metrics

We used Python and the libraries CSV and NumPy for most of our data manipulation. As defined earlier, our three main metrics are uniqueness, reliability, and randomness.

Reliability

is a measure of how often the same challenge outputs the same response. The same chip, when fed the same response, should always output the same challenge. This may not always be the case if: 1) the two RO frequencies being compared are already very similar, or if 2) the PUF is not stable over temperature or voltage variations. Consider a chip that produces responses R_{T1} for challenge C at time T1 and produces the response R_{T2} for the same challenge at time T2. A reliable PUF means that R_{T1} and R_{T2} are equal.

$$Reliability = (1 - \frac{1}{m} \sum_{t=T2}^{Tm} \frac{HD(R_{T1}, R_t)}{n}) \times 100\% \tag{5.1}$$

The designed PUF fulfills all our reliability requirements. Our proposed RO PUF results under reliability is shown in Figure 5.1. On average, it is 99.65% reliable, which indicates that the same challenge should almost always produce the same response. This value is on par with other research done on RO PUFs such as (1) and (4). From our serial scheme the results results are shown in Figure 5.2. The average reliability is 99.45%. Reliability increases when the differences in the frequencies are higher. Since the parallel scheme has more ring oscillators to choose from it is more likely to have higher differences in the frequencies.

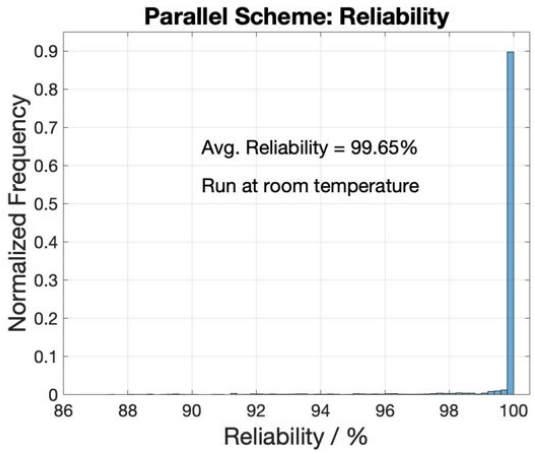


Figure 5.1: Parallel Scheme Reliability Results for 1000 Run Data Set

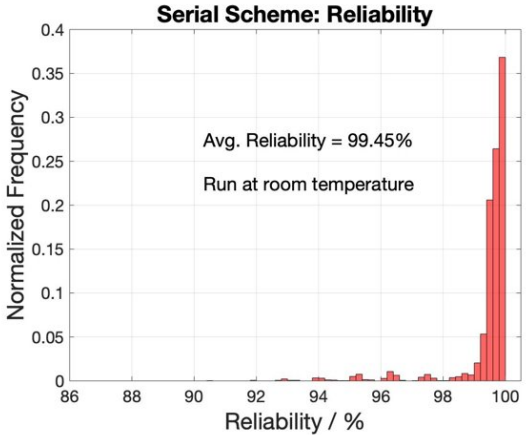


Figure 5.2: Serial Scheme Reliability Results for 1000 Run Data Set

Uniqueness

A unique PUF should have different behavior on each chip since the random process variation will be inherently different for each printed IC. The best metric for this is inter-Hamming distance. Inter-Hamming distance measures how far apart two responses generated by the same challenge on two different PUFs are. To measure uniqueness, we can implement the same PUF on multiple FPGAs and examine the differences between the responses. Also, we can choose different lookup tables (LUTs) for implementation on the same FPGA to have effectively different PUFs.

On applying the same challenges to two different chips:

$$InterHD = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=2}^{k} \frac{HD(R_i,R_j)}{n} \times 100\% \tag{5.2}$$

The uniqueness results of our purposed parallel RO PUF is shown in Figure 5.3. The uniqueness results of the serial RO PUF are show in Figure 5.4. Our parallel scheme has a mean inter-hamming distance of 46.4% and the serial

scheme has a mean inter-hamming distance of 62.7% which is very high. Since we have 8 bits this means that about 5 bits are changing on average.

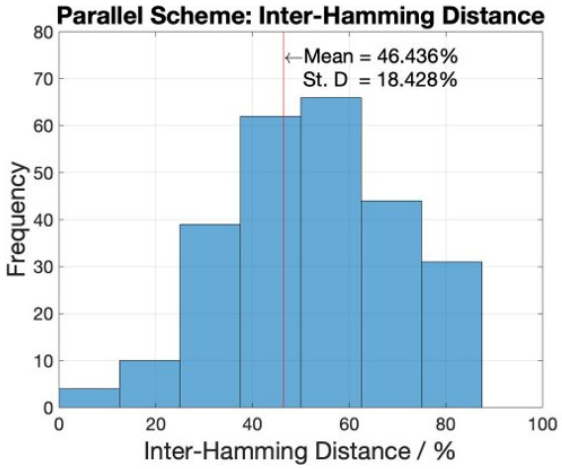


Figure 5.3: Parallel Scheme Interhamming Distance Results

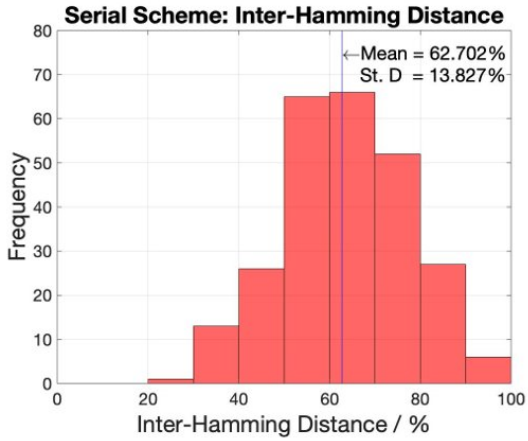


Figure 5.4: Serial Scheme Interhamming Distance Results

Randomness

A random RO PUF has challenge-response pairs that cannot be predicted unless the timing delays in every RO are characterized. Knowing any one CRP should provide no information about any other CRP. Calculating **"entropy"** of the response bits is an effective measure of randomness.

$$Entropy = -\sum_{i=0}^{N-1} p_i \log_2 p_i \tag{5.3}$$

Our average Shannon Entropy for each response averaged across all 256 possibilities is about 0.9253 per bit for the parallel scheme. While this number would ideally be higher, the design has strong unpredictability. For the serial

scheme the average Shannon Entropy is about 0.7963. This makes sense that we see a higher Shannon Entropy from the parallel scheme as we are using more ring oscillators which will give use more randomness. Every new piece of hardware we use adds a new chance for different physical characteristics to show up.

Chapter 6

Challenges

6.1 Inability to Simulate Randomness

In the early stages of simulating our designs in Synopsys VCS, there was no way to simulate randomness in the hardware as the "randomness" is introduced into the system as a function of its physicality. It is possible to manually specify delays in each inverter, though this is an infeasible solution if each simulation is to be repeated multiple times since it would involve manually changing the delay of each instantiated inverter (in the parallel scheme, there are close to 4000 inverters). The more practical solution is to synthesize the circuit on the FPGA and use the physicality of the hardware in the gate array. Monte Carlo simulations are another option that we considered. However, as much as it is desirable to perform this comprehensive type of simulation (where we would also be able to capture PUF performance even at various temperatures), it is not possible to carry out these simulations on a logic circuit.

6.2 Hardware Description Language (HDL) Code Optimization

Oftentimes, when using Electronic Design Automation (EDA) tools such as Vivado or VCS, optimizers are important process in the synthesis step since they erase logic that is not being used as well as simplify the circuit. In the case of ring oscillators, this is especially problematic since all ring oscillators can be simplified down to either a single inverter or buffer. In our design, we defined an ring oscillator as a NAND gate connected to 14 inverters as shown in Figure 6.1. When the the NAND gate's enable is high, the gate effectively turns into an additional inverter resulting 15 inverters. Logically, the optimizer believes that 15 inverters is equivalent to a single inverter. Consequently, the circuit is condensed down to that single inverter to get rid of delay. Figure 6.2 below shows this simplified RO. While the optimizer is correct in making this simplification, this does not work for our purpose since we want the delay.

```
wire w11;
wire w12;
wire w13;
wire w14;
wire w15;

assign w15 = ~(enable & w14);
assign w14 = ~ w13;             // w14 is the output we are interested in
assign w13 = ~ w12;
assign w12 = ~ w11;
assign w11 = ~ w10;
```

Figure 6.1: Verilog for Optimized Ring Oscillator

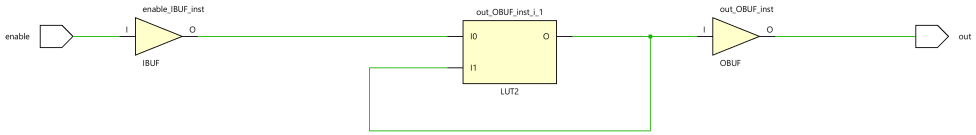


Figure 6.2: Optimized Ring Oscillator

In order to make the other inverters manifest, a small but fundamental change must be introduced and is tool-specific for Vivado (but can be replicated in other synthesis tools like Synopsys' Design Compiler with different syntax). A "dont_touch" directive is introduced in each wire when writing the Verilog which allows the full RO to be realized. Figure 6.3 shows the new Verilog with the "dont_touch" directive outlined in red. Figure 6.4 shows the corresponding full ring oscillator with all 14 inverters with one NAND gate synthesized by adding the directive.

```
(* dont_touch = "yes" *) wire w11;
(* dont_touch = "yes" *) wire w12;
(* dont_touch = "yes" *) wire w13;
(* dont_touch = "yes" *) wire w14;
(* dont_touch = "yes" *) wire w15;

assign w15 = ~(enable & w14);
assign w14 = ~ w13;             // w14 is the output we are interested in
assign w13 = ~ w12;
assign w12 = ~ w11;
assign w11 = ~ w10;
```

Figure 6.3: Verilog for a Full Ring Oscillator with All Inverters

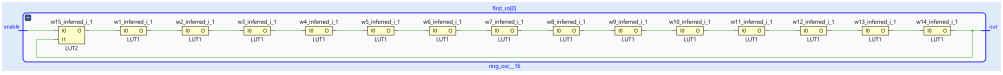


Figure 6.4: A Full Ring Oscillator with All Inverters

6.3 Automated Data Collection

During the creation of the testing infrastructure, we made a number of design decisions. Despite later settling on the Raspberry Pi as the tool of choice, we originally decided on using an Arduino to automate the collection of data. The Arduino has a number of benefits. Firstly, it is extremely close to the hardware operating in C allowing greater visibility into the bitstream sent and received. Secondly, we have tight control over electrical signals sent since it plays well with bread-boarding. Despite these benefits, the Arduino's close proximity to the hardware also gave us a number of challenges. First and foremost, the serial port was unwieldy. Opening the connection requires the computer as well as the Arduino to open the same channel. Since timing would be an issue, we used two Arduinos, one for sending challenges and one for receiving responses. The parsing of our eight-bit grey code needed multiple conversions between ASCII to its corresponding bit (0 or 1) and ultimately a GPIO LOW/HIGH and vice versa. The Arduino also outputs 5V which is too high for our FPGA. Finally, to make the automation worthwhile, a high baud rate was required. However, at baud rates upward of 192000, we saw bits dropped during transmission. These disadvantages pushed us to adopt the Raspberry Pi.

6.4 Timing

During the initial bringup of our automated testing environment, one of the issues we encountered was Timing. The first few inspections led us to believe that the "done" signal coming from the FPGA to the Raspberry Pi was never asserting, despite correct behavior in testbench simulations. We realized that due to timing issues between the FPGA and Raspberry Pi, the Pi was never latching the done signal. In the world of ASIC design, this problem is the Clock Domain Crossing (CDC) problem that is commonly encountered. In Figure 6.5, the "done" signal is high for one period of the FPGA clock. However, since the rate at which the Raspberry Pi is sampling (every rising edge) is too slow, the Pi only ever registers a LOW since the done signal comes back down by the second rising edge.

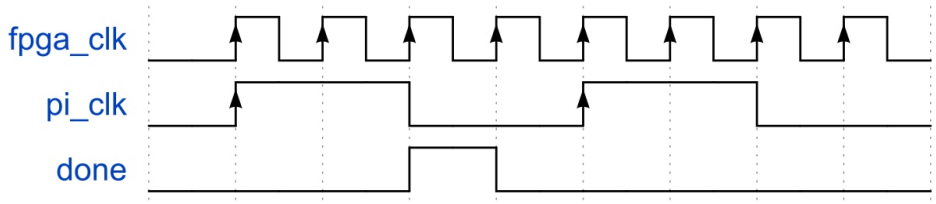


Figure 6.5: Bad Timing

We solved this in two manners. The first is to effectively slow the "fpga_clk" down to a rate that is equal or slower than the "pi_clk". If the clock on the FPGA is slower, then data has no problem passing from a slower clock domain to a faster one. However, this is a rudimentary solution.

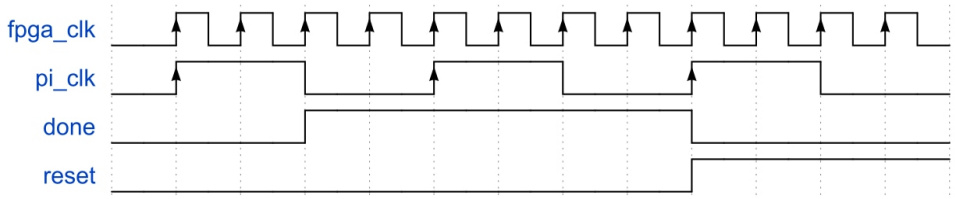


Figure 6.6: Timing that Demonstrates a Handshake

The second method that is more robust is to create a handshake. Therefore, even if data is passed from a fast clock domain to a slow one, it is guaranteed to be latched. In our case, we changed the "done" signal's behavior in such a way that it will assert for forever until a reset is sent through the system, effectively making it a "sticky bit." Since the reset is sent from the Raspberry Pi to the FPGA and runs on the Raspberry Pi's clock, the Pi is guaranteed to see the assertion. Figure 6.6 above shows the timing diagram with a handshake in place.

6.5 Design Placement

One method to measure the uniqueness metric is to manufacture our design. This is an expensive solution. Another alternative is to purchase many of the same FPGAs. While this is cheaper, there is a limit to how many we can purchase. Since Pat McGuire from Xilinx was kind enough to donate three FPGAs to us, we decided to make the most out of the limited hardware by placing our designs in different "slices" of each FPGA. This approach effectively simulates different chips with the same functional circuit coming off the line. Xilinx Application Engineers were able to assist us by pointing us to constraint file to specify the specific slices in which we wanted to place our design.

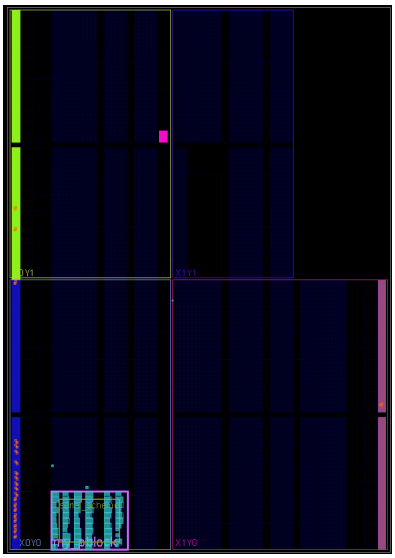


Figure 6.7: Serial Scheme First Placement

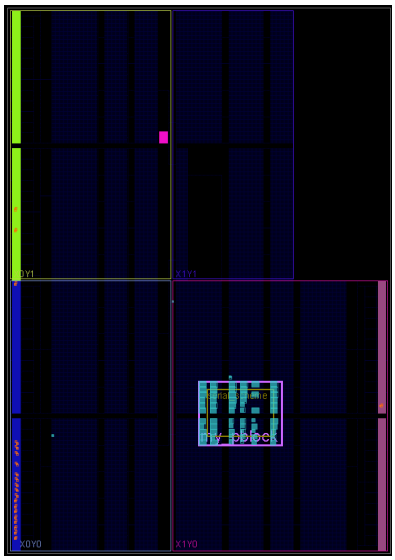


Figure 6.8: Serial Scheme Second Placement

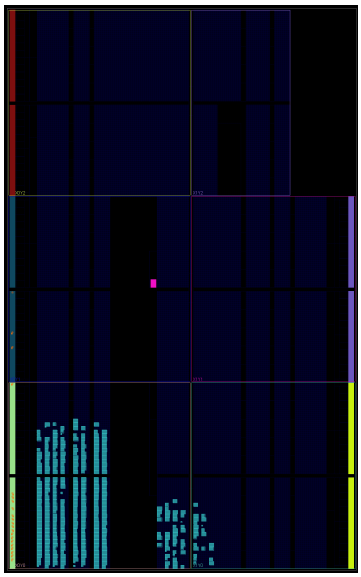


Figure 6.9: Parallel Scheme Optimized Placement

6.6 Reset Logic in the Serial Scheme

When comparing both the parallel and serial schemes, reset logic in the parallel scheme is simplified since it is handled by the software (refer to Figure 4.5) of the Raspberry Pi. On the other hand, serializing makes this logic more complicated as resets must be handled internal to the circuit. Resets must happen in a series of steps:

1. Pick a challenge
2. Finish the race between two chosen Ring Oscillators
3. Store resulting bit of race into the buffer
4. Increment the scrambler (to get a new comparison of ring oscillators)
5. Reset the counters
6. Repeat steps 2 through 5 an additional 7 times

The ordered list above suggests that these events happen sequentially and that there must be extra logic to dictate these resets. In our design, we chose to have the buffer govern this behavior. Therefore, the buffer must be clocked on some frequency. In our design, we chose to clock the buffer module on a 100 MHz clock. We once again ran into timing issues, specifically with Clock Domain Crossing. Each Ring Oscillator defines a clock domain (as it runs on its own unique frequency), since the counters are clocked on these ROs, all modules to the left of (and including) the Race Arbiter reside within that clock domain. Consequently, it is important to ensure that the data flows from the clock domain defined by the ROs to the buffer properly. We chose to use an asynchronous FIFO with a depth of four to ensure proper data latching.

Chapter 7

Conclusion

7.1 Key Takeaways

In this paper, we proposed parallel and serial RO PUF architectures and have found that these schemes are an effective method for uniquely characterizing chips. The parallel scheme provides secure authentication at a low cost, but at a large expense of area overhead. Therefore, it can be used as a standard benchmark by which to evaluate future PUF designs. The serial scheme drastically reduces overhead at little expense to randomness, but with some speed trade-offs.

7.2 Future Work

7.2.1 Expanded Reliability Testing with Different Ambient Conditions

Due to the COVID-19 complications, we were unable to test our parts performance at different ambient temperatures. Performing these tests will give us a more holistic reliability metric. Furthermore, potential future work includes conducting a High-Temperature Operating Lifetime (HTOL) test on our design running on the FPGA. An HTOL test is done by running the part at a high ambient temperature, around 85˚C, for an extended period of time. The Arrhenius equation can be applied to predict increased aging factor and therefore the performance of the design as the FPGA is used.

7.2.2 Increasing the PUF Bit Count

This change would take our implementation out of the realm of 'proof-of-concept' by increasing the width of the PUF from 8 bits to a more practical value such as 64 bits. An 8-bit PUF identifier can be brute forced, and so increasing the bit count makes the system more secure overall. This would also require the automated testing to use a serial communication protocol such as I^2C as there will not be enough GPIO pins to accommodate a wider PUF. It should be noted that testing the PUF and parameterizing it will take significantly longer since the input space explodes and becomes exponentially larger.

7.2.3 Collecting More Data

This project is heavily rooted in statistics, and collecting more data will strengthen our claims and provide more insights. Other than running the PUF for longer, the design can also be tested further by implementing it on different types of hardware (i.e. more and different types of FPGAs). Comparing our proposed scheme with one that has a circular shift register will give quantitative insights on the relative strengths of our design.

Chapter 8

Ethics

8.1 Ethical Considerations

Being able to authenticate chips has the ethical considerations of reliability, privacy, and safety. From the customer's standpoint, it is very important that they receive the chips that they think they should be receiving and that they are held up to the reliability standards. An application of a PUF is only sending a signal if the correct response comes back from the PUF authenticating it. This ensures privacy between devices. No matter the software security that is being used, if chips have not been authenticated, there no way to completely ensure that the correct chips are receiving the signals. Lastly, safety is a major concern, because if consumers are using chips that do not hold up to the security standards, they can have higher failure rates endangering people (i.e. exacerbated by their use in applications where human lives are at risk). For example, the Pentagon has received many compromised chips that go into systems such as anti-ballistic missiles. These compromised chips have a higher failure rate than the ones originally intended to be integrated into the system. This puts public safety in danger in and outside of the US.

8.2 Definition of "Goodness"

In our project, we are attempting to solve a problem that has recently arisen from an increased interest in attacking hardware in order to gain information or undermine the functionality of a product. Our design of a delay based physical unclonable function aims to prevent attackers from being able to pirate hardware and to combat counterfeiting. Therefore, the "goodness" in our intentions can be defined by our desire to secure information and champion privacy in a world where big data as well as technologies such as RFID that rely on digital IC authentication and small chips are increasingly becoming more commonplace.

8.3 Ethical Outcomes

We have found a new way to authenticate chips that will hopefully have increased reliability, randomness, and uniqueness making our PUF more secure. In doing so, we hope to have laid the groundwork for future work on PUFs. Still,

it is important to perform further testing as prematurely implementing the PUF in today's hardware security measures means that it is highly likely that security will be compromised and we will have failed to solve that problems that PUFs promise to resolve, namely IP piracy and counterfeiting. To mitigate this ethical risk, more extensive testing is required that is beyond the scope of this project.

8.4 Potential Misuse

There is always potential for misuse of PUFs. Some of these misuse cases can manifest in curious parties attempting to reverse engineer the PUF and clone function. As a result, our project would no longer be valid, and it would instead become a security risk.

At the core of hardware security is the relationship between designer and manufacture. Another typical misuse case would entail the storage of challenge and response pairs. It is possible that the manufacturer may be able to share these challenge-response pairs with a third party interested in attacking the design. Therefore, it is important to establish a trustful relationship between designer and manufacturer.

8.5 Conclusions

Hardware security is a growing concern and field since chips are now being fabricated overseas. A way to solve this is implementing PUFs to authenticate chips. We have been working on a ring oscillator PUF for authentication and are trying to improve the metrics of reliability, uniqueness, randomness, power, area, and timing. This will help protect the ethical considerations of safety, reliability, and privacy. Since the project came out successfully, we have made it easier to authenticate chips which benefits all the ethical considerations. However, we have to ensure that the design is not prone to reverse engineering and that we are not costing the end users too much.

Chapter 9

Acknowledgements

We would finally like to acknowledge people who have aided us in our efforts to develop these solutions to hardware security.

Firstly, we would like to thank Dr. Fatemeh Tehranipoor, our advisor, for her continual guidance throughout the year. Her work and dedication to the field of Hardware Security is unparalleled, and we are forever grateful for her advice and expertise on Physical Unclonable Functions.

We would also like to thank Pat McGuire from Xilinx for his donation of the Spartan 7 FPGAs as well as the technical support he and his team of Application Engineers have provided. When faced with numerous challenges, our team always had the aid of Xilinx AEs, especially in the early design phase as well as the placement phase.

Lastly, we would like to thank the Santa Clara University School of Engineering and the IEEE Circuits and Systems Society (CASS), Santa Clara Valley Chapter for their generous funding of our project. Along with providing the resources provided, IEEE CASS has provided us with the opportunity to present our design to industry professionals and advice during design process.

References

[1] Suh, G.E. and Devadas, S., 2007, June. Physical unclonable functions for device authentication and secret key generation. In 2007 44th ACM/IEEE Design Automation Conference (pp. 9-14). IEEE.

[2] Choudhury, M., Pundir, N., Niamat, M. and Mustapa, M., 2017, August. Analysis of a novel stage configurable ROPUF design. In 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS) (pp. 942-945). IEEE.

[3] Bernard, F., Fischer, V., Costea, C. and Fouquet, R., 2012. Implementation of ring-oscillators-based physical unclonable functions with independent bits in the response. International Journal of Reconfigurable Computing, 2012.

[4] K. Zhou, H. Liang, Y. Jiang, Z. Huang, C. Jiang and Y. Lu, "FPGA-based RO PUF with low overhead and high stability," in Electronics Letters, vol. 55, no. 9, pp. 510-513, 2 5 2019.

[5] M. J. Parvardeh and S. Baradaran Shokouhi, "A Ring Oscillator PUF Architecture With Enhanced Challenge-Response Set," 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, 2019, pp. 444-449.

[6] Erdinç Avaroğlu, "The implementation of ring oscillator based PUF designs in Field Programmable Gate Arrays using of different challenge," Physica A: Statistical Mechanics and its Applications, 2020

[7] R. Pramudita, S. Ramadhan, F. I. Hariadi and A. S. Ahmad, "Implementation Ring Oscillator Physical Unclonable Function (PUF) in FPGA," 2018 International Symposium on Electronics and Smart Devices (ISESD), Bandung, 2018, pp. 1-5.

[8] Tawil, Y., 2017. An Introduction To Counterfeit Ics: Counterfeiting, Detection And Avoidance Methods. [online] Atadiat. Available at: <https://atadiat.com/en/e-introduction-counterfeit-ics-counterfeiting-detection-avoidance-methods/ >[Accessed 5 June 2020].

Appendices

.1 Useful Resources

Code on GitHub Repository: https://github.com/Crimsonninja/senior_design_puf
Senior Design Video Recording: https://www.youtube.com/watch?v=_jGsgwwIHY4

.2 Senior Design 2020 Conference Slide Deck

SANTA CLARA UNIVERSITY
School of Engineering

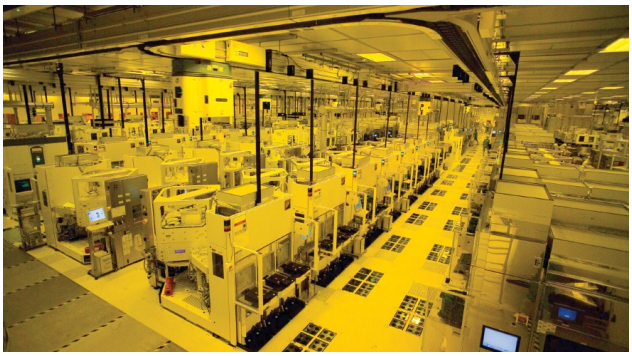
Delay-based Physical Unclonable Function

By: Abby Aguirre, Michael Hall, Timothy Lim, Jonathan Trinh
Advisor: Dr. Fatemeh Tehranipoor

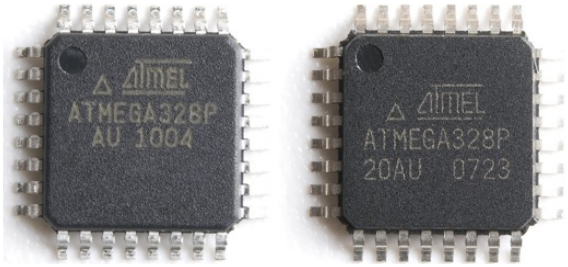
Outline of Presentation

- **Problem Statement**
- **Background**
- Project Objectives
- Project Plan
- Block Diagram
- Hardware Setup
- Design Challenges
- Project Outcomes and Results
- Final Project Timeline
- Conclusion

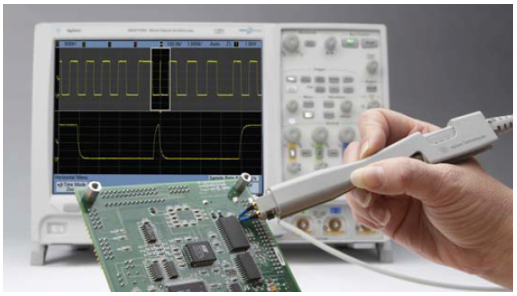
Outline of the Problem



ICs are manufactured abroad



15% of chips the Pentagon receives are counterfeit [1]

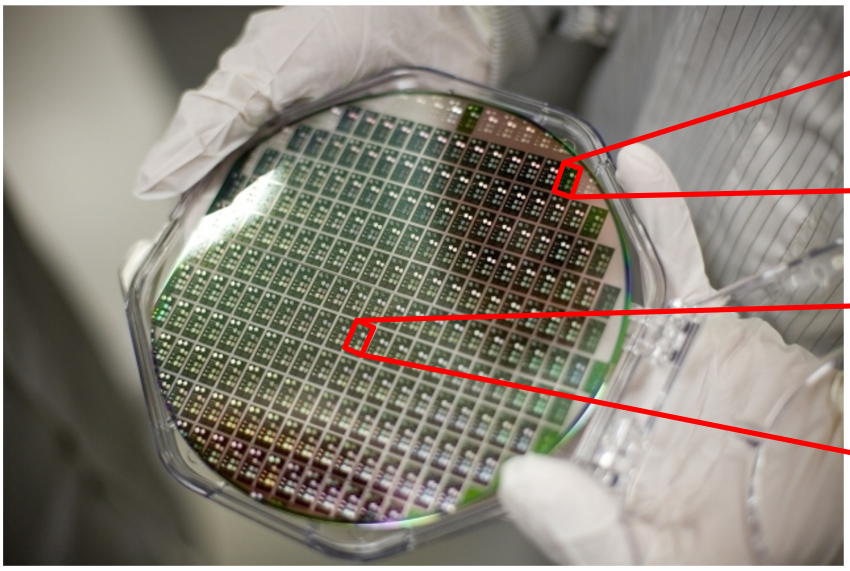


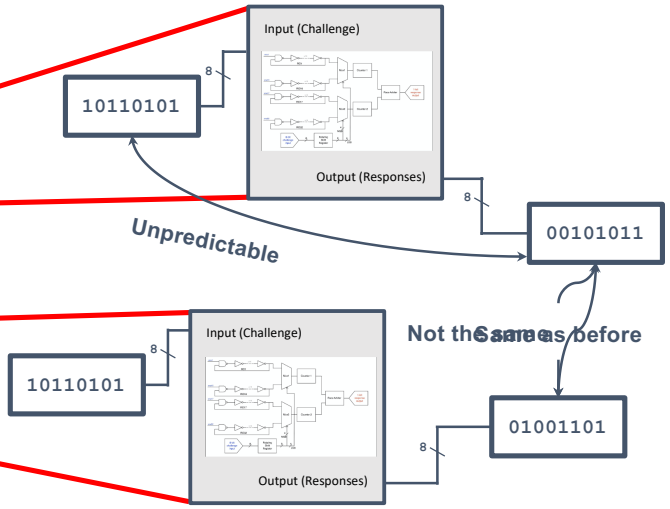
IP Piracy

Physical Unclonable Function

- Functionally identical
- Silicon Fingerprint
- Unpredictable output
- Static response
- Randomness, uniqueness, reliability

Random! Unique! Reliable!





PUF Metrics – Quantified

- **Reliability: Each chip produces consistent CRPs**
  - $Reliability = (1 - \frac{1}{m}\sum_{t=T2}^{Tm} \frac{HD(R_{T1},R_t)}{n}) \times 100\%$
- **Uniqueness: Bit differences in two chips' responses given the same challenge**
  - $InterHD = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=2}^{k} \frac{HD(R_i,R_j)}{n} \times 100\%$
- **Randomness: CRPs cannot be predicted**
  - $Entropy = -\sum_{i=0}^{N-1} p_i \log_2 p_i$

Outline of Presentation

- Problem Statement
- Background
- **Project Objectives**
- **Project Plan**
- **Block Diagram**
- Hardware Setup
- Design Challenges
- Project Outcomes and Results
- Final Project Timeline
- Conclusion

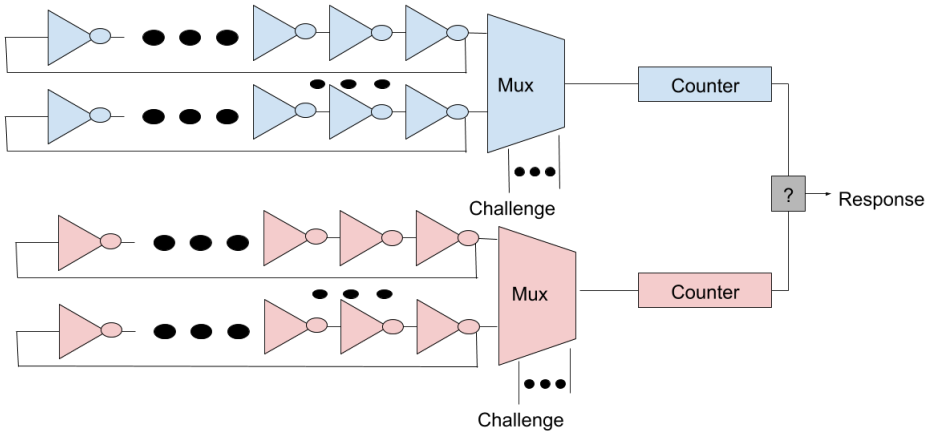
Project Objectives

Design, simulate, compare and post-process the data of a delay based PUF on an FPGA.

This includes:

- **Delay-based Ring Oscillator PUF on an FPGA**
- **2 different Ring Oscillator schemes**
- **8-bit ID for each circuit**
- **Optimization based on different metrics**
  - **Reliability, Uniqueness, Randomness**
- **Does not include:**
  - **Manufacturing**
  - **Supply voltage variations**
  - **Temperature variations**

Background – General Ring Oscillator PUF



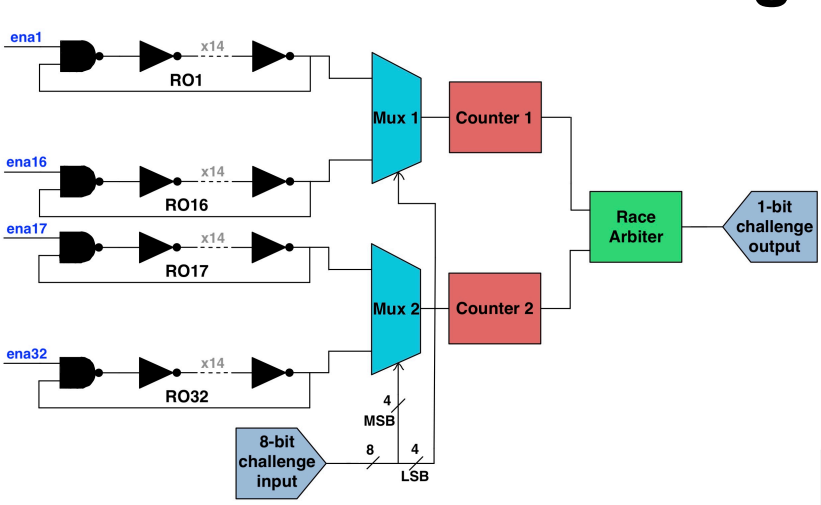
Project Plan

- Research (Fall Quarter)
- Simulate and prototype parallel and serial schemes (Fall Quarter)
- Verify and Synthesize on an FPGA (Winter Quarter)
- Data Collection (Winter and Spring Quarter)
- Analysis (Spring Quarter)

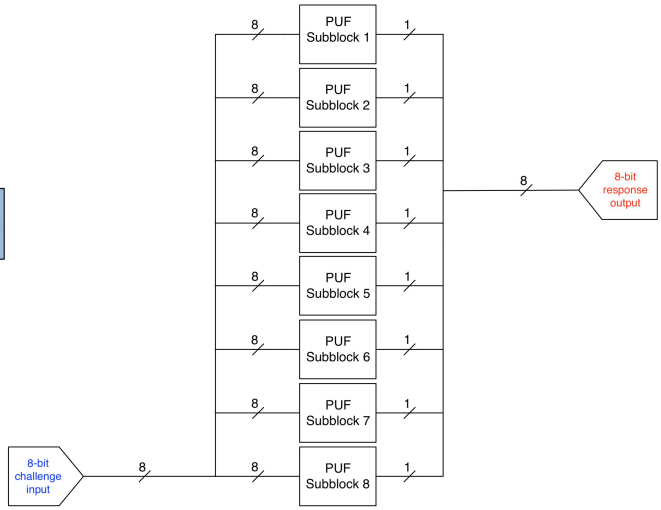
Division of Labor

- Abby Aguirre – Block-level Integrator & Data Scientist
- Tim Lim – System Architect & Test Engineer
- Michael Hall – Block-level Designer
- Jonathan Trinh – RTL Designer & Integrator

Parallel PUF Block Diagram

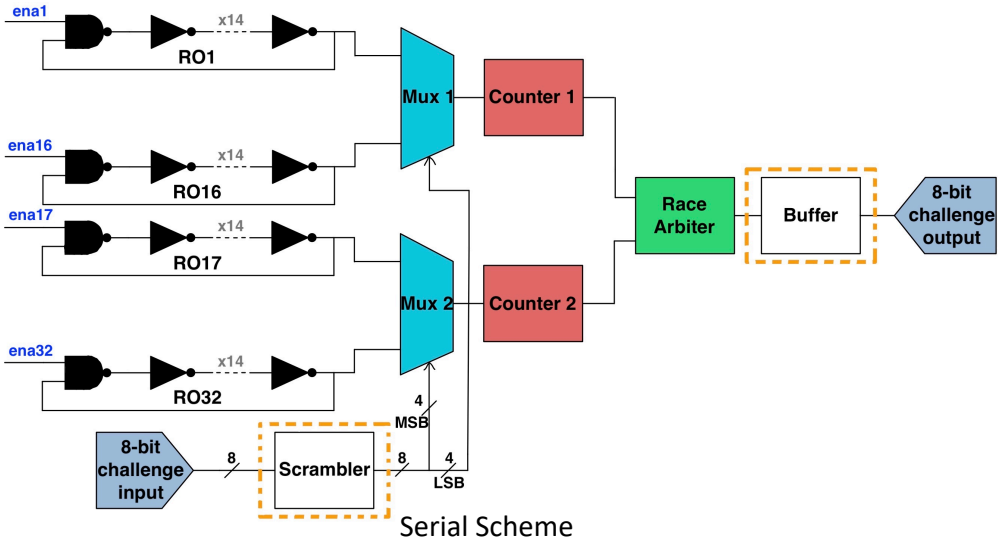


Parallel Subblock



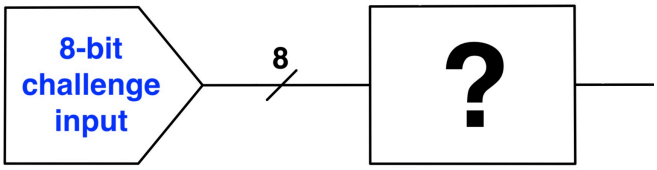
Overall Parallel Scheme

Serial PUF Block Diagram



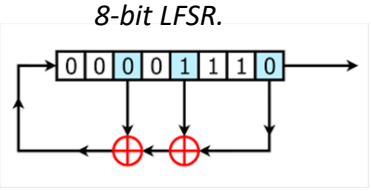
Serial Scheme

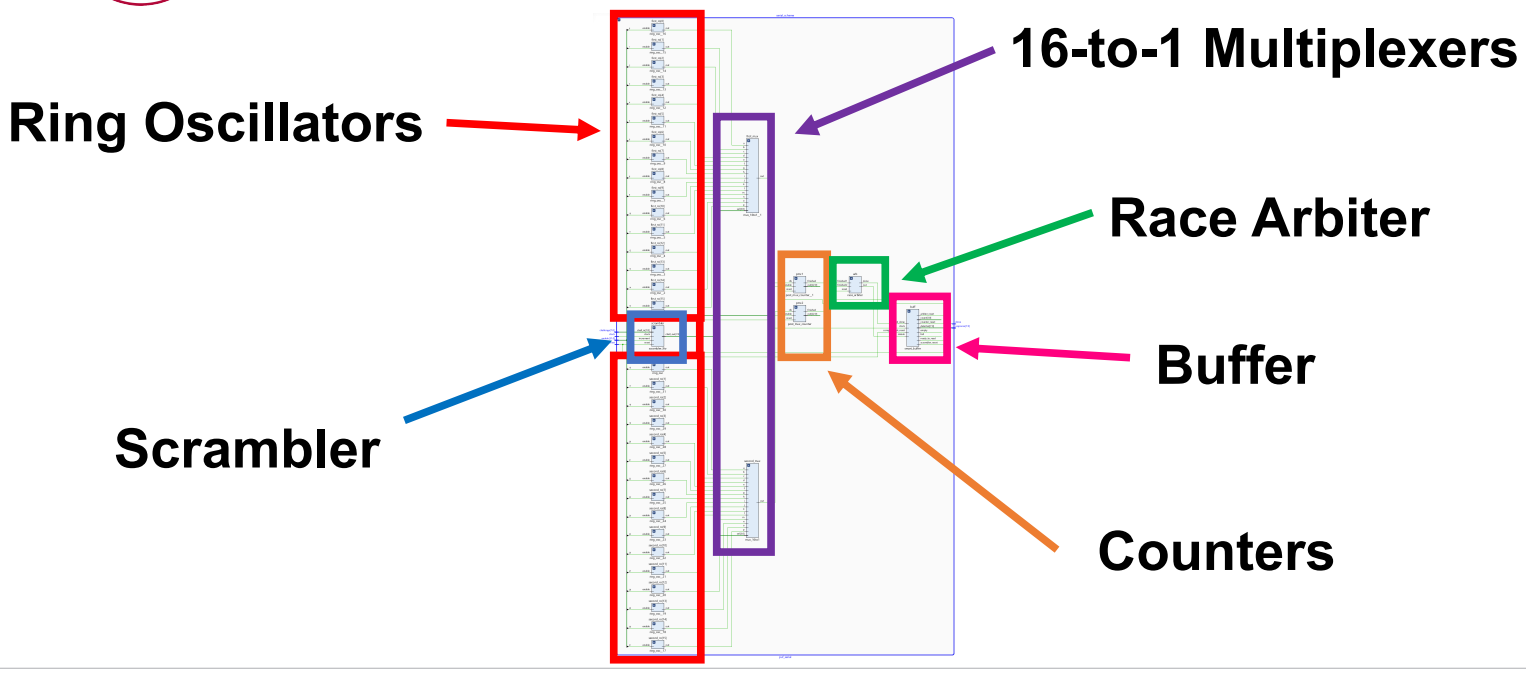
Serial Scheme Design



- **Circular shift register**
  - Lots of ‘bad’ challenges
- **Counter**
  - No bad challenges, but adjacency issues
- **LFSR and scrambler**
  - Shuffles adjacencies and removes them
  - Still vulnerable to netlist attacks

8-bit LFSR.



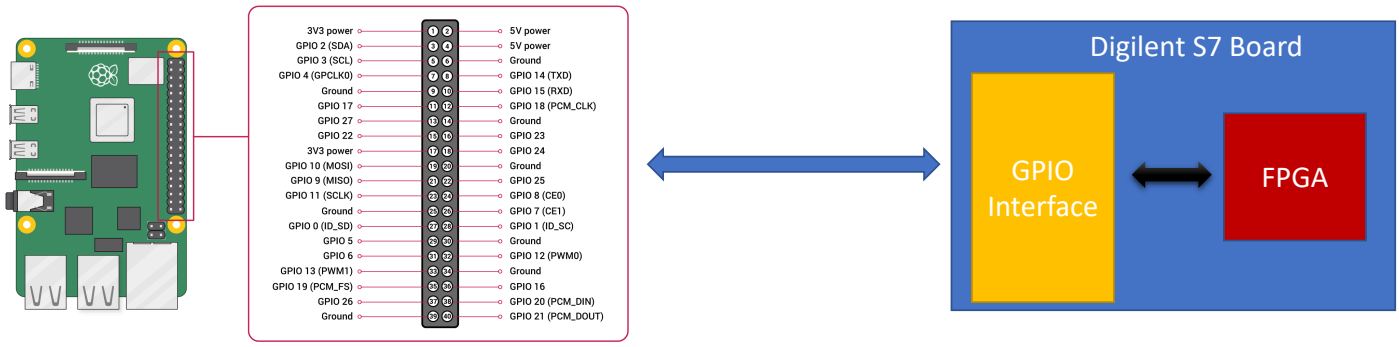

16-to-1 Multiplexers
Ring Oscillators
Race Arbiter
Buffer
Scrambler
Counters

Outline of Presentation

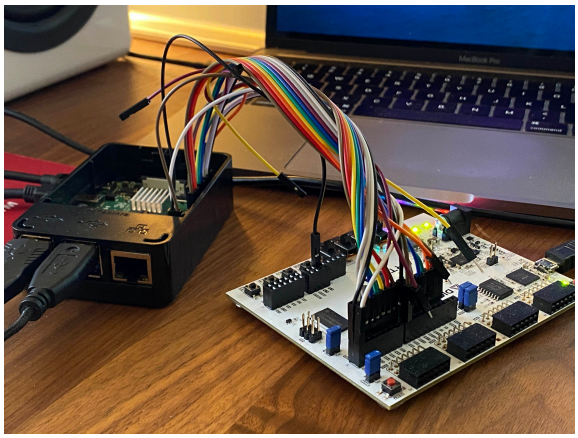
- Problem Statement
- Background
- Project Objectives
- Project Plan
- Block Diagram
- Hardware Setup
- Design Challenges
- Project Outcomes and Results
- Final Project Timeline
- Conclusion

System Hardware Setup

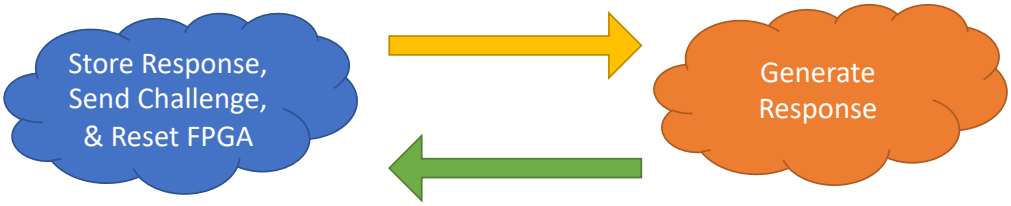
- To automate data collection, a Raspberry Pi ran Python scripts



Raspberry Pi



FPGA

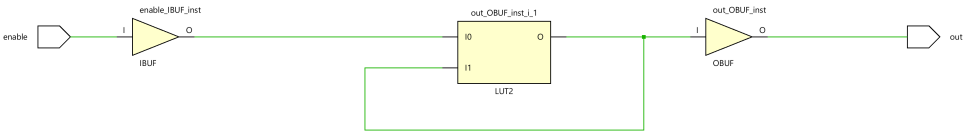


Design Challenges

- **Inability to simulate**
- HDL Code Optimization
- Automated Data Collection
- Timing
- Design Placement
- Reset Logic in the Serial Scheme

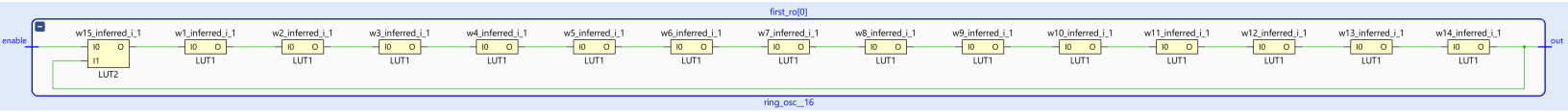
HDL Code Optimization

```
wire w11;
wire w12;
wire w13;
wire w14;
wire w15;

assign w15 = ~(enable & w14);
assign w14 = ~ w13;              // w14 is the output we are interested in
assign w13 = ~ w12;
assign w12 = ~ w11;
assign w11 = ~ w10;
```



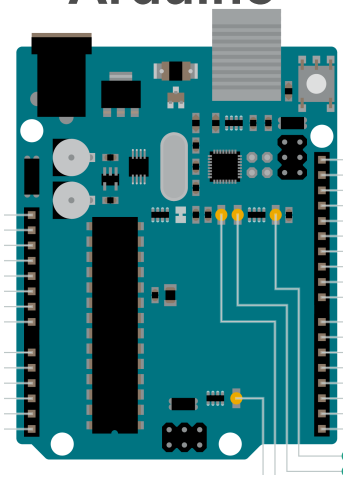
HDL Code Optimization

```
(* dont_touch = "yes" *) wire w11;
(* dont_touch = "yes" *) wire w12;
(* dont_touch = "yes" *) wire w13;
(* dont_touch = "yes" *) wire w14;
(* dont_touch = "yes" *) wire w15;

assign w15 = ~(enable & w14);
assign w14 = ~ w13;             // w14 is the output we are interested in
assign w13 = ~ w12;
assign w12 = ~ w11;
assign w11 = ~ w10;
```

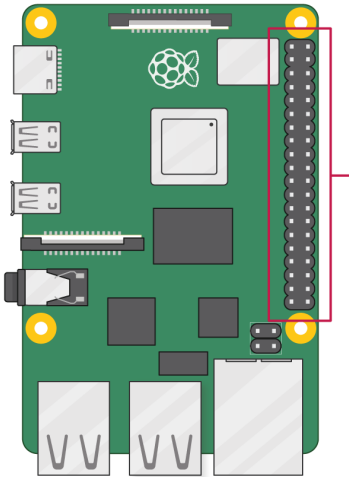


Data Collection

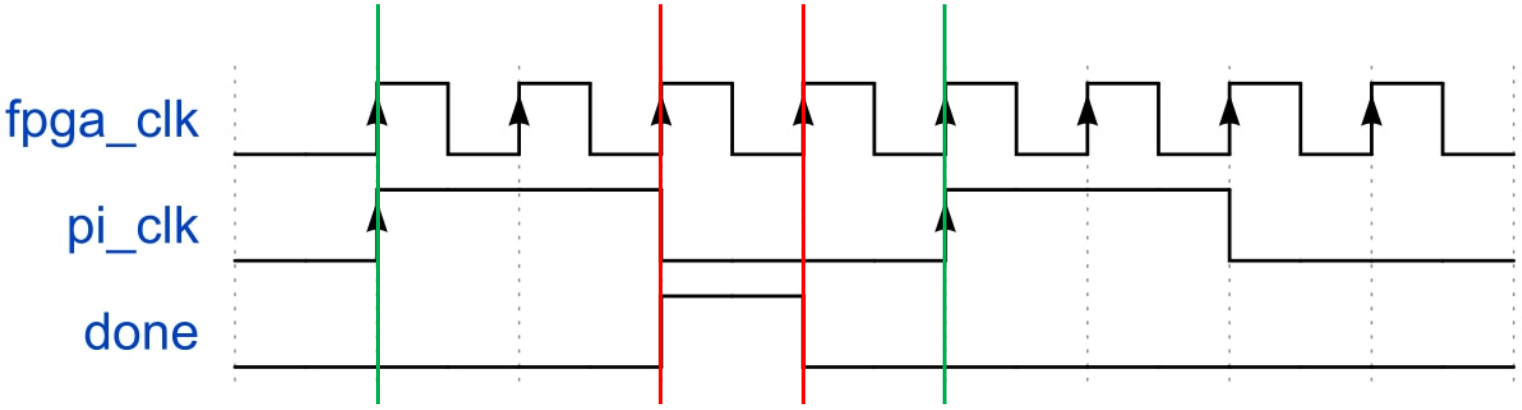
Arduino



Raspberry Pi

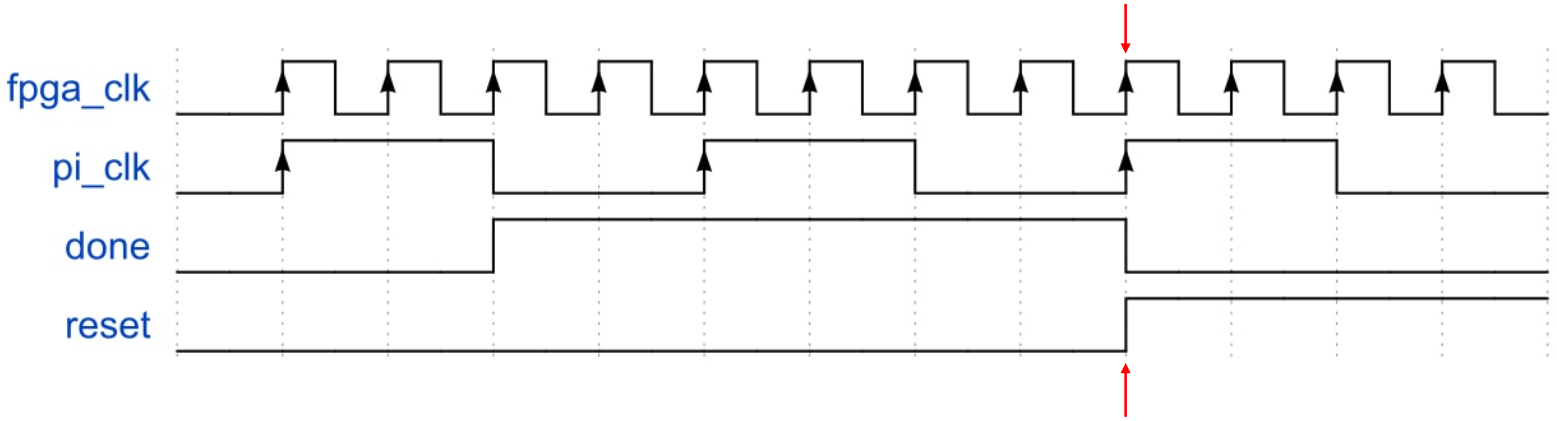


Timing



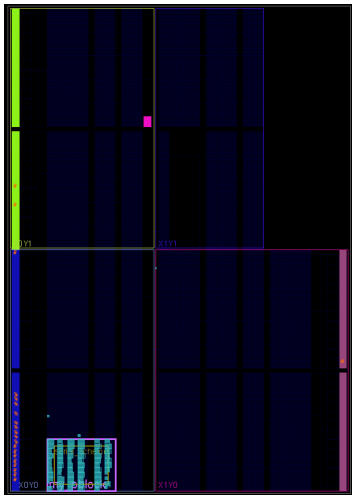
Raspberry Pi never latches the done signal!

Timing

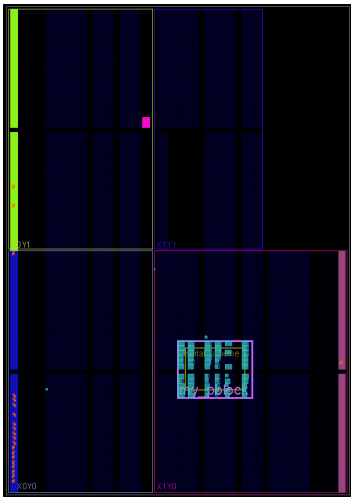


Make the “done” signal a sticky bit (a handshake)

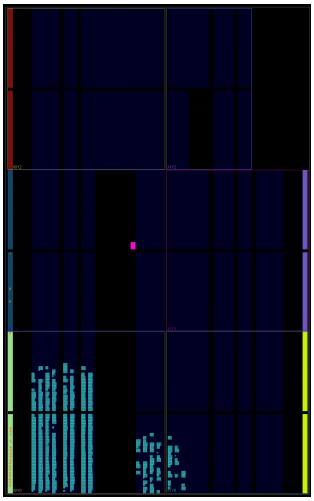
Screenshot of Implementation for Design Placement



Serial Scheme First Placement



Serial Scheme Second Placement



Parallel Scheme Optimized Placement

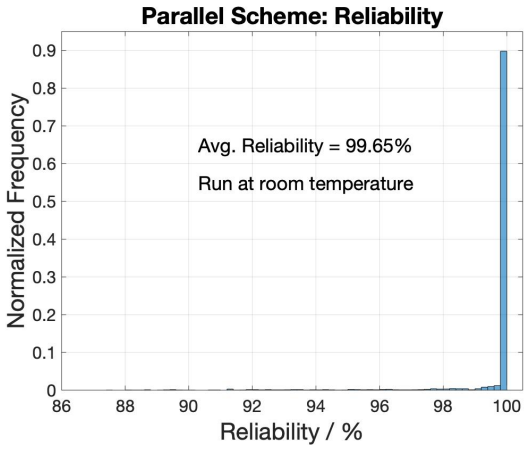
Design Challenges

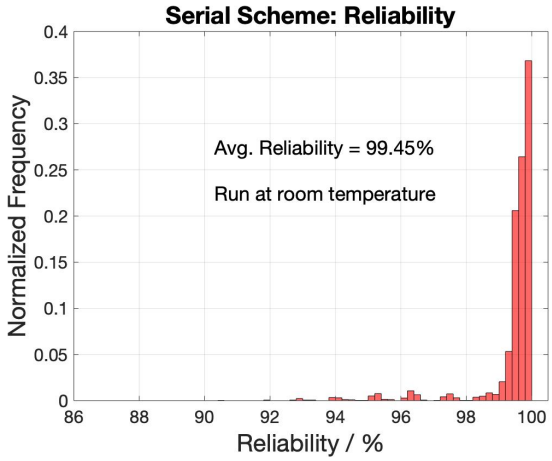
- Inability to simulate
- HDL Code Optimization
- Automated Data Collection
- Timing
- Design Placement
- **Reset Logic in the Serial Scheme**

Outline of Presentation

- Problem Statement
- Background
- Objectives
- Project Plan
- Block Diagram
- Hardware Setup
- Design Challenges
- Project Outcomes and Results
- Final Project Timeline
- Conclusion

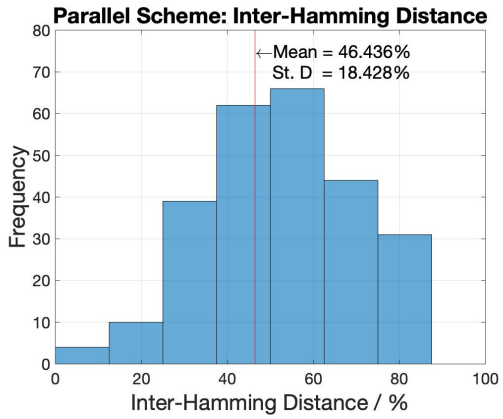
Reliability Performance Results

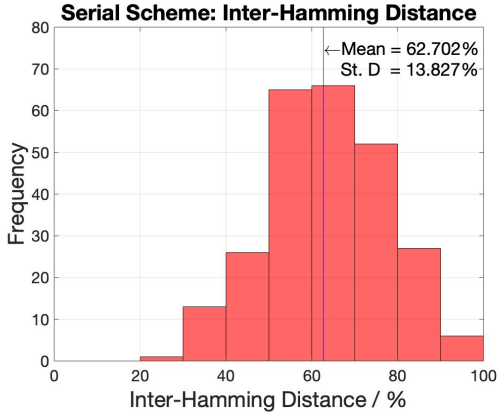




$$Reliability = (1 - \frac{1}{m}\sum_{t=T2}^{Tm}\frac{HD(R_{T1}, R_t)}{n})\times 100\%$$

Uniqueness Performance Results





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

Randomness Performance Results

$$Shannon\ Entropy = -\sum_{i=0}^{N-1} p_i \log_2 p_i$$

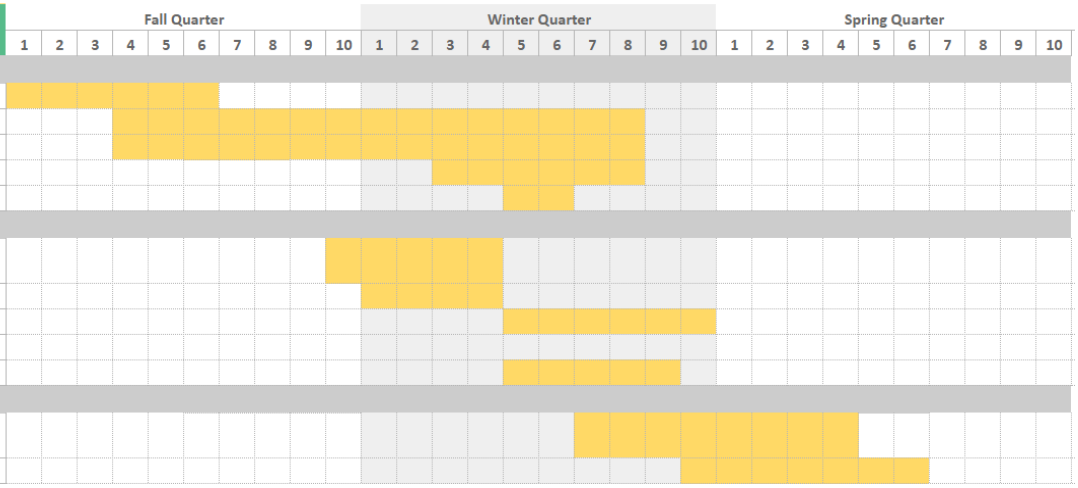
- **Parallel Entropy: 0.9253**
- **Serial Entropy: 0.7963**

Evaluation

Metric	Statistic	Parallel Scheme	Serial Scheme
Reliability	Reliability Mean	99.65%	99.45%
Uniqueness	Inter-Hamming Distance Mean	46.436%	62.702%
Randomness	Shannon Entropy Mean	0.9253	0.7963

Final Project Timeline

TASK NAME	TEAM MEMBERS
Preparation	
Research RO PUF Designs	All
Write Schemes in Verilog	Johnny and Michael
Verify in System Verilog	Johnny and Michael
Sythesis in Vivado	Abby
Load on FPGA	Johnny and Tim
Implementation and Testing	
Design Raspberry Pi Interface using GPIO Pins	Tim
Write Python Scripts for Automation	Tim and Abby
Collect Data	All
Temperature Variation and Aging*	Tim
Write Python for Post Processing	Johnny and Abby
Data Processing	
Process Data for Randomness, Reliability, and Uniqueness Metrics	Abby and Johnny
Compare Schemes	All



Conclusion

- Our PUF is an effective method for uniquely characterizing chips
- Parallel Scheme is a prime benchmark to evaluate other schemes
- Serial scheme is not as random, reliable, or unique as parallel, but still performs well with much less area

Paper Under Review

- A Systematic Approach for Internal Entropy Boosting in Delay-based RO PUF on an FPGA
- Authors: Abby Aguirre, Michael Hall, Timothy Lim, Jonathan Trinh, Wei Yan, and Fatemeh Tehranipoor
- IEEE International Midwest Symposium on Circuits and Systems (MWSCAS) 2020

Acknowledgements

- Professor Fatemeh Tehranipoor
- Pat McGuire and Xilinx
- Professor Jim Lewis
- SCU School of Engineering
- IEEE SCV CASS

References

[1] Suh, G.E. and Devadas, S., 2007, June. Physical unclonable functions for device authentication and secret key generation. In 2007 44th ACM/IEEE Design Automation Conference (pp. 9-14). IEEE.

[2] Choudhury, M., Pundir, N., Niamat, M. and Mustapa, M., 2017, August. Analysis of a novel stage configurable ROPUF design. In 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS) (pp. 942-945). IEEE.

[3] Bernard, F., Fischer, V., Costea, C. and Fouquet, R., 2012. Implementation of ring-oscillators-based physical unclonable functions with independent bits in the response. International Journal of Reconfigurable Computing, 2012.

[4] K. Zhou, H. Liang, Y. Jiang, Z. Huang, C. Jiang and Y. Lu, "FPGA-based RO PUF with low overhead and high stability," in Electronics Letters, vol. 55, no. 9, pp. 510-513, 2 5 2019.

[5] M. J. Parvardeh and S. Baradaran Shokouhi, "A Ring Oscillator PUF Architecture With Enhanced Challenge-Response Set," 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, 2019, pp. 444-449.

[6] Erdinç Avaroğlu, "The implementation of ring oscillator based PUF designs in Field Programmable Gate Arrays using of different challenge," Physica A: Statistical Mechanics and its Applications, 2020

[7] R. Pramudita, S. Ramadhan, F. I. Hariadi and A. S. Ahmad, "Implementation Ring Oscillator Physical Unclonable Function (PUF) in FPGA," 2018 International Symposium on Electronics and Smart Devices (ISESD), Bandung, 2018, pp. 1-5.