CRITICAL INFORMATION PROTECTION ON FPGAs THROUGH UNIQUE DEVICE SPECIFIC DIGITAL KEYS

THESIS

Miles E. McGee, CIV

AFIT/GCE/ENG/11-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCE/ENG/11-10

CRITICAL INFORMATION PROTECTION ON FPGAs THROUGH UNIQUE DEVICE SPECIFIC DIGITAL KEYS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Miles E. McGee, B.S.

CIV

September 2011

AFIT/GCE/ENG/11-10

CRITICAL INFORMATION PROTECTION ON FPGAs THROUGH UNIQUE DEVICE SPECIFIC DIGITAL KEYS

Miles E. McGee, B.S.

CIV

Approved:

________________________________	________
Dr. Yong C. Kim, Chairman	Date
________________________________	________
Major Todd R. Andel, Ph.D., USAF, Member	Date
________________________________	________
Major Mark D. Silvius, Ph.D., USAF, Member	Date

AFIT/GCE/ENG/11-10

Abstract

Field Programmable Gate Arrays (FPGAs) are quickly gaining popularity in both military applications and commercial technologies due to their low cost and reconfigurable capabilities. With FPGAs being used for military and other sensitive applications, the threat of an adversary attacking these devices is an ever present danger. While having the ability to be reconfigured is helpful for development, it also poses the risk of its hardware design being cloned. Static random access memory (SRAM) FPGA's are the most common type of FPGA used in industry. Every time an SRAM-FPGA is powered up, its configuration must be downloaded, to program the design on the device. If an adversary is able to obtain that configuration, they can clone possibly sensitive designs to other FPGAs. Cloning attacks must be prevented to ensure that the information stored on these devices remains secure. Protection against these types of attacks is especially pertinent within the Department of Defense, where FPGAs are being used in a variety of applications where cloning attacks could lead to critical designs being obtained by foreign governments.

A technique that can be used to protect FPGAs from these types of attacks is known as Digital Fingerprinting. Digital Fingerprinting takes advantage of the manufacturing variability that naturally occurs in the integrated circuit fabrication process, monitoring unique responses in path delay. If another factor can be introduced making the FPGA's operation dependent on more than the design specified within its configuration and response to external outputs, we can defend against cloning. This type

of solution would allow for an FPGA’s operation to be dependent on how the downloaded configuration interacts with the hardware itself. This research attempts to use Digital Fingerprinting technology to create unique device specific generated keys for use as encryption keys or control values for polymorphic circuits to protect information on FPGAs.

Acknowledgments

I would like to express my appreciation to my advisor Dr. Yong Kim for standing by me through my research, always providing a helping hand to get me back on track when I had lost my way. I would also like to thank my committee members Major Todd Andel and Major Mark Silvius, for their support proof-reading documentation and providing feedback in VLSI group meetings.

I would like to express heartfelt gratitude to the both the 11M and 12M students of the VLSI Research Group. The 11M students were full of helpful insight not only before they graduated, but made themselves available when I had questions, whether that was a phone call across base, or across the country. The 12M students took the time to listen to my ideas and provide assistance in any way they could, and served as a distraction when I needed to take a break.

I would also like to thank my fiancée, for letting me get by with ignoring wedding planning to work on my thesis, and telling me she was proud of me, even when I felt like quitting. Finally I would like to acknowledge my parents, who without their on-going support, I would not have ventured as far in my education.

Miles E. McGee

Table of Contents

List of Figures

List of Abbreviations

CRITICAL INFORMATION PROTECTION ON FPGAs THROUGH UNIQUE DEVICE SPECIFIC DIGITAL KEYS

I. Introduction

1.1 Research Motivation

In recent years, Field Programmable Gate Array (FPGA) use within the Department of Defense (DoD) has been on the rise. Within the DoD, FPGAs are being used in aircraft avionics systems, and in other system critical hardware[**1**][**2**]. FPGAs are available as commercial off-the-shelf products that are low cost and easy to obtain, making them ideal for product development and testing, and the choice for moderate production runs and rapid product development[**3**]. Protecting these systems is important to their continued use in the DoD as well as in the commercial sector.

Protecting FPGA systems from reverse engineering and product cloning is necessary if the DoD is going to continue relying on their use. An FPGA's configuration, can be easily cloned if the device bitstream can be intercepted. The bitstream is a list of instructions sent to the FPGA when it is powered up, instructing it to configure itself in a certain way. In a cloning attack, a FPGA's bitstream is intercepted and used to configure another FPGA, resulting in counterfeit systems. Such cloning attacks not only affect commercial gains, but can also lead to critical DoD technology being obtained by foreign nations or military groups. In order to ensure the security of the designs and information stored on FPGAs security measures need to be put in place to defeat cloning attacks.

Two types of countermeasures that can be used to thwart cloning attacks include active and passive countermeasures. An active form of protection involves monitoring of the FPGA by the software running on it to authenticate the device, or adding additional protections via software or hardware. Examples of active countermeasures include checking a device's serial number, additional verification hardware, and data encryption. One protection option already being offered by most FPGA manufacturers is an option to encrypt the bitstream so that a design cannot be directly extracted [**4**]. Bitstream encryption protects the design from being recovered, but as shown by [**5**][**6**] does not prevent against cloning.

Passive countermeasures are those which rely on a design being specifically created to operate on a particular FPGA. Passive countermeasures may rely on variables which cannot be controlled by an adversary such as device architecture, signal propagation delay, or power consumption. Passive countermeasures are more robust than active measures, because they cannot be easily defeated. An active countermeasure can be circumvented through hardware and software manipulation, but a passive countermeasure is dependent on the actual structure and composition of the device itself, which cannot be changed.

The ability to create a passive countermeasure which allows for a design which operates correctly only on a specified FPGA or group of FPGAs would ensure that a design cannot be cloned onto an unauthorized device. Such a design would either fail to run on an unauthorized FPGA, or operate incorrectly. This ability would also allow for the creation of unique device specific keys for encryption and identification. If the design functions correctly only on a specified FPGA, a key generation method used on one FPGA will produce a different key if run on another FPGA. These generated keys can then be used as encryption keys for encryption hardware running on the device, such that two FPGAs will not encrypt the data the same way, and since the key is generated on the board, the keys are unknown to the user, or an adversary.

These keys can also be used for device identification since they are unique to a specific board. A FPGA can be profiled; the generated key recorded, and then sent out into the field. Upon the devices return, it can be profiled again, a comparison of the two keys will show if the FPGA is indeed the same board. Device identification is very important, to ensure that a returned device hasn't been counterfeited or tampered with. This property is especially important within the DoD to ensure that a device on loan to another country or contracting company has not been tampered with in an espionage attempt or an attempt to cause harm.

1.2 Problem Statement

The Very Large Scale Integration (VLSI) Group at the Air Force Institute of Technology (AFIT) has been working topics leading to this research for several years. In 2009, Patel introduced a technique to identify FPGAs through physical variations that occur during manufacturing [**7**], and in 2010 and 2011, Anilao and Stanton laid the groundwork for digital key generation by producing a probe to measure propagation delay between two paths [**8**][**9**]. Through the use of this probe, a FPGA's structure can be utilized as a passive countermeasure. This technique can be used to positively identify an FPGA, and create unique keys to ensure the security of the FPGA and the design stored within it. By continuing the research efforts of previous VLSI group members, an improved identification and key generation method can be created.

1.3 Research Objectives and Contributions

The desired outcome of this research is the creation and analysis of device generated digital keys to be used for device protection and identification. This research is broken into three main parts; digital key generation, key comparison and analysis, and key implementation. These three parts have the goal of implementing a usable digital key system and analyzing its ability to create unique device specific keys.

1.3.1.1 Circuit DNA/Digital Key Generation

The generation of circuit DNA and digital keys is based off of the work by Stanton [**8**], leveraging his tunable probe design to characterize an FPGA and produce keys. By varying the number of buffers on a pair of paths, the probe can be used to determine how much delay is required on each line to equalize the delay between the two paths. When multiple readings are taken, the path delays can be averaged and used to and provide a single bit value which represents the delay. 128 probes are used in the design in order to generate a 128-bit digital key, which would be useful for encryption purposes. A full sweep of possible delay values on each probe will be used to generate Circuit DNA for device family matching. This research is evaluated in two ways; digital key stability and distinguishability.

1.3.1.2 Key Comparison and Analysis

In order for generated keys to be useful, they must be repeatable. The measure of the repeatability of a key is referred to as stability. Stability is an important measurement because it is a reflection of the ability for a particular FPGA to consistently produce the same key. A consistent key is a vital characteristic if the key is to be used for encryption, to ensure that data is encrypted properly each time ad can be subsequently recovered.

A measure to describe uniqueness of a key is what is referred to in this research as distinguishability. Ensuring that the keys generated on separate FPGAs are unique is important to this research. Keys which are the same across multiple boards are not secure for encryption because they are not unique to a specific device. Unique and stable keys are required for encryption to be effective.

1.3.1.3 Key Implementation

One of the main goals of this research is to produce unique and stable keys to be used for encryption keys. The ability to produce such keys concludes research that has been ongoing within the VLSI research group for several years, crossing from theoretical ideas to actual implementation. The implementation of the digital keys involves using them as both encryption keys for a symmetric key system such as the Advanced Encryption Standard (AES) and as part of an unclonable software/polymorphic circuit system[**10**][**11**][**12**]. This implementation is a final check to ensure that this research has culminated into a feasible working product.

1.4 Thesis Organization

The following chapters detail the process and the extent of this research while also providing the results. Chapter 2 contains the necessary background information required to understand the topics discussed in the research. Chapter 3 provides methodology behind the research, along with a proposed experiment. Chapter 4 contains the results of the experiment described in the previous chapter. Chapter 5 provides a summary of this research, its relevance, and the major contributions of this thesis; it also contains suggestions for further research.

II. Literature Survey

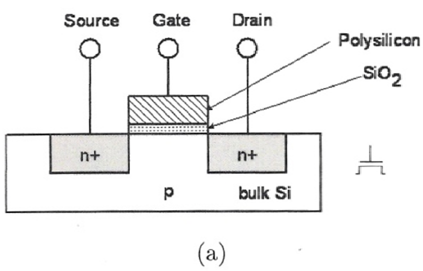
The following chapter presents some background information to orient the reader with the topics addressed in this thesis, and to give a better understanding of previous work that has been done in the field. This chapter is divided into four sections as follows: Section 2.1 covers physical variations within CMOS fabrication, Section 2.2 covers Physical Unclonable Circuits, Section 2.3 discusses the previous research by the VLSI group at AFIT on the topic of digital fingerprints, and Section 2.4 provides a summary of the literature survey.

2.1 Physical variations in CMOS circuit manufacturing

The basic building block of any integrated circuit is the transistor. At the most basic level of abstraction, a transistor functions as a switch. Two types of transistors exist; negative and positive Metal Oxide Semiconductor, commonly known as nMOS and pMOS, the difference between which is the determined by the type of electrical carrier (positive or negative) under its gate. Transistors contain three main parts, a gate, a source, and a drain. The minimum voltage required for a transistor to activate is known as the threshold voltage or V_t. When V_t has been met, the transistor will turn on, allowing current to flow. In an nMOS transistor, while the voltage between the source and drain, known as V_{ds}, is between 0V and $V_{gs} - V_t$, where V_{gs} is the voltage between the gate and source, it is in what is known as linear mode. While in linear mode, I_{ds}, which is the current from source to drain, increases linearly with V_{gs}. When $V_{ds}$is greater than $V_{gs} -$

V_t, the transistor is operating in a region known as saturation, where a raise in V_{ds} will not result in a increase in I_{ds} [**13**].

A transistor's size is determined by the width of the gate. A wider the gate, results in a larger channel width beneath the gate. A wider channel width results in a slower transistor. Transistor technology is described by the fixed length of the transistors gate. Currently, 32nm technology is common in commercial products, including Intel's Core i3, Core i5 and Core i7[**14**].



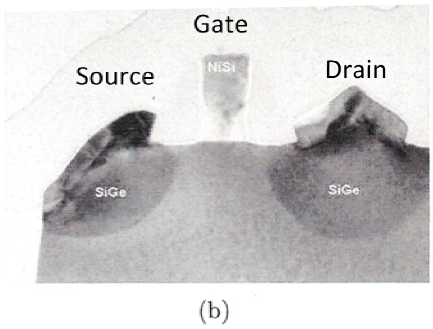


Figure 1. (a) An ideal transistor design. (b) Image of a transistor as viewed by an electron microscope [15**].**

With such small feature sizes, creating transistors exactly as they are designed becomes a difficult problem. Figure 1 (a) shows an ideal transistor, notice that the source and drain are both clearly defined, with sharp straight edges. In real world fabrication, transistors are not shaped the same way. Figure 1 (b) shows the image from a scanning electron microscope of a transistor. Due to the unpredictable behavior of atoms, they cannot be forced into forming straight lines. The diffusion of ions into the substrate results in random deviations[**16**]. Deviations occur due to circumstances which cannot be controlled in the manufacturing process, such as variation after oxide deposition and chemical mechanical planarization [**17**][**18**][**19**].

Creating transistors that are exactly alike is unlikely during the CMOS circuit manufacturing process. Each transistor created varies slightly from others around it in dimension. These variations cause the shape of each transistor to be unique and random. These variations in shape cause differences in performance characteristics, such as threshold or saturation voltage levels [**20**]. Since no two transistors are exactly alike, when many of the same type of integrated circuits are created using the exact same masks, significant differences in delay can exist between equivalent circuits fabricated with the same mask set [**21**].

2.2 Physical Unclonable Functions

Physical Unclonable Functions (PUFs) are manufactured devices which rely on the variations within the manufacturing process to affect the outcome of a circuit. PUFs were created to identify and authenticate a device through exploitation of delay variation and by measuring transient response [**22**][**23**]. The premise of this work is that PUFs can protect against counterfeiting. Since the probability of manufacturing two devices with the same manufacturing variations is very low, an adversary would have to fabricate a very large number of devices and perform measurements on all of them to identify one that would be able to serve as a potential counterfeit [**24**][**25**].

Several different types of PUF circuits have been suggested and implemented. The Arbiter PUF used in [**21**], shown in Figure 2, is a multiplexer based circuit that depends on pairs of multiplexers whose select lines are connected together and controlled by the user. The propagated signal delay through the system effects which signal gets latched into the D-Latch at the end of the multiplexer chain. Arbiter PUF shows promise in that it is very stable, producing a probability of variation less than 1% on a single device. The real problem with this design is that when tested, its ability to produce distinguishable results across devices was low, with a probability of variation of about 25%, meaning that only ¼ of the time it produced different results on different devices, making it unacceptable as a form of unique identification.

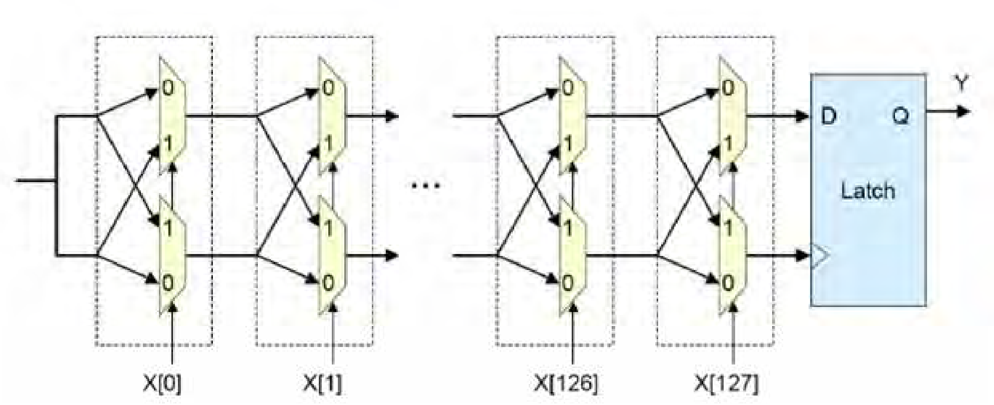


Figure 2. Arbiter PUF [21**]**

Ring Oscillator (RO) PUF, shown in Figure 3, relies on a specified number of oscillators which oscillate at different frequencies due to manufacturing variances in the device. The number of oscillations over a specified amount of time is stored in a counter after each RO. The output of the RO PUF is determined by a comparator which compares two RO, assigning a value of '1' or '0' based on which counter has a higher count. RO PUF's weakness lies in that it is very susceptible to temperature, voltage and other environmental conditions, resulting in an unstable output.

Another form of PUF that has been tested is the Butterfly PUF (BPUF)[**26**], shown in Figure 4. BPUF uses groupings of two D-latches to generate unique outputs. The BPUF is excited by placing a '1' value on the excite line; this causes the D-Latches to enter an unsteady state for a short amount of time. When a steady state is achieved, its value is output on the "out" line. The steady state signal produced is dependent on the physical variations of the transistors used in the design. According to the authors of [**26**], the results of this research are promising in that BPUF is stable over a temperature range of -20 to +80°C, but has a noisy response which causes the design to be unstable.

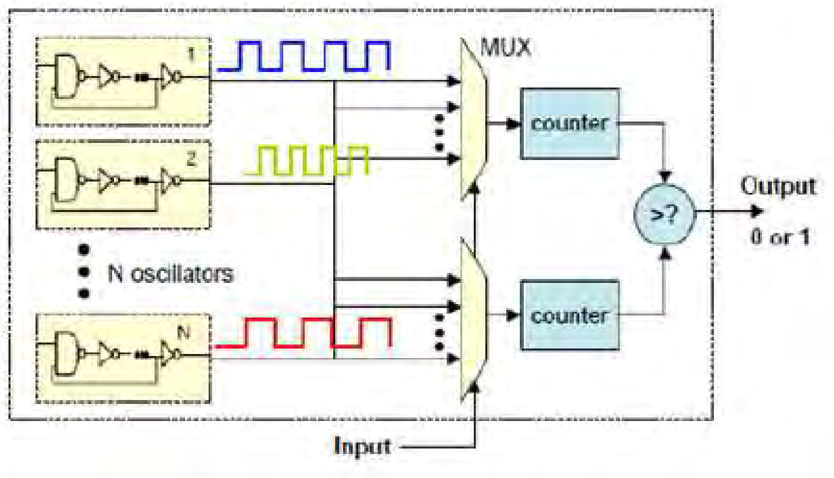


Figure 3. Ring Oscillator PUF[21]**]**

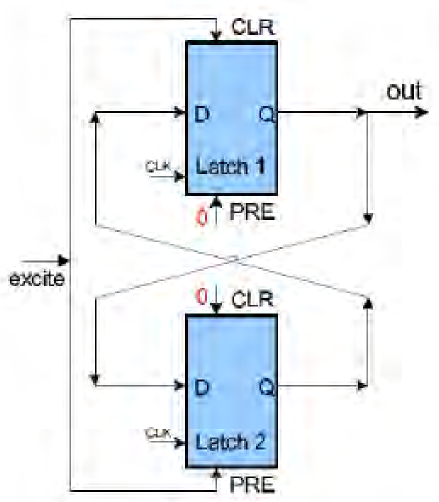


Figure 4. Butterfly PUF [26**]**

2.3 Digital Fingerprint

Similar to PUF, the Digital Fingerprint (DF) concept was developed concurrently at AFIT. In [**15**], Crouch's research involved taking advantage of imperfectly shaped transistors and allowing their effects to stack upon one another in a large combinational circuit magnifying their effect. Crouch's design implemented a 64-bit combinational multiplier with Linear Feedback Shift Registers (LFSR) generating pseudo-random inputs. 16-bit one-hot-state shift registers were used on the outputs to detect and calculate the total number of glitches[**27**] generated at each output, shifting a '1' value through the register. Crouch was able to uniquely identify each board he tested based off of the results in the shift registers, his work served as a proof of concept for the Digital Fingerprint technique.

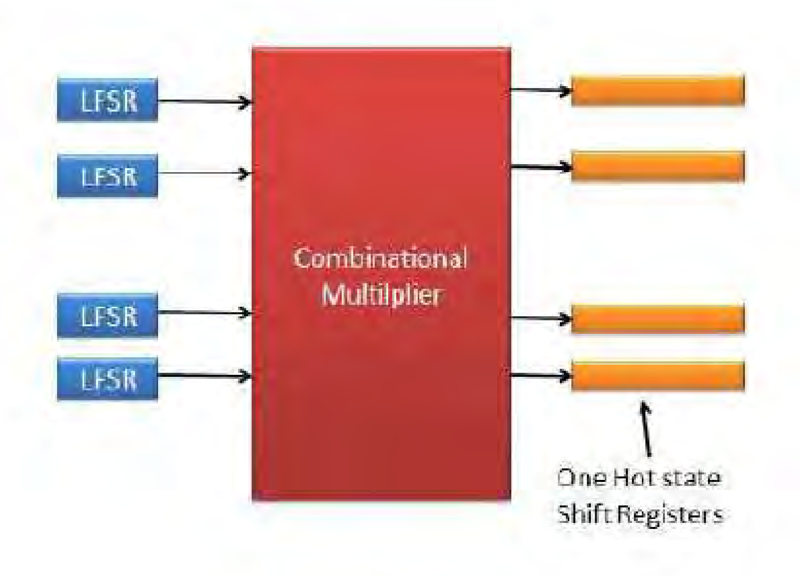


Figure 5. Digital Fingerprint Generator used by Crouch [15**]**

In [**7**], Patel improved upon the DF concept introduced by Crouch. He improved upon the hardware design, reducing the amount of user inaction required for testing to reduce the amount of user error added to the testing, he also extensively tested the system looking at factors which Crouch had not. In his research, Patel was able to demonstrate that there were timing requirements which had to be met in order for glitches to be recorded by the shift registers. He showed that digital fingerprints have several factors: input combinations to the system, the type of combinational circuit used, the location of the design on the FPGA, and the method used to detect and record the glitches [**28**].He also showed that operating temperature had an effect on glitch generation.

Anilao introduced a smaller glitch generating circuit in [**9**], which was then modified by Stanton in [**8**] to produce the Tunable Glitch Probe (TGP) laying the groundwork for this research. Anilao's design (shown in Figure 6) included a network of buffers and multiplexers allowing for the delay in the system to be controlled by the user. Anilao's design took advantage of the fact that swapping the input of the circuit between minterms can cause a glitch consisting of a unintended value to appear at the output for a very small amount of time. The Karnaugh Map of Anilao's circuit [**9**] can be seen in Figure 7 with the minterms circled.

Stanton modified Anilao's design reducing it from three inputs (A, B, and C) to just a single input. Stanton also added buffers on the primary inputs of the design to add additional delay between probes when they are connected in series, to ensure that a measurable path delay difference exists between probes. Stanton's design can be seen in Figure 8.

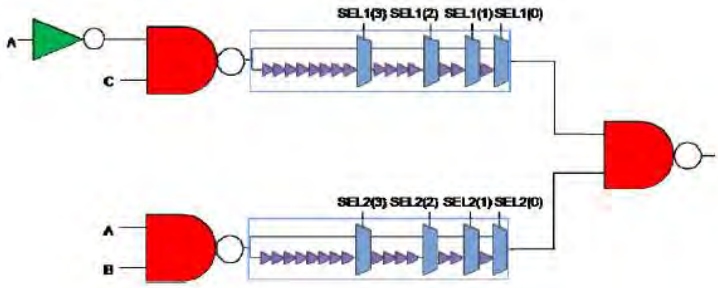


Figure 6. Glitch Circuit Introduced by Anilao [9] implementing the equation F = A'C + AB.

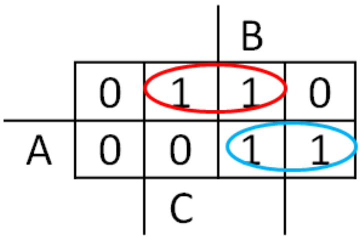


Figure 7. Karnaugh Map of Anilao's Design from [9]

The TGP measures the delay between two paths by using a one hot state shift register to count glitches which occur at its output when its input is pulsed. Figure 9 shows an example of outputs that the shift register may receive. The red and blue lines are representative of the paths in the TGP in Figure 8. Adding or removing delay in the tunable delay element will cause the colored lines in Figure 9 to shift to the right and left respectively. When the delay has been equalized, the output of the TGP will be a constant signal as in Figure 9 (c), this is because the path controlling the falling edge of the glitch has more delay than the path controlling the rising edge.

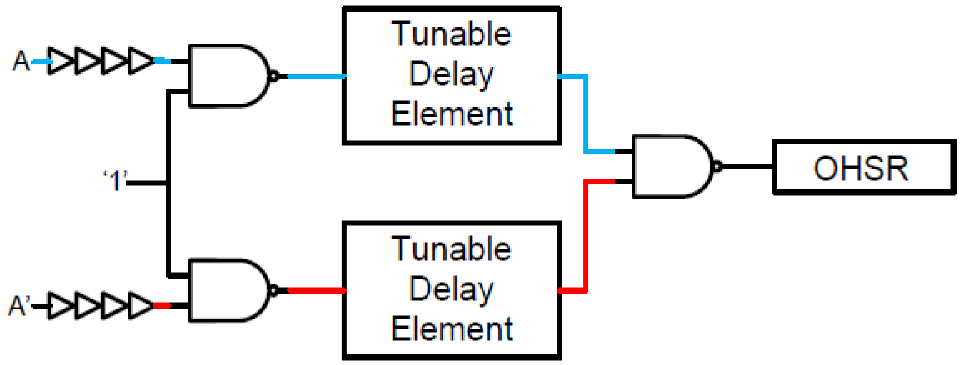


Figure 8. Tunable Probe[8]

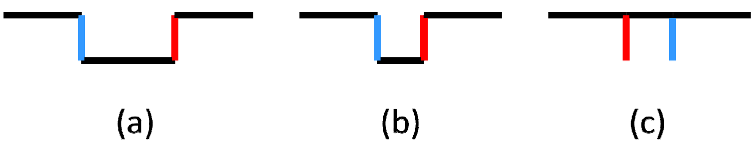


Figure 9. Glitch Probe outputs. (a) and (b) represent glitches while (c) represents both paths being balanced

Stanton used the TGP as a foundation for the concept of Circuit DNA and a fingerprinting unit. Using an array of TGPs, he used a characterized an FPGA by sweeping the tunable delay elements across a range of 0-256 buffers recording them in a table format, which he referred to as Circuit DNA. An example of Circuit DNA can be seen in Figure 10. Circuit DNA was a byproduct of Stanton's research, generated in the early stages of his research. Stanton instead focused on using the TGP to produce a repeatable fingerprint. By using a cascading array of TGPs, with buffer select lines connected together, he sought to generate unique fingerprint strings to describe individual FPGAs. His digital fingerprint was more than 90% stable, and among the FPGAs he tested had a distinguishability of about 4.5%.

While Circuit DNA seemed to be a side effect of Stanton's research, it provides an important basis for new research. By sweeping through all 256 possible input buffer values between zero and fifteen on both upper and lower paths, Stanton was able to determine at which buffer combinations the TGP produced a balanced delay. As can be seen in Figure 10, blocks in the table colored green represent buffer combinations which result in more delay in the bottom path of the TGP, while yellow blocks represent combinations where the top path contains more delay.

Number of Buffers on Bottom Path

Number of Buffers on Top Path	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	10	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0
5	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0	0
6	10	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0
7	10	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0
8	10	10	10	10	10	10	10	0	0	0	0	0	0	0	0	0
9	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0
11	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0
12	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0
13	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
14	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0
15	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0

Figure 10. Circuit DNA for one TGP[8]

The locations where green and yellow blocks meet in the table represent buffer combinations where instability could occur when compared across several FPGAs. Stanton's design for a system to generate a digital fingerprint based on the TGP involved cascading groups of TGPs in order to accumulate their delay to measurable levels.

Cascading the TGPs resulted in Circuit DNA being able to be generated for all TGPs in the system which would produce different results due to the accumulation of the delay.

2.4 Literature Review Summary

PUF designs and AFIT's Digital Fingerprint depend on physical random variations due to the integrated circuit manufacturing process. Several promising designs have already been proposed, but show that a trade off exists between stability on a single device, and distinguishability across multiple devices. Research already performed in the area of Digital Fingerprints provides a starting point for further research, providing a tunable probe for delay measurement.

III. Methodology

The following chapter explores the methodology behind three major elements of this research. These elements include the generation of Circuit DNA, Digital Fingerprints, and Digital Keys (DK). Circuit DNA as described in Section 2.3, is a method used to characterize the delay between paths on the FPGA which can be used as unique identifiers for the device. Digital Fingerprinting involves using the information obtained through Circuit DNA to create unique identifiers that rely on a specific input combination to the digital glitch probe as discussed in Section 2.3. Digital Key generation is a method which relies on the uniqueness of Circuit DNA to generate unique bit strings based on input values to the digital glitch probe which have been chosen in a specific manner, which can then be used as keys for polymorphic circuits, software, or encryption.

This chapter also introduces two performance metrics and explains their importance to this research. The first measure describes the repeatability on a specific FPGA of Circuit DNA or a digital key. This measure is known as stability, and is important to ensure that the same result can be achieved consistently. The other metric is called distinguishability; this measure ensures that a digital key is unique to a specific FPGA. Distinguishability is an important measure because it demonstrates a key's ability to uniquely identify a device, allowing for the possibility of unclonable software.

Chapter 3 also covers the experimental process used in this research and concludes with a summary of the methodology.

3.1 Problem Definition

3.1.1 Goals and Hypothesis

The goal of this research is to employ Circuit DNA extraction and Digital Fingerprint generation to produce distinguishable and repeatable digital keys. This goal is accomplished through the use of a Tunable Glitch Probe similar to design set forth by Stanton[**8**], to increase stability and allow for more user control. The hypothesis behind this work is that by measuring the delays between paths on an FPGA, the device can be characterized. This characterization can be utilized for both device identification, and for creation of unique device identifiers for use as encryption keys.

3.1.2 Research Approach

The desired outcome of this research is the creation and implementation of digital keys based on the physical random variations in the structure of an FPGA. This research uses these variations for the characterization of an FPGA's path delay which is used to generate unique device specific identifiers that can be used as encryption keys. Variation in path delay is demonstrated in Figure 11, where the arrows point at the locations where the signal at two paths is being measured, both of which have been provided with the same input. Both signals arriving at the two points simultaneously will result in no path delay being measured, but if the signal is measured on one path before the other, a delay is observed which can be used as an identifying feature. This approach of using path delay to characterize an FPGA is accomplished through three main phases; Circuit DNA/Key Generation, Comparison and Analysis, and Implementation.

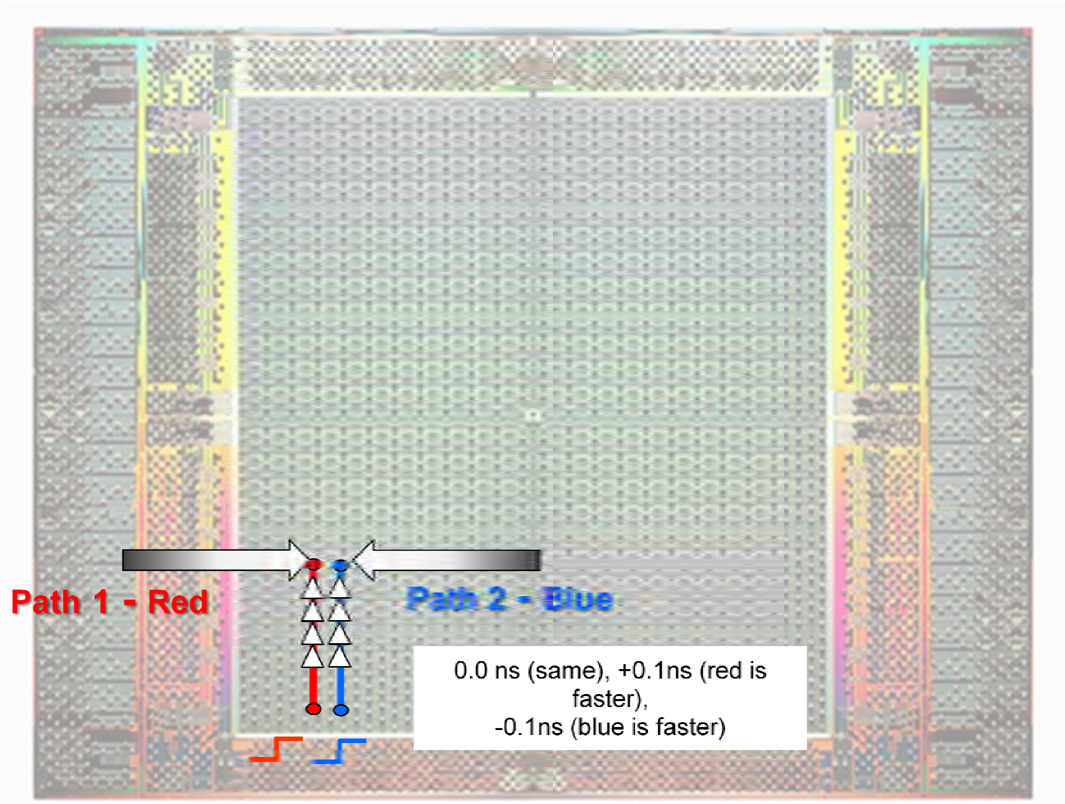


Figure 11. Measurement of Path Delay

3.1.2.1 Circuit DNA/Key Generation

Leveraging the hardware design created by Stanton in [**8**], this research provides new software using the response from the TGPs to generate Circuit DNA and digital keys. The software provides a method for individual buffer value selection on hardware system containing 128 TGP, something which was not available in past research efforts. Using improved software and a modified hardware design, Circuit DNA data is collected and analyzed. Digital keys are then generated based on the results of the Circuit DNA analysis.

3.1.2.2 Key Comparison and Analysis

Comparison of the digital keys is very important in order to determine stability and uniqueness of the keys. The purpose of this comparison is to ensure that the keys generated are suitable for encryption. For a key to be suitable for encryption it must have a high rate of repeatability on a single FPGA while also remaining unique to that device. A bit-wise comparison of each key generated against other keys generated both on the same and on different FPGAs will be made to ensure keys are acceptable. Using the metrics described in Section 3.5, each key will be checked for suitability.

3.1.2.3 Key Implementation

Keys deemed appropriate for encryption use are used in conjunction with an FPGA AES implementation [**29**][**30**] made available by the AFIT VLSI group. The purpose of this exercise is for a practical implementation of this research, and for verification of the digital key concept.

3.2 System Boundaries

The System Under Test (SUT) is the Circuit Identification system (CiDs), as shown in Figure 12. CiDs consists of three primary components: the tunable glitch probe, an FPGA, and the on-board PowerPC processor.

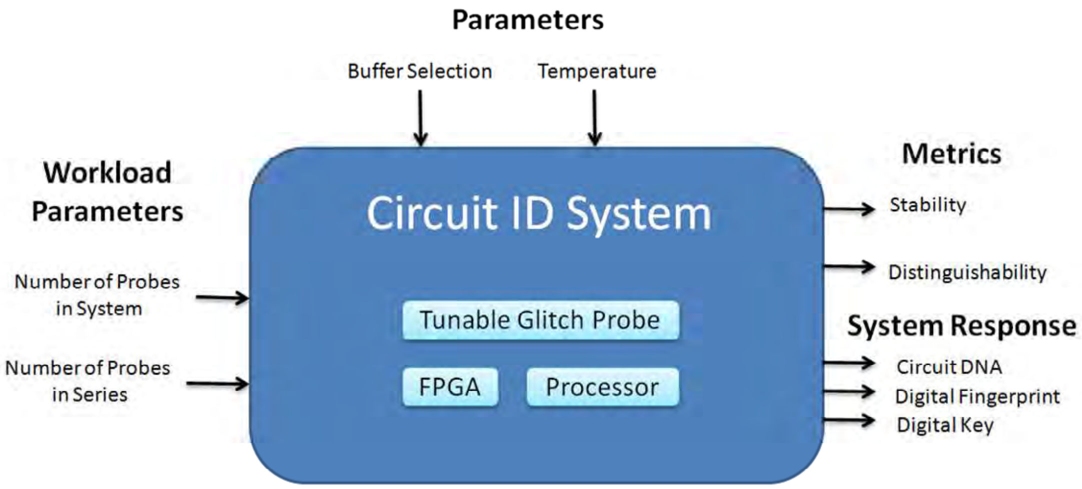


Figure 12. Circuit Identification System

3.2.1 Tunable Glitch Probe

The tunable glitch probes are the heart of the CiDs design. The probe is designed to take two user defined buffer values, which are then applied to two signals within the probe. When the delay on these signals is equalized at a comparator, the system reports a value of 1. When the signals arrive at separate times, a value of 0 is reported. The tunable glitch probe used in this research is a two level design similar to that used by Stanton[**8**]. The design implements an AND-OR architecture and a redesigned buffer component, which now uses registers to store the value rather than two inverters which result in two gate delays per inverter.

3.2.2 Xilinx Virtex-5 FX FPGA

The Xilinx Virtex-5 FX FPGA consists of an array of reprogrammable gates whose function is designated by the user. The FX series of the board also contains the PowerPC 440 processor which is also used by the CiDs.

3.2.3 Processor Core

The Virtex-5 FX FPGA [**31**] contains an on-chip PowerPC 440 microprocessor which is utilized for CiDs. The PowerPC, acts as an interface between the user and the array of tunable glitch probes, handling user input and sending control signals to the probes. The PowerPC is responsible for communicating with the peripherals in the system including the TGPs and the UART which facilities the user interface through the RS232 port.

3.3 System Services

The system used in this research provides the following three services.

3.3.1 Circuit DNA

Circuit DNA returns 128 16x16 tables which describe the delay characteristics of the tunable glitch probes on the FPGA under test, one 16x16 table for each probe used in the system.

3.3.2 Digital Fingerprint Generation

Digital fingerprint generation uses one specific user chosen buffer combination for delay on the upper and low paths for every probe in the system, and returns a 128-bit string representing the digital fingerprint.

3.3.3 Digital Key Generation

Digital key generation is similar to digital fingerprint generation, the only difference being that 128 buffer combinations are specified by the user, an upper and lower buffer value for each tunable glitch probe in the system.

3.4 Workload parameters

Workload parameters describe the requests for service within the system. These parameters describe the quantity and type of information that the system processes.

3.4.1 Number of Tunable Probes in System

The number of tunable probes used within the system directly affects the ability of the system to produce usable device specific keys. The number of probes used within the system is equal to the number of unique bits that can be used as a DF or DK. The number of probes used, also determines how much area of the FPGA that the system is using. Ideally, an implementation with the minimum space is desirable since it allows for more space to be used by other systems on the FPGA. A trade-off exists between the system footprint and the number of unique bits generated by the system. Using too few probes can result in a DF or DK which contains too few bits, increasing the possibility that another FPGA will generate the same DF or DK. For example if only ten probes are used, there are 1,024 possible DF or DKs that can be generated, if a population contains 1,025 boards, we know by the pigeon hole principle, that two of the boards must generate the same DF/DK. For this research, 128 probes are used in the system, producing the possibility of 2^{128} possible combinations for DFs/DKs, far more possibilities than FPGA's in existence.

3.4.2 Number of Tunable Probes in Series

The number of tunable probes in series affects the delay in the signal entering each probe. If too much delay exists between two signals before they reach a probe, the ability to equalize the delay is diminished. When too many probes are placed in series, the accumulation of delay within the system becomes too great for an individual probe to overcome, resulting in a probe that produces a constant value for all buffer input combinations. Stacking too many probes in series can cause enough skew between a pair of paths which cannot be balanced with only fifteen probes in each path. Limiting the number of probes being used in series allows for uniqueness in keys as described in Section 3.4.2. For this research, the 128 probes being used are divided into partitions of 32 probes in series, resulting in four groupings of 32 probes. This grouping was based off of observations made on research by Stanton [**8**], resulting in a design which for most circuits placements (discussed in Section 3.7), results in probes which produce non-constant results. Using 32 probes in series ensures enough cumulative delay has built up on paths being measured in probes on individual chain to allow for unique delay characteristics to be observed, but not so much cumulative delay that the fifteen possible buffers on the paths in each probe are unable to equalize path delay. Adding more probes in series in the system has shown to result in path delays which cannot be equalized, while using too few probes in series results in lack of distinguishable characteristics in Circuit DNA.

3.5 Performance Metrics

Two metrics are being used to characterize the acceptability of a both DFs and DKs, these metrics are used to compare multiple DFs and DKs to one another. The two metrics being used are a test for stability to ensure that a DF or DK on a given device is consistent, and a test for distinguishability to determine if DFs and DKs on multiple boards vary sufficiently enough to fall within defined bounds of acceptability.

3.5.1 Stability

A measurement of stability is a reference to how consistent a DF or DK is when generated multiple times on the same FPGA. Ideally, an FPGA will always produce the same DF and DK if given the same buffer input values. A stability measurement is presented as a percentage of bits which are similar in two given DF or DK values. DF and DK values generated on the same board may not always be exactly the same due to environmental conditions such as temperature and radiation. Stability across a temperature range is tested through the use of the ESPEC BTZ-133 Temperature Chamber. This piece of equipment, shown in Figure 13, allows for data to be collected over a temperate range for stability comparison. Temperature variation is collected for an ambient temperature range of 0–85 degrees Celsius. For a DF to be considered stable, it must have a stability measurement of 90% or greater when multiple readings are compared.

Two types of temperature stability will be tested. Limited Temperature Range Stability (LTRS), tests for stability over normal operating conditions at room temperature (20 degrees Celsius) plus or minus 10 degrees. Full Temperature Range Stability (FTRS), tests for stability of a DF or DK over a temperature range of 0-80 degrees Celsius, simulating extreme environmental variance. Ideally a DF and DK should have FTRS, but due to changes in transistor response based on operating temperature this is not always the case. At the bare minimum, a DF or DK must be LTRS in order to be usable.

Figure 13 ESPEC BTZ-133 Temperature Chamber

3.5.2 Distinguishability

Distinguishability is measurement of how different one DF or DK is from another. DFs and DKs generated on different boards should be sufficiently different from one another in order to avoid the possibility of another board generating the same DF or

DK. A distinguishability measurement consists of a bit-wise comparison of two measurements, reporting the percentage of bits which are dissimilar, essentially a Hamming distance expressed as a percentage. The distinguishability measurement can be found in equations (3.1) and (3.2).

$$\%\,Distinguishability = \frac{B_{Total} - B_{Similar}}{B_{Total}} * 100 \tag{3.1}$$

$$\%Distinguishability = 1 - \left(\frac{B_{Similar}}{B_{Total}}\right) * 100 \tag{3.2}$$

For example, if two DKs consisting of 128-bits have 42 bits in common, they would have a distinguishability of ($\frac{128-42}{128} * \ 100$) = 67.2%. For the purposes of this research, in order for a DF or DK to be considered distinguishable, it must present a distinguishability of at least 10%, real world implementation would preferably be much greater.

3.5.3 Stability versus Distinguishability

Stability and distinguishability are both important measurements which affect the ability to use a generated DF or DK, but there is a relationship between the two which must be kept in balance. While it is very easy to pick buffer values which result in a very stable DF or DK, it most likely won't result in a key which is distinguishable. For example, using the Circuit DNA from a single probe shown in Figure 14, a choice of seven buffers on the top path and thirteen buffers on the bottom path (marked in red), will most likely be a very stable choice, in that no matter how many times a DF or DK is generated on this FPGA, the result will be a 0 for this probe. This effect is due to the buffer location not being near the transition line for this probe; this buffer choice will most likely always result in a 0 on any FPGA because the transition area on Circuit DNA tends to be in the same general area on all FPGAs. This buffer selection results in a very stable choice, but also an indistinguishable bit.

The opposite situation is also true, a buffer combination choice my be very distinguishable, but results in poor stability. Figure 15 shows a buffer choice which is probably very distinguishable, but most likely is very unstable. A choice of three buffers on the bottom path, and six buffers on the top path results in an inconsistent reading from the probe, 60% of the time it reads a value of 1. This buffer location is likely to fluctuate over many runs, causing poor stability, but it is a distinguishing characteristic of the DNA for this probe, and is in a good location for DF and DK generation, on a transition point.

Number of Buffers on Bottom Path

Number of Buffers on Top Path

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	10	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0
4	10	10	10	10	0	0	0	0	0	0	0	0	0	0	0	0
5	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0	0
6	10	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0
7	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0	0
8	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0	0
9	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0	0
10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
11	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0
12	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0
13	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0
14	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
15	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0

Figure 14. Poor Distinguishability Choice

Number of Buffers on Bottom Path

Number of Buffers on Top Path	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	10	10	10	$\boxed{6}$	0	0	0	0	0	0	0	0	0	0	0	0
7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	0	0
8	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0	0
9	10	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0
10	10	10	10	10	10	10	10	0	10	0	0	0	0	0	0	0
11	10	10	10	10	10	10	10	10	10	9	0	0	0	0	0	0
12	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0
13	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0
14	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
15	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0

Figure 15. Poor Stability Choice

3.6 System Parameters

System parameters if varied have an effect on the metrics or response of a system; the parameters of CiDs are described below.

3.6.1 Buffer Selection

Buffer selection is the main input to the system from the user. The selection of buffers controls the delay introduced to each probe, which is applied to the signals within the probe, affecting the output of each individual probe. The legal range of buffer values for both the upper and lower paths is from zero to fifteen. This allows for a total of 256 possible buffer combinations.

3.6.2 Temperature

Core temperature of the SUT has an effect on the response of the transistors in the system. Core temperature is affected by the ambient temperature around the SUT, and by controlling the ambient temperature, measurements can be made at different core temperature values.

3.7 Factors

The factors within the system that are varied for this research are detailed below.

3.7.1 Design Placement

The location of the TGPs in the design directly influences the response of CiDs. The placement on the FPGA dictates which transistors are being used, and thus affect the delay in each individual probe response used to generate the Circuit DNA, DF or DK. The Xilinx PlanAhead tool was used to create several design placement options on the FPGA, these placement options were then compared to choose placements which didn't result in probes which produced constant values.

3.7.2 Temperature

The core temperature affects the response of the transistors on the FPGA resulting in varying transistor delay. Normally, higher core temperature results in slower response from the transistors. Core temperature is affected by the ambient temperature around the FPGA. For this research, data is collected across a temperature range of 0-80 degrees Celsius.

3.8 Evaluation Technique

The evaluation technique used in this research is measurement on real hardware. The experiment set up consists of several (3-5) Virtex-5 FX ML507 FPGA Development Boards which are running CiDs, a notebook computer connected to the FPGA via an RS232 connection for communication with CiDs and connected to the JTAG interface on the ML507 board via a Xilinx Development Board Programmer over USB, used to program the FPGA. This system is then contained within the ESPEC BTZ-133 Temperature Chamber, in order to collect data over a temperature range as described in Section 3.7.2. Data collected is transmitted to the notebook via an RS232 connection over the Hyper Terminal. This data is stored on the notebook for comparison and analysis.

3.9 Experimental Process Overview

The process by which this research is carried out is detailed in the chart presented in Figure 16. The process begins with the design of the CiDs system, both hardware and software design. It continues with Circuit DNA generation and comparison to ensure proper design placement. Once an acceptable design placement has been achieved, Circuit DNA is generated for several FPGAS and used to create input buffer selections for the creation of digital keys. Finally digital keys are generated on several FPGAs at varying temperatures and compared in order to test stability and distinguishability of the keys. This process is described in more detail in the following sections.

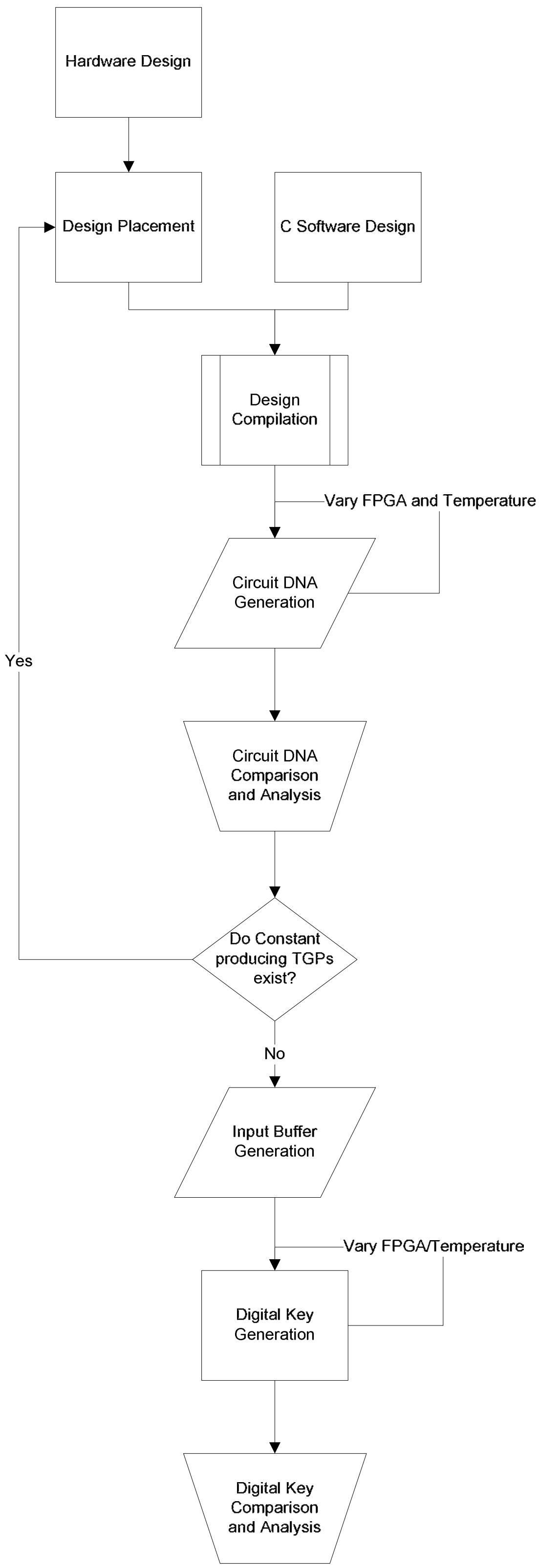


Figure 16. Experimental Process

3.9.1 Hardware Design

The CiDs system was designed within the Xilinx EDK Platform Design Studio version 12.4. An overview of the Hardware design can be seen in Figure 17. The hardware design utilizes the PowerPC 440 core on the FPGA, as opposed running a soft-core MicroBlaze processor available as a programmable design by Xilinx. The PowerPC processor is used as a communication interface between the other hardware elements used in the design and the software. The other hardware elements used in the design are contained in user defined pCores connected to the PowerPC via a Processor Local Bus (PLB).

The main backbone of the CiDs system lies in the user created core containing TGPs previously discussed. This pCore contains the major hardware design required for this research. The design is created using VHDL with input and output ports connected to the PowerPC via the PLB. The design includes 128 TGPs in four groupings of 32 cascaded probes as shown in Figure 18. This design allows for the input signal to propagate through 32 probes while also providing output glitch readings at each probe. The cascading of the probes allows for the accumulation of delay in the input signal as it propagates through the probes resulting in different amounts of delay in the signal as it is accessed by each probe.

The hardware design also includes two additional pCores. These pCores include a UART interface and a BRAM controller. The UART interface is used to facilitate user input between the hyper-terminal and the PowerPC core. The BRAM controller facilitates memory access to the Block RAM located on the Xilinx development board, used to store the device software and system variables.

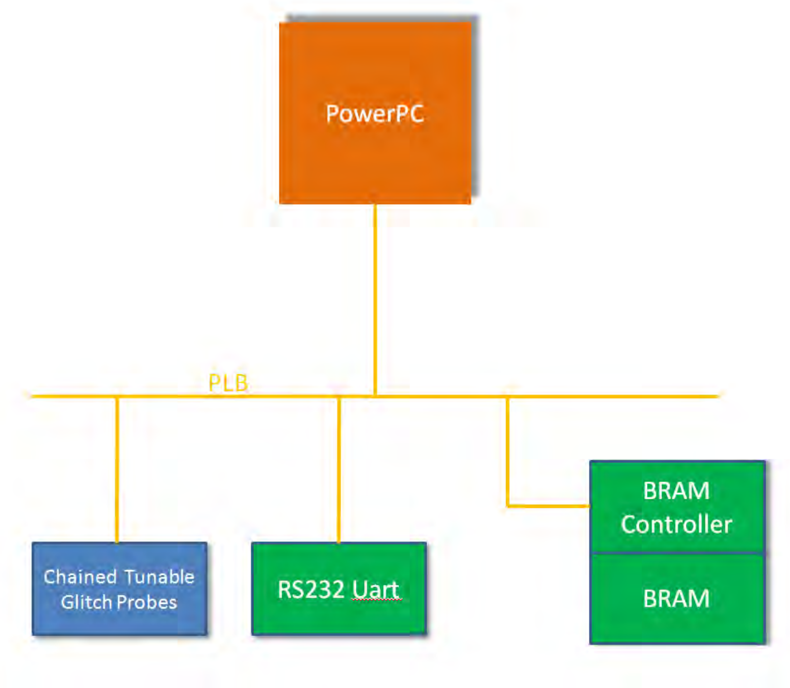


Figure 17. Block diagram of the CiDs design

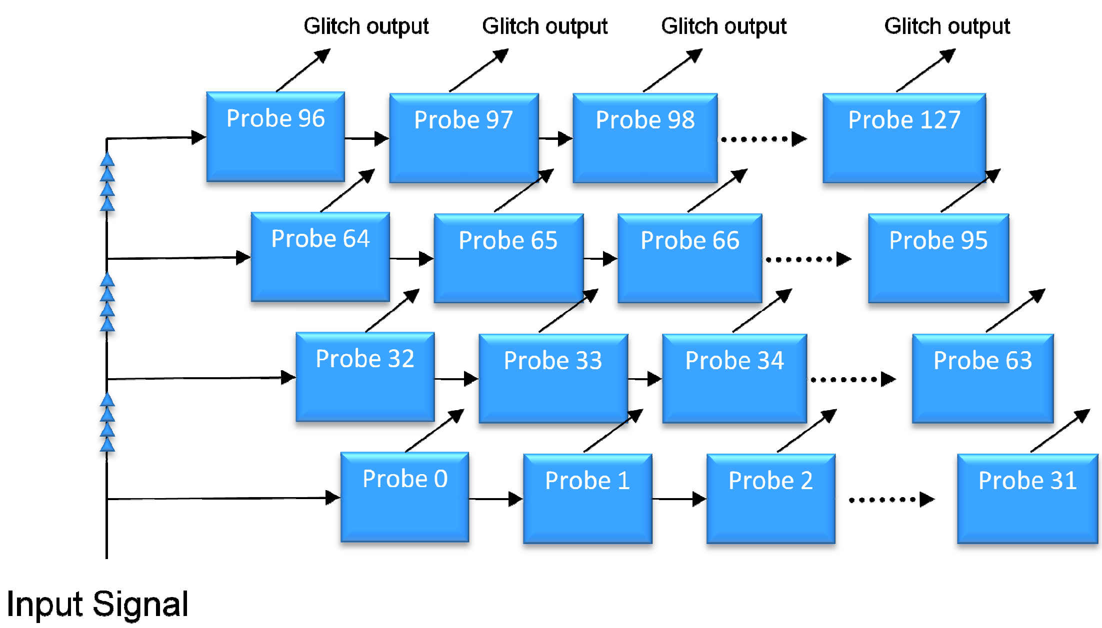


Figure 18. Tunable Glitch Probe connection design

3.9.2 Software Design

The software phase of the design facilitates interfacing with the registers of the chained_design_selectable pCore, and the generation of Circuit DNA and Digital Keys. The software is written in C and runs on the PowerPC. The main tasks initiated the software include; initiation and control of the input signal sent to the TGPs, data collection and user interface. The software instructs the PowerPC to place a signal on the PLB that strobes the input signal to the TGPs, and then handles the corresponding output from the TGPs. Once the data is collected, it is stored in a three-dimensional array and formatted into the Circuit DNA, or used to produce digital keys based off of additional user input for input buffer values.

3.9.3 Design Placement

The placement of the hardware design on the FPGA is crucial to the distinguishability of Circuit DNA and digital keys. Using the PlanAhead software included in the Xilinx ISE Design suite, the design was placed in several locations on the board with the effects on Circuit DNA being observed after compilation. The details of this step are discussed in Section 4.2.3.1.

3.9.4 Design Compilation

Compilation of a design created in Xilinx EDK is usually a straight forward process consisting of simply pressing a button which automatically generates a netlist of the design, then using that netlist to create a bitstream that programs the FPGA with the hardware design. The hardware bitstream is then merged with the software and can then be used to program the FPGA with both hardware and software. Since this research involved manually placing the hardware design on the FPGA using the PlanAhead tool, compiling the hardware and software was not as straight forward. When using PlanAhead, a bitstream is generated containing only the hardware design that has been manually placed. This bitstream cannot be combined within EDK with the written C code; this requires the use of a command line tool called Data2MEM [**32**][**33**]. This tool takes both a hardware bitstream and the compiled software as inputs and creates a bitstream which can be used to program the FPGA. This bitstream must them be downloaded to the FPGA via the iMPACT tool included in the Xilinx ISE Design Suite. The bitstream generated via the Data2MEM command line tool, once programmed on the FPGA can be used for testing and data collection.

3.9.5 Circuit DNA Generation, Comparison and Analysis

Circuit DNA was generated not only for data collection and analysis purposes but also to determine if the hardware placement on the FPGA was a good choice. Several placements were tested, with Circuit DNA being generated for each layout. The Circuit DNA was checked for probes which produced constant values, due to routing delays. If constant producing probes were found the placement was adjusted and retested.

Once a placement was decided on which did not produce constant probe readings, Circuit DNA data was collected for stability and distinguishability comparison on several FPGAs at varying temperature settings. These data sets were then compared using a custom perl script which compared individual table entries within Circuit DNA to report statistics on the stability and distinguishability of the comparisons. The perl script used can be found in Appendix A.1.

3.9.6 Input Buffer Generation

Input buffer generation for use as selection values for Digital Keys is accomplished through the use of another Perl script. This script takes in an input of a comparison report from the Perl script used for Circuit DNA comparison and then generates two 128 character hex strings to be used as input buffer selections for the TGPs. This Perl script chooses a value which is not constant between multiple boards and assigns it as an input selection value, if there are no non-constant locations, it will randomly choose a buffer value between 0 and 15. This script can also be found in Appendix A.2.

3.9.7 Digital Key Generation, Comparison and Analysis

Using the generated input buffer selection, the user is able to manually input the value into the CiDs system via the hyper-terminal to produce digital keys. The input buffer strings are parsed and used as lookup variables in the Circuit DNA that has been generated for the particular FPGA under test. Digital keys are generated across a temperature range on several FPGAs and then compared for stability and distinguishability using a Perl script which performs a bitwise comparison on a list of keys reporting back the number of matching bits, along with a stability and distinguishability measurement. This script available for review in Appendix A.3.

3.10 Acceptable Parameters and Expected Results

In order for Circuit DNA and digital keys to be considered successful and usable they must meet certain stability and distinguishability requirements. In order for Circuit DNA to be considered acceptable for use for digital key generation it must be at least 90%

stable at a limited temperature range of 20°±10°C. Digital Keys on the other hand must be nearly 100% stable and have a distinguishability of greater than 10%.

The expected results of this research coincide with the acceptable parameters. Previous work with Circuit DNA by Stanton [**8**], has already shown that circuit DNA is already about 90% stable, and with modifications made to the TGP, improvement is expected of around 5%. Stability of digital keys is expected to be 99.5% and 99.9% to account for possible error. Distinguishability of digital keys is expected to vary widely between about 15% and 70% due to input buffer selection.

3.11 Methodology Summary

This chapter defines the experimental methodology for the Circuit Identification System. This research focuses on improving the Circuit DNA and Digital Key extraction through modification and more in depth data analysis. The SUT is clearly bounded, along with the system parameters, and its factors. The performance metrics of DF and DKs are clearly defined with values required for acceptance. An evaluation technique and experimental design has been described involving the test of CiDs on multiple hardware devices.

IV. Results

This chapter discusses and analyzes the data collected from the experiment detailed in Chapter 3. The chapter begins with an explanation of the experimental set up and then describes the individual tests performed and the results obtained.

4.1 Experimental Setup

Figure 19 depicts the hardware setup used for this thesis. The Xilinx Virtex 5 ML-507 Development board is placed within the ESPEC BTZ-133 Temperature Chamber, and connected via USB and RS-232 to a notebook computer used for programming the development board, and as a user interface to the system. A lab power supply was used to ensure that stable voltage was supplied to the board. A digital multi-meter was also used in conjunction with a temperature probe to monitor ambient temperature within the ESPEC chamber.

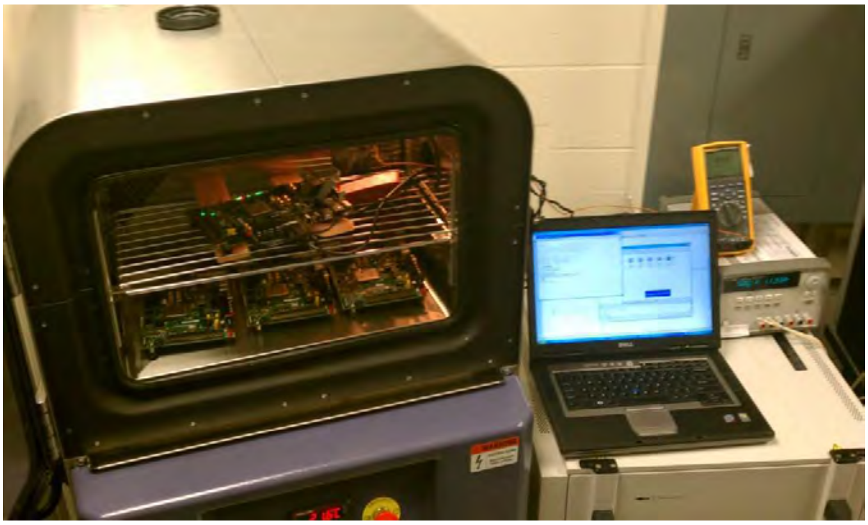


Figure 19. Test Setup

Figure 20 shows the ML-507 Development board and its external connections. A ribbon cable connects the USB programmer to the 16-pin JTAG connector on the board used for programming the FPGA and for monitoring Core Temperature. A RS-232 connection is used to connect the board to a serial port on the notebook computer, enabling user input and data collection via the hyper terminal.

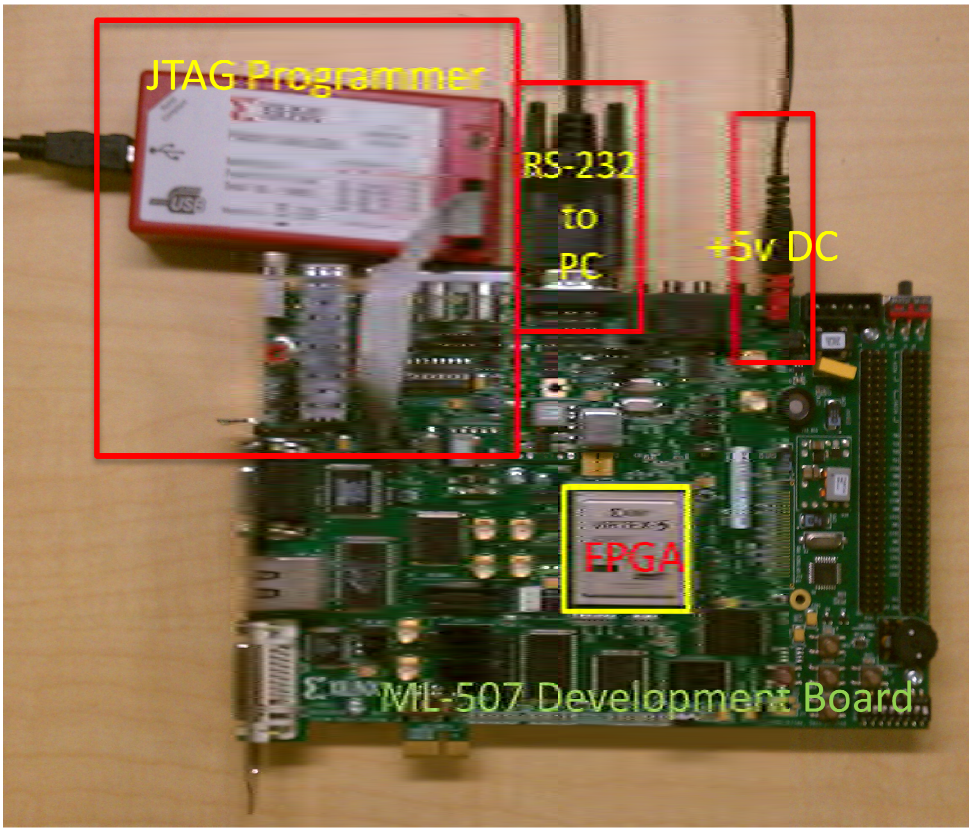


Figure 20. ML 507 Development board with connections

4.2 Circuit DNA

Circuit DNA was used as a basis for all of the experimentation performed for this thesis. Circuit DNA was produced and then used as a tool for generating digital keys. To ensure that Circuit DNA was suitable for key generation, it had to be tested for stability and distinguishability.

4.2.1 Stability

To determine the stability of Circuit DNA , it was generated on an FPGA across a large temperature range. Multiple sets of Circuit DNA were produced at each temperature, and these were compared with other readings at the same temperature to determine stability at individual core temperatures. Figure 21 shows results for this test for one board, the data points represents what percentage of the individual probe entries were different in the comparison. Since the CiDs system includes 128 TGPs, and there are 256 table entries for each of the TGPs, resulting in a total of 32,768 entries. With such a large number of possibilities, even a couple hundred entries being different still only translate to a fraction of a percent. The large number of entries being compared positively affects stability by allowing many entries to be different without having and adverse affect.

The values from Figure 21 when compared to the total number of entries and averaged together result in a stability of 99.86%. This value is much higher than expected, yielding a positive result. Stability near 100% is ideal for Circuit DNA since it reflects very little variation at a stable temperature, resulting in very few unstable entries which could possibly be chosen for digital key generation.

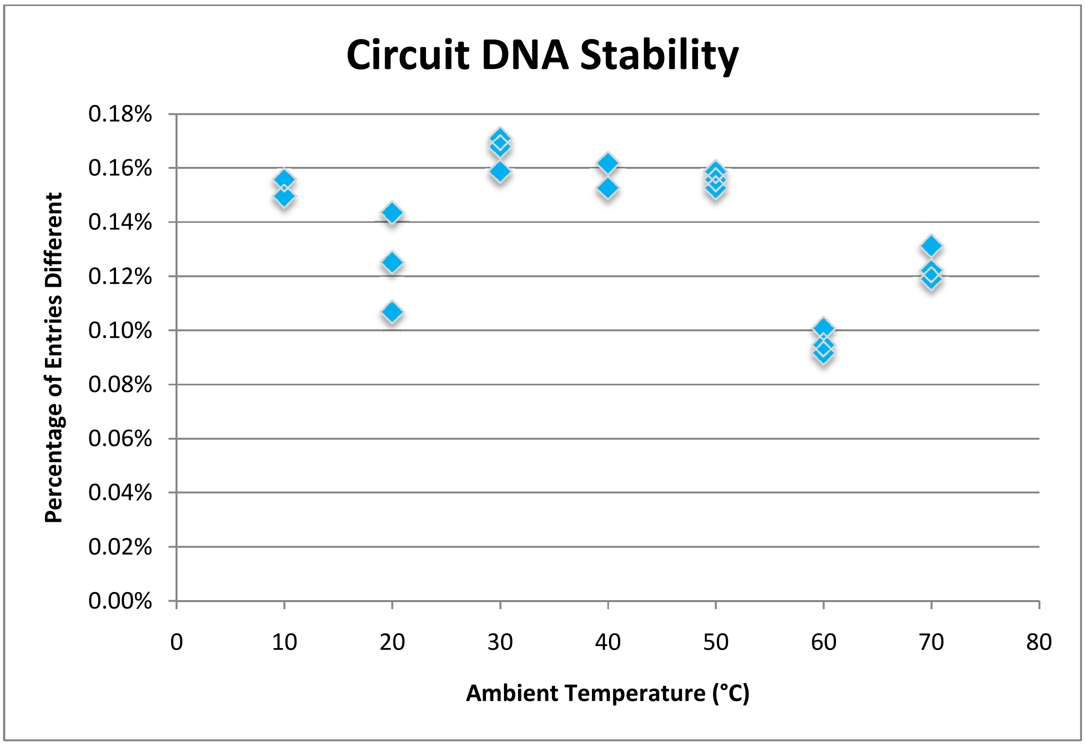


Figure 21. Graph representing percentage of different Circuit DNA entries across a temperature range

4.2.2 Temperature Range Results

Using a similar approach to that found in Section 4.2.1, comparisons can be made between Circuit DNA at different temperature values. Figure 22 details the change in one Circuit DNA probe over a limited temperature range. Notice that only one of the entries changes in this probe over a 20 degree range (outlined in red).

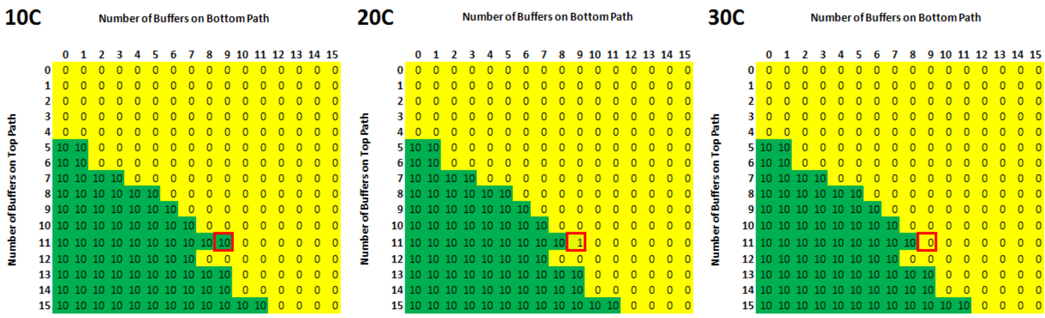


Figure 22. Circuit DNA effect of temperature change, a red square denotes the changing value.

A comparison of Circuit DNA produced at an ambient temperature of -20°C (core temperature of 2.0°C) and 70°C (core temperature of 108.5°C) resulted in 705 entries being different, which is still a stability of 97.9%. At a more limited range, the results are even better, at both an ambient temperature range of 10-30°C and 10-70°C average stability is 99.6%. These results are much better than expected and are much improved versus results of Stanton's implementation in [**8**]. This high level of stability across a wide temperature range ensures the stability of digital keys across a wide temperature range assuming that ideal input buffer values are chosen.

Figure 23 details the effect of temperature as it changes across a wide temperature range. The points on the graph represent the number of Circuit DNA entries which have changed between two temperature readings. As you can see there seems to be no correlation between the number of entry changes and the temperature, each board reacts in a different way. This test also shows that across a temperature range the number of entry changes is very low, a couple hundred changes out of 32,768 total entries, resulting in very high stability. Additional Circuit DNA temperature stability results can be found in Appendix B.

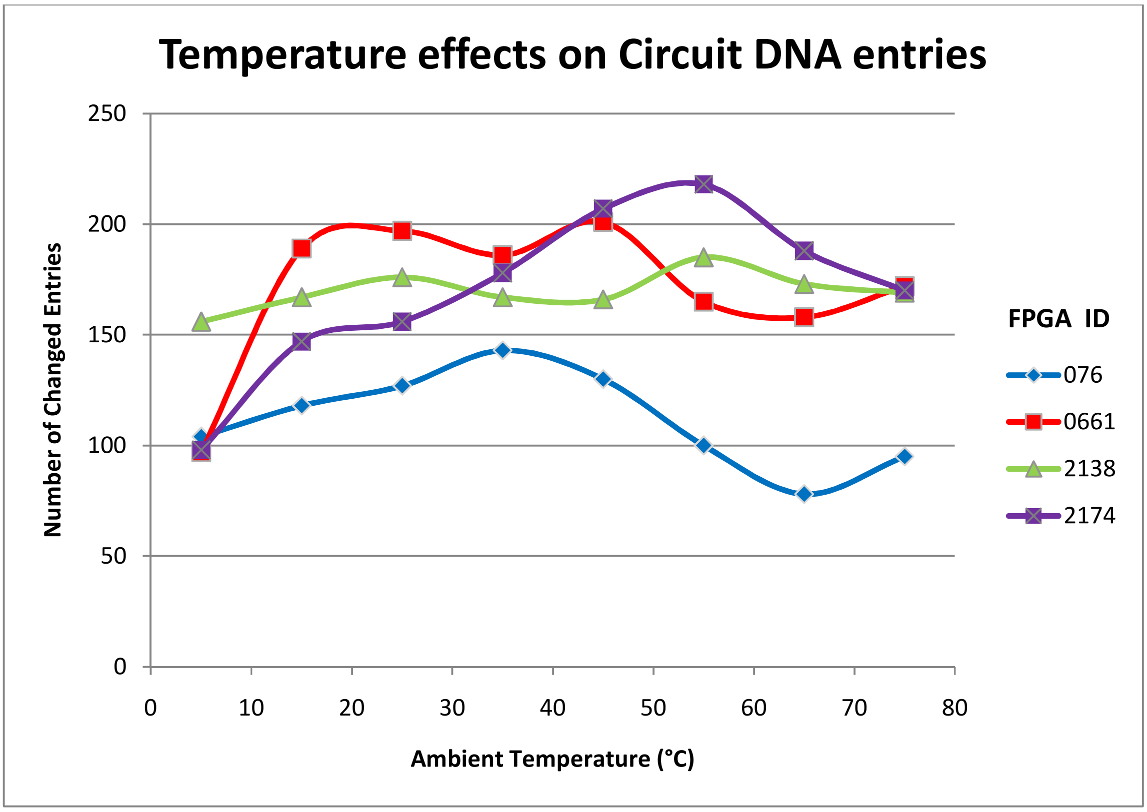


Figure 23. Graph representing the number of changed DNA entries across a temperature range

4.2.3 Distinguishability

Circuit DNA had to be generated on several boards and compared to one another in order to ensure against possible collisions which could occur due to another board producing the same set of Circuit DNA. Theoretically, a collision is very unlikely to occur since each circuit DNA report consists of 128 tables containing 256 entries resulting in 256^{128} $(1.79 * 10^{308})$ possible combinations. The main challenge present to ensure distinguishability is assuring that none of the probes in the system produce a constant value; this is achieved through design placement.

4.2.3.1 Design Placement

Location of the design on the FPGA proved to be very important to circuit DNA which was distinguishable across multiple boards. The Xilinx PlanAhead tool was used to create multiple design layouts for testing. The layouts were individually tested before settling on a final test layout. For each of the designs, the 128 probes were divided into four 32 probe units, these units were then placed on the FGPA with the goal of measuring the path delay in several areas of the FPGA. Measuring the delay in several areas of the FPGA is important for future implementation to ensure against device tampering. Many of the designs tested resulted in probes which produced all ‘0’ readings. These constant producing probes were caused by too much path delay between probes. This excessive path delay is due to the routing of the connections on the FPGA being too long. Probes measuring these delays are unable to balance the delay between the two lines, resulting in a ‘0’ reading for all locations. Figure 24 shows placement designs which were rejected due to excessive routing delay. The final design used for testing is shown in Figure 25. Notice how in these designs there are four distinct blocks, these each contain 32 chained TGPs. The design in Figure 25 reduces the distance between blocks of probes, resulting in less routing delay, while also spreading the probes across multiple areas of the FPGA.

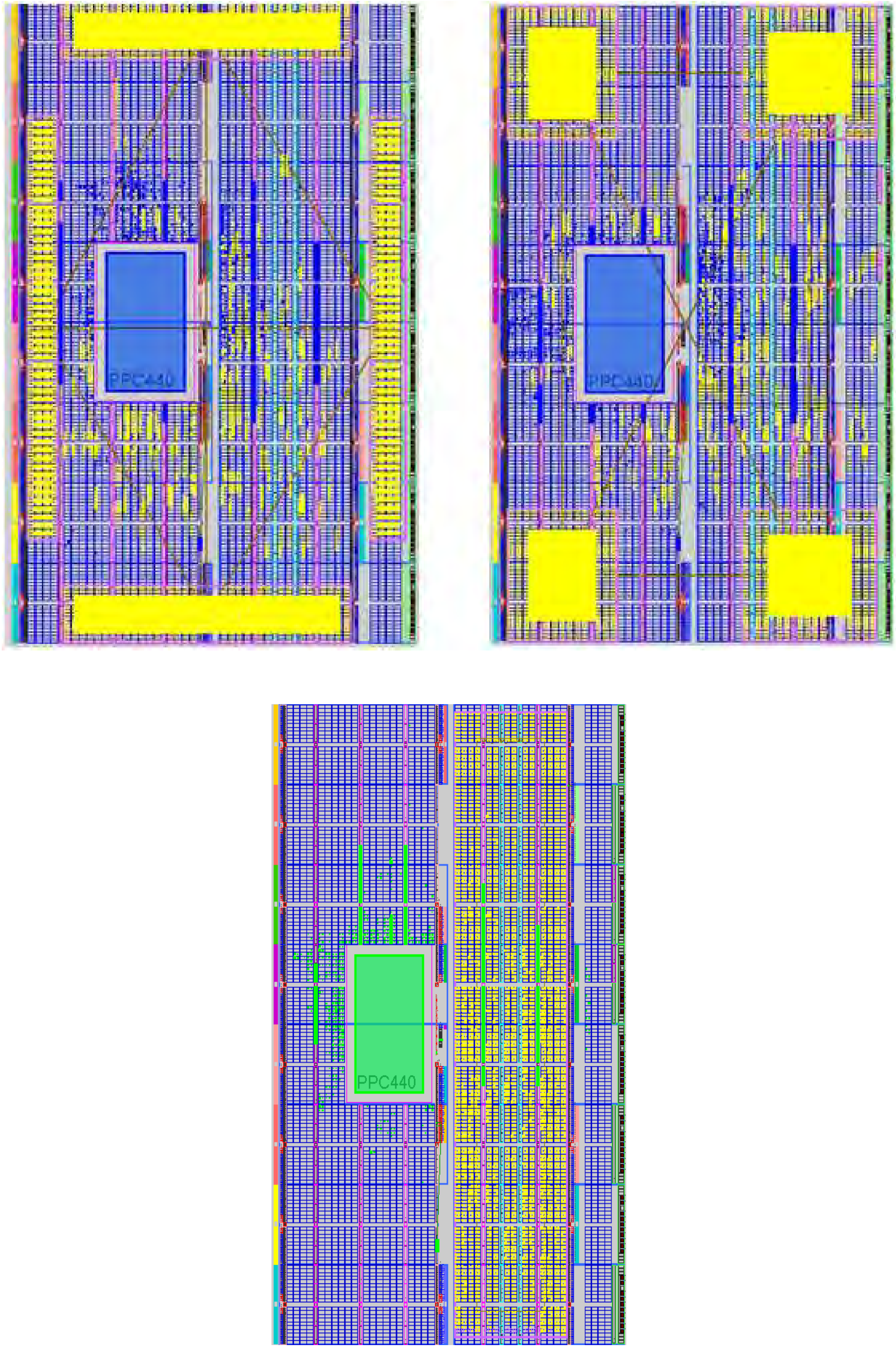


Figure 24. Three rejected circuit placement options

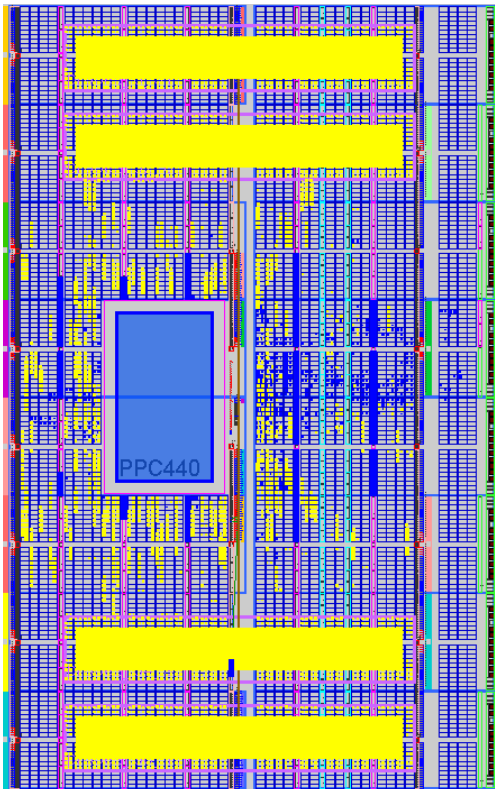


Figure 25. Design placement used for testing

4.3 Digital Key

4.3.1 Choosing Buffer Selection Values

As discussed in Chapter 3, each of the TGPs used in the system must be provided with two buffer selection values to set the delay on the upper and lower paths. The determination as to which buffers must be chosen has proven to be one of the most difficult challenges in this research. Buffer values must be chosen in such a way that they tend to fall on transition areas on the Circuit DNA tables. A transition area is the area on the Circuit DNA graph where readings change from displaying glitches to being stable with no glitches. These areas generally include the two points on the table on either side of the division line between green and yellow blocks. An example transition area can be seen in Figure 26 in red.

Number of Buffers on Bottom Path

Number of Buffers on Top Path	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	10	10	10	0	0	0	0	0	0	0	0	0	0	0	0	0
5	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0	0
6	10	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0
7	10	10	10	10	10	10	0	0	0	0	0	0	0	0	0	0
8	10	10	10	10	10	10	10	0	0	0	0	0	0	0	0	0
9	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0	0
10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0
11	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0	0
12	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0	0
13	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0
14	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0
15	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0

Figure 26. Circuit DNA Table showing Transition Area

A transition area is not necessarily the same on every FPGA since circuit DNA is different between boards. The problem in choosing buffer values is that we have no idea which point will provide the most distinguishable results on all boards. We can make an educated guess as to which points will provide good buffer values based on data collected on several boards. One such approach to making educated selections involved attempting to determine the slope of the diagonal line formed by the transition area and choosing points along that line as selection values. This approach was abandoned early in the design process due to the irregularly of the shape of the transition area in many probes which resulted in selection values that weren't in the selection area,

Figure 27 shows the approach taken which makes an entry by entry comparison of multiple FPGAs. The figure shows the Circuit DNA tables for the same probe on three different FPGAs, a fourth table shows which values on these boards are different from the first. The values which are different fall in the transition area of one or all of the boards. We know that by choosing one of these locations as the buffer selection values for this probe, the bit represented by this probe will not be the same on all three of these FPGAs. This process is fairly simple for 2 or 3 boards, but in order to produce keys which are distinguishable across many boards, we must make educated guesses by making comparisons between several boards and choosing the values which are different most often. For this research, digital keys were produced on four different FPGA's using buffer selection values produced through such a comparison.

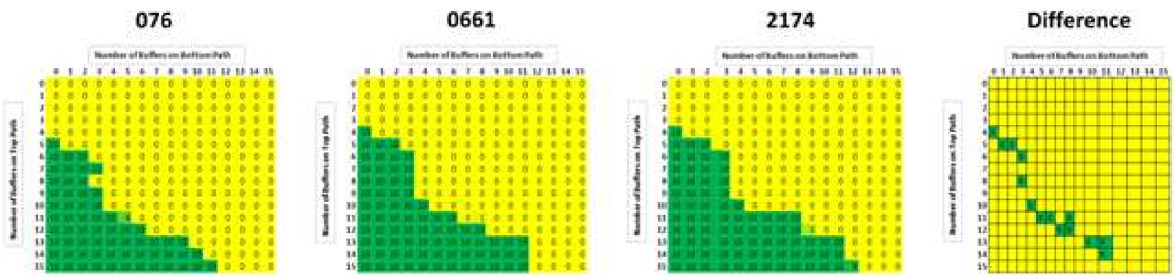


Figure 27. Comparison of multiple Circuit DNA tables for choosing input buffer values

4.3.2 Digital Key Stability

The most important test performed for this research had the purpose of determining the stability of digital keys generated. This test was performed by generating and comparing multiple keys generated using the same input buffer selection values on the same FPGA. The initial tests performed indicated that the keys being generated were not as stable as would be needed for use as a digital key. These initial tests showed that multiple keys generated on an individual board using the same input buffer values were between 98-99% stable, this translates to the possibility of one to two bit difference for a 128-bit key. While many of the keys generated were exactly the same, occasionally a generated key would contain one or two flipped bits.

Ideally in order for a key to be usable it needs to be repeatable 100% of the time. In order to combat this issue of stability, a polling scheme was implemented in the software with the goal of eliminating the occasional flipped bits. The implementation of this polling scheme involved generating several keys and then performing a bit-wise comparison of the keys using the bit value which occurred more than half the time. The Law of Large Numbers [**34**](sometimes referred to as the "Law of Averages"), would suggest that if enough keys are sampled and compared, a usable key could be generated that is always the same. As this is not an ideal world, the possibility for error always exists, therefore their always exists the possibility that a bit-flip may occur. For testing purposes seven samples were taken and compared for each key generated. This number was chosen due to the amount of time required to generate the samples. A tradeoff needed to be made between stability and runtime, collecting more samples would theoretically result in keys which are more stable, but at the cost of increased runtime.

4.3.2.1 Constant Temperature Testing

The first test performed on keys generated using the majority polling scheme described above was to measure stability at a constant temperature. A total of seven keys were generated and polled to create a single digital key, this was performed ten times for each key and then a bit-wise comparison was made between each key. This comparison generated a percentage of bits which were the same. These percentages were then averaged together to produce a final stability rating expressed as a percentage. A sample report generated is shown in Table 1. As can be seen from the data, the average stability of the FPGA with serial number ending in 0661 is 99.6%. This process was repeated at several different ambient temperature settings on the ESPEC Temperature Chamber, with the FPGA core temperature being closely monitored.

Figure 28 is representative of most of the boards tested. In most cases the keys generated were between 99% and 100% stable, with an instability occurring at a core temperature of about 45°C. Be aware that these results reflect only stability in a bit-wise comparison between keys generated at a constant temperature, and are not a reflection of comparison with keys generated at other core temperature values.

Table 1. Stability Report for FPGA 0661 at 31°C

Keys being Compared	Percentage of Matching Bits	Number of Matching Bits
0|0	100.0%	128
0|1	100.0%	128
0|2	99.2%	127
0|3	99.2%	127
0|4	99.2%	127
0|5	100.0%	128
0|6	100.0%	128
0|7	99.2%	127
0|8	99.2%	127
0|9	100.0%	128
1|0	100.0%	128
1|1	100.0%	128
1|2	99.2%	127
1|3	99.2%	127
1|4	99.2%	127
1|5	100.0%	128
1|6	100.0%	128
1|7	99.2%	127
1|8	99.2%	127
1|9	100.0%	128
2|0	99.2%	127
2|1	99.2%	127
	.	
	.	
	.	
6|9	100.0%	128
7|0	99.2%	127
7|1	99.2%	127
7|2	100.0%	128
7|3	100.0%	128
7|4	100.0%	128
7|5	99.2%	127
7|6	99.2%	127
7|7	100.0%	128
7|8	100.0%	128
7|9	99.2%	127
8|0	99.2%	127
8|1	99.2%	127
8|2	100.0%	128
8|3	100.0%	128
8|4	100.0%	128
8|5	99.2%	127
8|6	99.2%	127
8|7	100.0%	128
8|8	100.0%	128
8|9	99.2%	127
9|0	100.0%	128
9|1	100.0%	128
9|2	99.2%	127
9|3	99.2%	127
9|4	99.2%	127
9|5	100.0%	128
9|6	100.0%	128
9|7	99.2%	127
9|8	99.2%	127
9|9	100.0%	128
Average Stability:	**99.60%**	

4.3.2.2 Temperature Effects

Comparing the results gained in Section 4.3.2.1 with one another paints a picture of stability across a temperature range. Performing a bitwise comparison of the ten keys generated at each temperature demonstrates the practically of key generation in both a limited and large temperature range. Unfortunately the results of the testing for this research did not show promise outside of a very limited operating window. On average the stability of the digital key was in the low 90's for an ambient temperature range of 20°C ±10°C, with measurements taken in 5° intervals. The results from four of the boards tested can be seen in Figure 29. As you can see in the figure, one of the boards tested actually had an average stability of 89.7%, much lower than the others.

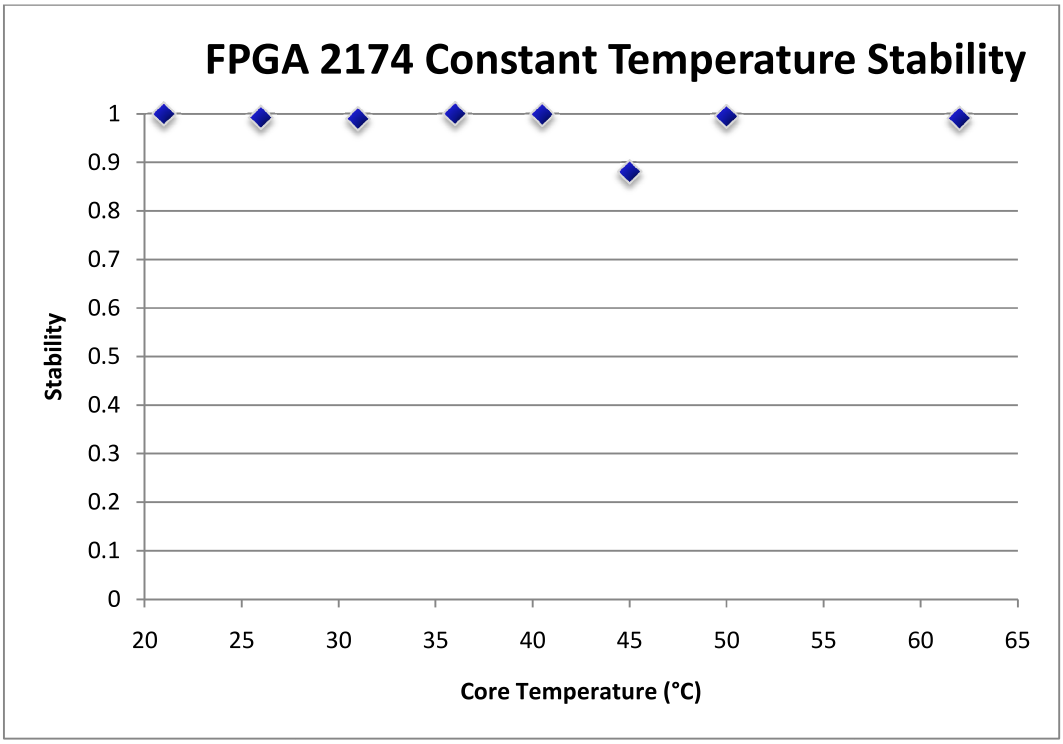


Figure 28. Graph detailing stability readings across a temperature range

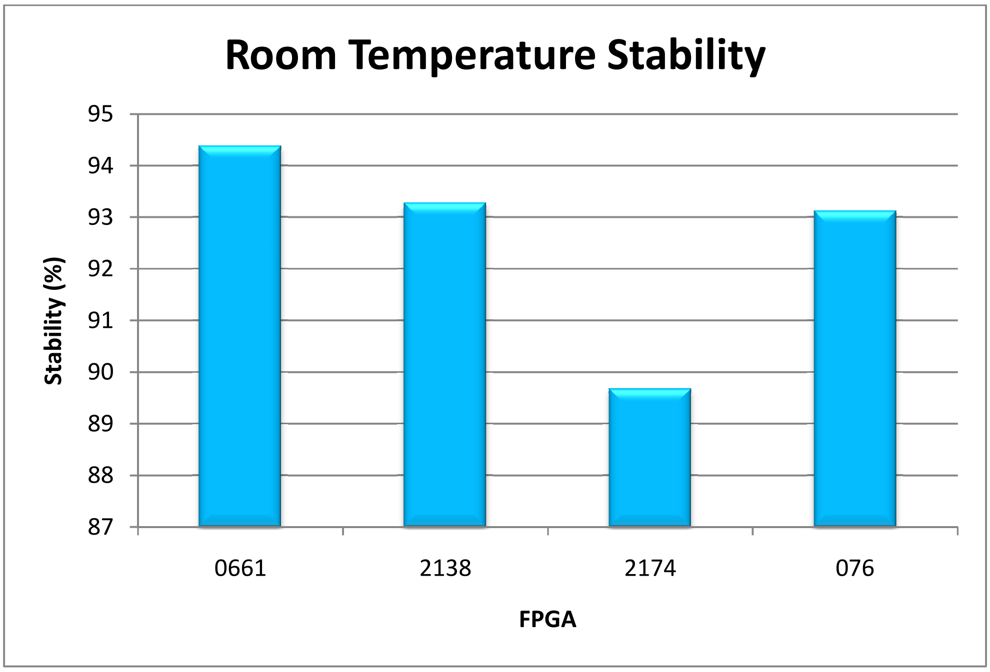


Figure 29. Average stability at room temperature, 20 ±10°C

4.3.3 Digital Key Distinguishability

As stated in Section 3.5, distinguishability is a measure of how different two digital keys are. Using Equation 3.2, with a bitwise comparison of all of the keys generated at a particular temperature setting, we can obtain a measurement of how distinguishable the keys are from one another. Sample results from this test can be found in Appendix C. The results shown in Figure A are representative of each of the readings taken at a limited temperature range. Keys generated on the boards used for this research were consistently greater than 15% distinguishable, with an average distinguishability of 65%.These results reflect those that were expected. While only a small amount of distinguishability is required for use in encryption a high level of distinguishability as shown in the results allows for digital keys to be used as identifiers for an FPGA since the likely hood of a collision is unlikely due to a large number of bits being different.

4.4 Summary of Results

The results in this section were divided between Circuit DNA and digital keys, describing both their stability on a single FPGA and their ability to be distinguished from one another when produced on separate devices.

Tests performed on several boards producing Circuit DNA showed that not only is Circuit DNA repeatable at a given temperature, it is also greater than 99% stable across a limited temperature range. Circuit DNA proves to very reliable over a large temperature range with an average stability of 97%. While being very stable, Circuit DNA also proves to have enough distinguishability to allow for the creation of unique digital keys.

Digital key generation results show a stability of greater than 99% at constant core temperature. Results across a limited temperature range of 20°±10°C, resulted in an average stability of about 92%, less than optimal for digital key generation. Digital key distinguishability proved to be very good at both constant and varying temperatures ranging anywhere from 15-60% distinguishable, results which are favorable for device identification.

V. Conclusions

This chapter provides some conclusions drawn from this research effort and their significance. These conclusions include those drawn pertaining to digital key stability and input buffer selection. This chapter also provides recommendations for future work which could improve the results gained through this research.

5.1 Conclusions about Digital Key stability

The main issue unearthed in this research is that without perfect stability, we cannot ensure that information encrypted using device specific generated keys will have the same results each and every time. While distinguishability is important to ensure against possible key collisions, all that is needed for a digital key to be unusable for encryption is a difference of a few bits. This is due to the avalanche effect present in cryptographic algorithms[**35**]; a difference of one bit in a large key will yield drastically different results. A difference of one bit in 128-bit digital key results in a very high stability, but since the key is not a 100% match, the avalanche effect would cause a cryptographic algorithm to yield a completely different result. I believe that the key to this problem lies in input buffer selection.

5.1.1 Input Buffer Selection

As previously discussed in Section 4.3.1, input buffer selection directly affects which values are used for the digital key based off Circuit DNA. Input buffer selection also indirectly controls stability, values chosen may not always reflect a ‘10’ or ‘0’ value within the Circuit DNA leading to a stability problem which can easily be fixed by selecting a different set of inputs. This research has shown that research effort needs to be placed in the area of input buffer selection, to be able to produce a product that can operate realistically outside of a lab setting.

5.2 Research Conclusions

The outlook of Circuit DNA being used for device specific digital keys looks promising. The improvements made to the TGP design have increased stability across a large temperature range allowing it to become more effective for field use. The connection between Circuit DNA and key generation needs to be improved in order to ensure stability for digital keys.

While digital key stability is still an issue with this research, it does not overshadow it as to prevent it from yielding positive results. Since stability is very high in a limited room temperature setting, device specific keys generated on a board can be used with a high level of trust given that core temperature is monitored and maintained at a nearly constant value. While this is not ideal for field use, it can still be used in controlled environments. Due to the possibility of keys being less than 100% stable in an extended temperature range, a simple protocol change could more than account for the stability issue. Generating multiple keys and encrypting data multiple times can account for the possibility of slightly different keys, ensuring that at least one of the datasets can be decrypted later.

5.3 Contributions

The contributions of this research include:

- ✓ Implemented 128-bit digital fingerprint system with user controllable buffer selection for individual tunable probes.
- ✓ Improved stability of Circuit DNA over a large temperature range to greater than 95%.
- ✓ Development of a simple buffer selection scheme using data gathered from multiple boards.
- ✓ Improved stability of previous tunable glitch probe design from a reported 96% to nearly 100% in a limited temperature range.

5.4 Future Work

- *Improved method of input buffer selection* values for use in digital key generation through the comparison of Circuit DNA between many FPGAs over a large temperature range.
- *Implementation on Virtex 6*, paired with 128-bit hardware AES. This would involve modifying existing software to operate with the Xilinx Micro-Blaze soft-core processor.
- *Digital Key implementation* as controlling values for polymorphic architecture. Using a circuit design which utilizes polymorphic gates, digital keys could be used to enable proper functionality.

Appendix A

A.1 Circuit DNA Comparison Perl Script

This Perl script is used to generate a comparison of multiple Circuit DNA data sets to generate a comparison report as detailed in Section 4.3.1.

```
#!/usr/local/bin/perl

###########################################
#                    Circuit DNA Data Comparison
#
#Author: Miles McGee
#Date: Aug 2011
#Description: This Perl Script takes in multiple circuit DNA data sets and
# compares them entry by entry reporting the number of matching loctaions
# to be used for statical purposes.
###########################################

##take in 3 Circuit DNA files as command line arguments
#these can be modified to include fewer or more input files
open(DNA0, $ARGV[0]) or die "Can't open '$ARGV[0]'\n";
open(DNA1, $ARGV[1]) or die "Can't open '$ARGV[1]'\n";
open(DNA2, $ARGV[2]) or die "Can't open '$ARGV[2]'\n";
open(DNA3, $ARGV[3]) or die "Can't open '$ARGV[2]'\n";

@CDNA0 = <DNA0>;
$length = @CDNA0;
@CDNA1 = <DNA1>;
@CDNA2 = <DNA2>;
@CDNA2 = <DNA3>;

##create output file
open(OUTPUT, ">Compare_out_2.txt");
open(OUTPUT2,">Comparelist2.txt");
##read each line in the file
$k = 0;
$total = 0;
while($k < $length){
        #find the probe label
        if($CDNA0[$k] =~ m/^Probe:/){
            print OUTPUT $CDNA0[$k];  #print probe label
            print "\n$CDNA0[$k]";
            print OUTPUT2 "\n$CDNA0[$k]";
            print OUTPUT " ,0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,";  #print label line
            $k++;  # skip the number label line
```

```
#do this for each line
for($lines = 0; $lines < 16; $lines++){
    $k++;
    #convert line to array for comparison
    print OUTPUT "\n";
    #print $CDNA0[$k];
    @line0 = split(',', $CDNA0[$k]);
    @line1 = split(',', $CDNA1[$k]);
    @line2 = split(',', $CDNA2[$k]);
    @line3 = split(',', $CDNA3[$k]);

   #print line number
   if($lines < 10){
       print OUTPUT "$lines,";
   }
   elsif($lines == 10){
       print OUTPUT "A,";
   }
   elsif($lines == 11){
       print OUTPUT "B,";
   }
   elsif($lines == 12){
       print OUTPUT "C,";
   }
   elsif($lines == 13){
       print OUTPUT "D,";
   }
   elsif($lines == 14){
       print OUTPUT "E,";
   }
   elsif($lines == 15){
       print OUTPUT "F,";
   }

     #do this for each column (each item in each line)
     for($column = 1; $column < 17; $column++){
         #compare each item from each of the three files
         if($line0[$column] != $line1[$column] && $line0[$column] != $line2[$column]){ ##
         this lien can be modified to account for more input fiese
         ##|| ($line0[$column] == $line1[$column] && $line0[$column] != $line2[$column] ))(## && $line0[$column] != $line3[$column]){
             #if they are not equal print an 'X'
             $item = $column-1; #this fixes an indexing problem
             print OUTPUT "X,";
             $total++;
             print OUTPUT2 "$lines,$item\n"
         }
         else{
             print OUTPUT "_,";
         }
     }
  }
  print OUTPUT2 "X";
  print OUTPUT "\n\n\n\n\n";
}
```

```
        }
        else{
            #skip the blank lines
            $k++;
        }
    }
    print "\nTotal: $total";
    #close files
    close(DNA0);
close(DNA1);
close(DNA2);
close(DNA3);
close OUTPUT;
```

A.2 Input Buffer Selection Perl Script

```
#!/usr/local/bin/perl

##########################################
#                   Input Buffer Selection
#
#Author: Miles McGee
#Date: Aug 2011
#Description: This Perl Script takes in a report generated by the DNA
# Comparison script and produces 2 strings of 128 hex characters to be
# used as input buffer values for digital key generation.
##########################################

open(LIST, $ARGV[0]) or die "Can't open '$ARGV[0]'\n";

@LIST1 = <LIST>;
$length = @LIST1;
@upper = "";
@lower = "";

my $range = 16;#this is because strawberry perl was compiled with the wrong number of randbits
##create output file
open(OUTPUT, ">keys.txt");
for($k = 0; $k < $length; $k++){
    if($LIST1[$k] =~ m/^Probe:/){
        $k++; #skip to next line
        if($LIST1[$k] =~ m/^X/){
            #if there arent any values generate 2 random values
            $random_number = int(rand($range));
            push(@upper, $random_number);
            $random_number2 = int(rand($range));
            push(@lower, $random_number2);

        }
        else{
            @array = split(',' ,$LIST1[$k]);
            chomp $array[1];
            push(@upper, $array[0]);
            push(@lower, $array[1]);
        }
    }
}
```

```
print OUTPUT "Upper:\t";
for($i = 0; $i < 129; $i++){
    if( $upper[$i] == 10){
            print OUTPUT "a";
    }
    elsif( $upper[$i] == 11){
        print OUTPUT "b";
    }
    elsif( $upper[$i] == 12){
        print OUTPUT "c";
    }
    elsif( $upper[$i] == 13){
        print OUTPUT "d";
    }
    elsif( $upper[$i] == 14){
        print OUTPUT "e";
    }
    elsif( $upper[$i] == 15){
        print OUTPUT "f";
    }
    else{
        print OUTPUT $upper[$i];
    }
}
print OUTPUT "\nLower:\t";
for($i = 0; $i < 129; $i++){
    if( $lower[$i] == 10){
            print OUTPUT "a";
    }
    elsif( $lower[$i] == 11){
        print OUTPUT "b";
    }
    elsif( $lower[$i] == 12){
        print OUTPUT "c";
    }
    elsif( $lower[$i] == 13){
        print OUTPUT "d";
    }
    elsif( $lower[$i] == 14){
        print OUTPUT "e";
    }
    elsif( $lower[$i] == 15){
        print OUTPUT "f";
    }
    else{
        print OUTPUT $lower[$i];
    }
}
close LIST;
close OUTPUT;
```

A.3 Bitwise Digital Key Comparison Perl Script

```perl
#!/usr/local/bin/perl

###############################################
#                    Bitwise Digital Key Comparison
#
#Author: Miles McGee
#Date: Aug 2011
#Description: This Perl Script takes in a list of digital keys an reports back
# stability and distinguishability statistics on them.
###############################################


##input format
#perl bitwiseCompare inputfile.txt outputfile.txt

#take in list of keys for comparison
open(KEYS, $ARGV[0]) or die "Can't open '$ARGV[0]'\n";
chomp($ARGV[1]);

#create array that contains the lines in the file
@keys = <KEYS>;

##create output file
open(OUTPUT, ">$ARGV[1]");

$numKeys = @keys;
print "Number of Keys: $numKeys\n";
print OUTPUT "Number of Keys: $numKeys\n";

for($i = 0; $i < $numKeys; $i++){
    #store current line in array
    @current = split(//, $keys[$i]);
    ##compare each line with every other line
    for($j = 0; $j < $numKeys; $j++){
        ##be aware that this approach compares the current line with itself, so each line should have a comparison of 100%
        @nextLine = split(//, $keys[$j]);
        #check each bit
        $sum = 0;
        for($k = 0; $k < 128; $k++){
            if($current[$k] == $nextLine[$k]){
                $sum++;
            }
        }
        $percentSame =($sum / 128);
        $percentDist = 1-($sum / 128);
        push(@averageArray, $percentSame);
        print "$i|$j\t$percentSame\t$sum\t\t $percentDist    \n";
        print OUTPUT "$i|$j\t$percentSame\t$sum\t\t $percentDist   \n";
    }
}


$sum = 0;
for($m = 0; $m < $numKeys; $m++){
    $sum = $sum + $averageArray[$m];
}
$average = $sum / $numKeys;
print "\nAverage Stability: $average";
print  OUTPUT "\nAverage Stability: $average";
close OUTPUT;
close KEYS;
```

Appendix B

B.1 Analysis of Circuit DNA Entry Changes Across a Large Temperature Range

The following tables detail the change in location of the transition point across a temperature range. The entries in the table represent the average percentage of lines within a probe that have changed since the last temperature reading (Ambient temperature readings taken at 10°C were compared with readings taken at 0°C). Values close to zero provide the best stability at a given temperature.

The readings have been split into four tables which represent the probe chains as shown in Figure 18. An average is shown at the bottom of the last tables which represents the average percentage of transition locations at that temperature. The purpose of this table is to aid in temperature range selection for best possible stability. This table could also be used to determine which probes provide the most stable transition points at a particular temperature, leading to the ability to choose specific probes for a digital key, requiring more probes in the system to retain a key size of 128-bits.

Table 2. Analysis of Circuit DNA entry changes across a Large Temperature Range

Approx Core Temp		20	30	40	50	65	80	95	110
Ambient Temp		10	20	30	40	50	60	70	80
Probe	127	3.1%	4.7%	4.7%	9.4%	4.7%	7.8%	7.8%	4.7%
	126	1.6%	0.0%	9.4%	4.7%	9.4%	7.8%	4.7%	12.5%
	125	4.7%	1.6%	12.5%	4.7%	10.9%	7.8%	1.6%	9.4%
	124	7.8%	6.3%	10.9%	7.8%	4.7%	1.6%	4.7%	6.3%
	123	0.0%	0.0%	1.6%	6.3%	7.8%	7.8%	4.7%	6.3%
	122	3.1%	0.0%	3.1%	7.8%	9.4%	4.7%	4.7%	4.7%
	121	0.0%	1.6%	3.1%	3.1%	6.3%	6.3%	3.1%	6.3%
	120	0.0%	6.3%	4.7%	7.8%	1.6%	3.1%	7.8%	3.1%
	119	4.7%	1.6%	1.6%	3.1%	3.1%	4.7%	4.7%	6.3%
	118	1.6%	1.6%	0.0%	1.6%	1.6%	3.1%	1.6%	3.1%
	117	0.0%	3.1%	6.3%	1.6%	0.0%	6.3%	6.3%	7.8%
	116	1.6%	3.1%	3.1%	3.1%	3.1%	3.1%	4.7%	6.3%
	115	1.6%	1.6%	1.6%	3.1%	3.1%	3.1%	4.7%	9.4%
	114	0.0%	4.7%	4.7%	1.6%	4.7%	0.0%	6.3%	7.8%
	113	1.6%	1.6%	3.1%	3.1%	9.4%	1.6%	9.4%	9.4%
	112	0.0%	4.7%	1.6%	1.6%	6.3%	4.7%	0.0%	7.8%
	111	3.1%	7.8%	0.0%	1.6%	3.1%	0.0%	1.6%	1.6%
	110	1.6%	3.1%	3.1%	3.1%	6.3%	7.8%	1.6%	0.0%
	109	3.1%	3.1%	3.1%	6.3%	1.6%	1.6%	1.6%	7.8%
	108	3.1%	3.1%	1.6%	3.1%	4.7%	1.6%	0.0%	6.3%
	107	3.1%	3.1%	1.6%	0.0%	0.0%	0.0%	6.3%	7.8%
	106	0.0%	1.6%	1.6%	6.3%	6.3%	4.7%	4.7%	4.7%
	105	0.0%	1.6%	1.6%	1.6%	3.1%	0.0%	0.0%	3.1%
	104	0.0%	3.1%	3.1%	0.0%	4.7%	1.6%	3.1%	7.8%
	103	1.6%	4.7%	6.3%	1.6%	4.7%	3.1%	6.3%	4.7%
	102	0.0%	0.0%	0.0%	4.7%	1.6%	1.6%	6.3%	3.1%
	101	3.1%	1.6%	1.6%	1.6%	0.0%	1.6%	4.7%	0.0%
	100	1.6%	3.1%	4.7%	0.0%	1.6%	6.3%	1.6%	3.1%
	99	1.6%	0.0%	1.6%	6.3%	3.1%	0.0%	0.0%	3.1%
	98	1.6%	4.7%	0.0%	0.0%	3.1%	4.7%	1.6%	1.6%
	97	3.1%	1.6%	1.6%	3.1%	1.6%	0.0%	3.1%	1.6%
	96	0.0%	1.6%	0.0%	0.0%	1.6%	0.0%	0.0%	0.0%

Approx Core Temp	20	30	40	50	65	80	95	110
Ambient Temp	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**
95	1.6%	1.6%	6.3%	6.3%	6.3%	7.8%	3.1%	3.1%
94	1.6%	4.7%	7.8%	4.7%	4.7%	3.1%	4.7%	12.5%
93	3.1%	4.7%	6.3%	7.8%	7.8%	7.8%	6.3%	10.9%
92	4.7%	4.7%	9.4%	9.4%	7.8%	6.3%	4.7%	9.4%
91	4.7%	7.8%	3.1%	6.3%	10.9%	4.7%	3.1%	9.4%
90	1.6%	9.4%	6.3%	4.7%	6.3%	3.1%	6.3%	7.8%
89	4.7%	3.1%	7.8%	9.4%	4.7%	3.1%	7.8%	9.4%
88	4.7%	3.1%	4.7%	6.3%	4.7%	4.7%	6.3%	4.7%
87	6.3%	1.6%	6.3%	6.3%	9.4%	7.8%	7.8%	7.8%
86	1.6%	9.4%	4.7%	7.8%	7.8%	4.7%	3.1%	10.9%
85	3.1%	7.8%	10.9%	6.3%	7.8%	3.1%	6.3%	14.1%
84	3.1%	6.3%	7.8%	4.7%	4.7%	4.7%	6.3%	4.7%
83	0.0%	1.6%	4.7%	4.7%	4.7%	9.4%	4.7%	10.9%
82	1.6%	1.6%	3.1%	1.6%	3.1%	3.1%	3.1%	10.9%
81	3.1%	7.8%	4.7%	10.9%	4.7%	9.4%	3.1%	3.1%
80	6.3%	3.1%	3.1%	6.3%	6.3%	4.7%	1.6%	7.8%
79	6.3%	6.3%	3.1%	1.6%	3.1%	6.3%	1.6%	7.8%
78	1.6%	4.7%	1.6%	9.4%	4.7%	6.3%	3.1%	3.1%
77	0.0%	0.0%	6.3%	1.6%	0.0%	3.1%	4.7%	3.1%
76	3.1%	3.1%	0.0%	6.3%	4.7%	0.0%	1.6%	3.1%
75	0.0%	1.6%	3.1%	3.1%	1.6%	1.6%	0.0%	4.7%
74	1.6%	0.0%	1.6%	3.1%	3.1%	1.6%	0.0%	3.1%
73	0.0%	0.0%	1.6%	1.6%	1.6%	0.0%	3.1%	0.0%
72	3.1%	0.0%	3.1%	1.6%	0.0%	9.4%	3.1%	1.6%
71	1.6%	0.0%	1.6%	3.1%	0.0%	0.0%	0.0%	1.6%
70	0.0%	1.6%	0.0%	0.0%	1.6%	1.6%	0.0%	0.0%
69	0.0%	0.0%	1.6%	0.0%	0.0%	0.0%	0.0%	1.6%
68	0.0%	0.0%	0.0%	0.0%	3.1%	0.0%	0.0%	0.0%
67	0.0%	0.0%	0.0%	0.0%	0.0%	1.6%	3.1%	3.1%
66	1.6%	0.0%	0.0%	1.6%	0.0%	1.6%	0.0%	0.0%
65	0.0%	0.0%	1.6%	4.7%	1.6%	3.1%	0.0%	1.6%
64	0.0%	1.6%	1.6%	1.6%	1.6%	0.0%	0.0%	0.0%

Approx Core Temp	20	30	40	50	65	80	95	110
Ambient Temp	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**
63	0.0%	1.6%	1.6%	1.6%	0.0%	1.6%	4.7%	3.1%
62	0.0%	3.1%	3.1%	0.0%	1.6%	0.0%	1.6%	3.1%
61	0.0%	1.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
60	0.0%	1.6%	3.1%	1.6%	0.0%	1.6%	0.0%	4.7%
59	0.0%	3.1%	1.6%	3.1%	1.6%	0.0%	0.0%	4.7%
58	3.1%	1.6%	0.0%	0.0%	0.0%	0.0%	1.6%	6.3%
57	4.7%	0.0%	3.1%	1.6%	1.6%	1.6%	0.0%	0.0%
56	0.0%	0.0%	3.1%	1.6%	4.7%	1.6%	0.0%	4.7%
55	3.1%	0.0%	1.6%	4.7%	3.1%	3.1%	1.6%	6.3%
54	1.6%	0.0%	0.0%	1.6%	6.3%	4.7%	1.6%	3.1%
53	0.0%	1.6%	3.1%	3.1%	3.1%	7.8%	1.6%	6.3%
52	0.0%	4.7%	6.3%	1.6%	6.3%	4.7%	1.6%	6.3%
51	3.1%	3.1%	1.6%	6.3%	4.7%	1.6%	4.7%	7.8%
50	1.6%	1.6%	4.7%	3.1%	6.3%	3.1%	3.1%	3.1%
49	0.0%	1.6%	1.6%	1.6%	3.1%	1.6%	3.1%	1.6%
48	3.1%	6.3%	1.6%	0.0%	1.6%	3.1%	3.1%	0.0%
47	1.6%	1.6%	0.0%	3.1%	3.1%	4.7%	3.1%	4.7%
46	0.0%	0.0%	3.1%	1.6%	1.6%	0.0%	0.0%	7.8%
45	1.6%	0.0%	1.6%	0.0%	0.0%	6.3%	1.6%	4.7%
44	0.0%	0.0%	0.0%	1.6%	1.6%	7.8%	1.6%	4.7%
43	1.6%	1.6%	0.0%	1.6%	0.0%	4.7%	0.0%	9.4%
42	3.1%	0.0%	0.0%	1.6%	1.6%	3.1%	0.0%	4.7%
41	0.0%	4.7%	1.6%	1.6%	0.0%	0.0%	0.0%	3.1%
40	1.6%	0.0%	0.0%	0.0%	3.1%	0.0%	1.6%	7.8%
39	0.0%	1.6%	1.6%	0.0%	3.1%	1.6%	1.6%	1.6%
38	3.1%	0.0%	0.0%	3.1%	1.6%	1.6%	0.0%	1.6%
37	0.0%	0.0%	0.0%	1.6%	0.0%	0.0%	0.0%	1.6%
36	0.0%	1.6%	1.6%	1.6%	1.6%	1.6%	3.1%	4.7%
35	0.0%	0.0%	0.0%	4.7%	3.1%	0.0%	3.1%	0.0%
34	0.0%	0.0%	3.1%	0.0%	1.6%	3.1%	0.0%	0.0%
33	0.0%	1.6%	0.0%	1.6%	0.0%	0.0%	0.0%	4.7%
32	1.6%	0.0%	0.0%	3.1%	1.6%	3.1%	1.6%	1.6%

Approx Core Temp	20	30	40	50	65	80	95	110
Ambient Temp	10	20	30	40	50	60	70	80
31	3.1%	7.8%	4.7%	7.8%	7.8%	10.9%	7.8%	14.1%
30	3.1%	4.7%	9.4%	12.5%	6.3%	14.1%	9.4%	15.6%
29	3.1%	10.9%	10.9%	12.5%	9.4%	14.1%	3.1%	14.1%
28	3.1%	1.6%	9.4%	17.2%	12.5%	10.9%	3.1%	12.5%
27	3.1%	9.4%	7.8%	7.8%	4.7%	7.8%	6.3%	14.1%
26	0.0%	14.1%	12.5%	7.8%	4.7%	9.4%	15.6%	10.9%
25	1.6%	7.8%	7.8%	7.8%	0.0%	6.3%	4.7%	4.7%
24	1.6%	17.2%	3.1%	9.4%	6.3%	6.3%	12.5%	6.3%
23	3.1%	3.1%	6.3%	7.8%	7.8%	9.4%	4.7%	10.9%
22	3.1%	6.3%	3.1%	6.3%	10.9%	6.3%	6.3%	10.9%
21	3.1%	4.7%	10.9%	12.5%	10.9%	10.9%	9.4%	9.4%
20	3.1%	7.8%	6.3%	12.5%	3.1%	10.9%	4.7%	3.1%
19	6.3%	6.3%	4.7%	10.9%	12.5%	7.8%	6.3%	9.4%
18	6.3%	6.3%	4.7%	6.3%	6.3%	7.8%	6.3%	4.7%
17	0.0%	3.1%	6.3%	6.3%	10.9%	7.8%	4.7%	10.9%
16	0.0%	3.1%	9.4%	4.7%	12.5%	6.3%	3.1%	14.1%
15	6.3%	1.6%	4.7%	7.8%	3.1%	4.7%	3.1%	7.8%
14	0.0%	6.3%	7.8%	3.1%	12.5%	3.1%	3.1%	10.9%
13	1.6%	6.3%	6.3%	4.7%	7.8%	4.7%	4.7%	3.1%
12	1.6%	4.7%	9.4%	4.7%	6.3%	6.3%	6.3%	3.1%
11	3.1%	1.6%	3.1%	6.3%	12.5%	6.3%	0.0%	4.7%
10	1.6%	3.1%	7.8%	1.6%	6.3%	1.6%	0.0%	9.4%
9	0.0%	4.7%	3.1%	6.3%	7.8%	1.6%	3.1%	3.1%
8	4.7%	0.0%	7.8%	3.1%	6.3%	3.1%	3.1%	7.8%
7	3.1%	1.6%	3.1%	4.7%	4.7%	3.1%	4.7%	0.0%
6	1.6%	1.6%	0.0%	3.1%	1.6%	0.0%	0.0%	1.6%
5	0.0%	1.6%	0.0%	1.6%	3.1%	1.6%	0.0%	1.6%
4	0.0%	1.6%	0.0%	4.7%	7.8%	1.6%	1.6%	6.3%
3	1.6%	1.6%	1.6%	3.1%	3.1%	4.7%	0.0%	6.3%
2	3.1%	3.1%	1.6%	0.0%	6.3%	3.1%	6.3%	4.7%
1	0.0%	0.0%	0.0%	1.6%	1.6%	1.6%	1.6%	0.0%
0	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Average	**3.0%**	**4.4%**	**4.9%**	**5.2%**	**5.6%**	**5.4%**	**4.5%**	**6.8%**

Appendix C

C.1 Sample Digital Key Distinguishability Results

This table details the results of a distinguishability test performed at an ambient temperature of 20°C on four FPGAs. Each FPGA is represented by two keys, the key groupings for the FPGA's is 0-1, 2-3,4-5, and 6-7. Notice that keys from the same board have a very low distinguishability measurement, while keys from separate boards have much larger measurements.

Table 3. Sample Digital Key distinguishability results

Keys Being Compared	Stability Measurement	Number of Matching Bits	Distinguishability Measurement
0|0	100.00%	128	0.00%
0|1	99.22%	127	0.78%
0|2	4.69%	6	95.31%
0|3	4.69%	6	95.31%
0|4	85.16%	109	14.84%
0|5	84.38%	108	15.63%
0|6	71.09%	91	28.91%
0|7	70.31%	90	29.69%
1|0	99.22%	127	0.78%
1|1	100.00%	128	0.00%
1|2	5.47%	7	94.53%
1|3	5.47%	7	94.53%
1|4	84.38%	108	15.63%
1|5	83.59%	107	16.41%
1|6	71.88%	92	28.13%
1|7	71.09%	91	28.91%
2|0	4.69%	6	95.31%
2|1	5.47%	7	94.53%
2|2	100.00%	128	0.00%
2|3	100.00%	128	0.00%
2|4	19.53%	25	80.47%
2|5	18.75%	24	81.25%
2|6	33.59%	43	66.41%
2|7	34.38%	44	65.63%
3|0	4.69%	6	95.31%
3|1	5.47%	7	94.53%
3|2	100.00%	128	0.00%
3|3	100.00%	128	0.00%
3|4	19.53%	25	80.47%
3|5	18.75%	24	81.25%
3|6	33.59%	43	66.41%

3|7	34.38%	44	65.63%
4|0	85.16%	109	14.84%
4|1	84.38%	108	15.63%
4|2	19.53%	25	80.47%
4|3	19.53%	25	80.47%
4|4	100.00%	128	0.00%
4|5	99.22%	127	0.78%
4|6	78.13%	100	21.88%
4|7	77.34%	99	22.66%
5|0	84.38%	108	15.63%
5|1	83.59%	107	16.41%
5|2	18.75%	24	81.25%
5|3	18.75%	24	81.25%
5|4	99.22%	127	0.78%
5|5	100.00%	128	0.00%
5|6	77.34%	99	22.66%
5|7	76.56%	98	23.44%
6|0	71.09%	91	28.91%
6|1	71.88%	92	28.13%
6|2	33.59%	43	66.41%
6|3	33.59%	43	66.41%
6|4	78.13%	100	21.88%
6|5	77.34%	99	22.66%
6|6	100.00%	128	0.00%
6|7	97.66%	125	2.34%
7|0	70.31%	90	29.69%
7|1	71.09%	91	28.91%
7|2	34.38%	44	65.63%
7|3	34.38%	44	65.63%
7|4	77.34%	99	22.66%
7|5	76.56%	98	23.44%
7|6	97.66%	125	2.34%
7|7	100.00%	128	0.00%
Average Stability:	**58.16%**		

Bibliography

[1] Ivan Gonzalez and Esam El-Araby, "Classification of Application Development for FPGA-Based Systems," in *Aerospace and Electronics Conference (NAECON)*, Dayton, OH, 2008, pp. 203-208.

[2] Richard Pedersen, "FGPA-Based Military Avionics Computing Circuits," *IEEE Aviation and Electronics Magazine*, pp. 9-13, July 2004.

[3] Actel Corporation. (2002, August) Design Security with Sctel FPGAs. [Online]. http://www.actel.com/documents/DesignSecurityPPT.pdf

[4] Xilinx Corporation. (2002) Triple DES Encryption in Selected Virtex-II Devices. [Online]. http://www.xilinx.com/support/documentation/white_papers/wp155.pdf

[5] Amir Moradi, Markus Kasper, and Christof Paar. (2011, July) On the Portability of Side-Channel Attacks - An Analysis of the Xilinx Virtex 4 and Virtex 5 Bitstream Encryption Mechanism. [Online]. http://eprint.iacr.org/2011/391.pdf

[6] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. (2011, July) International Association for Cryptologic Research. [Online]. http://eprint.iacr.org/2011/390.pdf

[7] Hiren Patel, "A Top Down Approach to Creating a Digital Fingerprint to Uniquely Identify Field Programmable Gate Arrays," Air Force Institute of Technology, Wright-Patterson AFB, OH, Masters Thesis 2010.

[8] William Stanton, "Circuit DNA Extraction and Digital Key Generation," Air Force Institute of Technology, Wright-Patterson AFB, OH, Masters Thesis 2011.

[9] Jennifer Anilao, "Utilizing the Digital Fingerprint Methodology for Secure Key Generation," Air Force Institute of Technology, Wright-Patterson AFB, OH, Masters Thesis 2010.

[10] Y. Kim and J. T. McDonald, "Considering Software Protection for Embedded Systems," *CrossTalk: the Journal o fDefense Software Engineers*, pp. 4-8, Sept/Oct 2009.

[11] W. Luo, Z. Zhang, and X. Wang, "Designing polymorphic circuits with polymorphic gates: a general design approach ," *Circuits, Devices & Systems, IET*, vol. 1, no. 6, pp. 470-476, December 2007.

[12] Lukas Sekanina, Lukas Starecek, Zednek Kotasek, and Zbysek Gajda, "Polymorphic Gates in Design and Test of Digital Circuits," in *Unconventional Computing*, vol. 7, March 2008.

[13] Donald A. Neamen, *Microelectronic: Circuit Analysis and Design*, 3rd ed. New York, NY, United States of America: McGraw-Hill, 2007.

[14] Intel. (2009) Introduction to Intel's 32nm Process Technology. [Online]. http://download.intel.com/pressroom/kits/32nm/westmere/Intel_32nm_overview.pdf

[15] James W. Crouch, "Digital Fingerprinting of Field Programmable Gate Arrays," Air Force Institute of Technology, Wright-Patterson AFB, OH, Masters Thesis 2008.

[16] H. Mahmoodi and K. Roy, "Estimation of Delay Variations due to Random-Dopant Fluctuations in Nanoscale CMOS Circuits," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 9, pp. 1787-1796, Sept. 2005.

[17] D.S. Boning and S. Nassif, "Models of Process Variations in Device Interconnect," in *Design of High Performance Microprocessor Circuits*, A. Chandrakasan, W. Bowhill, and F. Fox, Eds.: IEEE Press, 2000, ch. 6.

[18] G. S. May and S. M. Sze, *Fundamentals of Modern VLSI Devices*. Hoboken, NJ: Wiley & Sons, 2004.

[19] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*. New York: Cambridge University Press, 1998.

[20] Hiren Patel and et al., "Increasing Stability and Distinguishability of the Digital Fingerprint in FPGAs Through Input Word Analysis," Air Force Research Laboratory, Wright-Patterson AFB, OH,.

[21] G.E. Suh and S Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *Design Automation Conference*, San Diego, CA, 2004, pp. 9-14.

[22] M. Majzoobi, F. Koushanfar`, and M. Potkonjak, "Lightweight Secure PUFs," in *IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 670-673.

[23] Heike Busch, Miroslava Sotakova, Stefan Katzenbeisser, and Radu sion. (2010) The PUF Promise. [Online]. http://dx.doi.org/10.1007/978-3-642-13869-0_21

[24] Blaise Gassend and et al., "Silicon Physical Random Functions," in *Computer and Communications Security*, New York, NY, 2002, pp. 148-160.

[25] Jae W. Lee et al., "A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications," in *Symposium on VLSI Circuits*, 2004.

[26] S.S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended Abstract: The Butterfly PUF Protecting IP on every FPGA," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on* , Anaheim, CA, pp. 67-70.

[27] A. A. Gaffar, J. A. Clarke, and G. A. Constantinides, "Modeling of glitch effects in FPGA based arithmetic circuits," in *IEEE International Conference on Field Programmable Technology*, 2006, pp. 349-352.

[28] Hiren J. Patel and et al., "Creating a Unique Digital Fingerprint Using Existing Combinational Logic," in *International Symposium on Circuits and Systems*, 2009.

[29] National Security Administration, "CNSS Policy No. 15, Fact Sheet No. 1: National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information," in *Committee on National Security Systems*, 2003.

[30] National Institute of Standards and Technology, Federal Informaiton Processing Standards Publication 187 Announcing the ADVANCED ENCRYPTION STANDARD (AES), 2001.

[31] Xilinx. (2009) Virtex-5 Family Overview. [Online]. http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf

[32] Xilinx. (2009) Data2MEM User Guide. [Online]. http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/data2mem.pdf

[33] Xilinx. (2011, July) PlanAhead Software Tutorial: Partial Reconfiguration of a Processor Peripheral. [Online]. http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/PlanAhead_Tutorial_Reconfigurable_Processor.pdf

[34] Kenneth H. Rosen, *Discrete Mathematics and Its Applications*, 6th ed. New York, NY, United States of America: McGraw-Hill, 2007.

[35] Aram Kahlili. (2001) University of Maryland Computer Science Lecture: Symmetric Cryptography. [Online]. http://www.cs.umd.edu/~waa/414-F01/symmetric.pdf

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From – To)*
15-09-2011	**Master's Thesis**	Aug 2010 – Sept 2011

4. TITLE AND SUBTITLE
Critical Information technology on FPGAs through Unique Device Specific Keys

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
McGee, Miles E., Mr.

5d. PROJECT NUMBER
ENG 10-326

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB, OH 45433-7301

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCE/ENG/11-10

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Dr. Robert L. Herklotz
Program Manager – Information Operations and Security
Air Force Office of Scientific Research (AFOSR/RSL)
875 N. Randolph Street, Suite 325, Room 3112
Arlington, VA 22203-1768
(703) 696-6565; Robert.herklotz@afosr.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Field Programmable Gate Arrays (FPGAs) are being used for military and other sensitive applications, the threat of an adversary attacking these devices is an ever present danger. While having the ability to be reconfigured is helpful for development, it also poses the risk of its hardware design being cloned. Static random access memory (SRAM) FPGA's are the most common type of FPGA used in industry. Every time an SRAM-FPGA is powered up, its configuration must be downloaded. If an adversary is able to obtain that configuration, they can clone sensitive designs to other FPGAs. A technique that can be used to protect FPGAs from these types of attacks is known as Digital Fingerprinting (DF). DF takes advantage of the manufacturing variability that naturally occurs in the integrated circuit fabrication process. If another factor can be introduced making the FPGA's operation dependent on more than the design specified within its configuration and response to external outputs, we can defend against cloning. This solution would allow for an FPGA's operation to be dependent on how the downloaded configuration interacts with the hardware itself. This research uses DF technology to create unique device specific keys for use as encryption keys or control values for polymorphic circuits to protect information on FPGAs.

15. SUBJECT TERMS
FPGA, Digital Fingerprint, Circuit ID, Authentication, Stability, Distinguishability, Circuit DNA

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Dr. Yong Kim (ENG)
REPORT	**ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)*
U	**U**	**U**	**UU**	88	(937) 255-3636 x4620 Email:yong.kim@afit.edu