June 2020

Implementation of SR Flip-Flop Based PUF on FPGA for Hardware Security

Sai Praneeth Sagi
University of South Florida

Implementation of SR Flip-Flop Based PUF on FPGA for Hardware Security

by

Sai Praneeth Sagi

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
Department of Electrical Engineering
College of Engineering
University of South Florida

Co-Major Professor: Dr. Srinivas Katkoori, Ph.D.
Co-Major Professor: Dr. Wilfrido Moreno, Ph.D.
Dr. Nasir Ghani, Ph.D.

Date of Approval:
June 19, 2020

Keywords: Weak PUF, Strong PUF, Slices, LUTs, Bitstream

DEDICATION

I dedicate this work to my Family for their Unconditional love and support.

ACKNOWLEDGEMENTS

I will begin with Dr. Srinivas Katkoori for believing in my potential and giving me the opportunity to work in the subject which I am passionate about. Working with Dr. Srinivas Katkoori in the subject which I wanted gave meaning to my Master's degree at University of South Florida. I thank him for bringing me success with his constant help and guidance, I am forever grateful to him. I would like to thank my parents for their love and support which I cannot express in words. I would like to thank Dr. Wilfrido Moreno and Dr. Nasir Ghani for offering their precious time to serve as members on my thesis committee. I would like to thank my uncle, Dr. Ranganadha Rao Vemuri, and my Professor from V. R. Siddhartha Engineering College, Dr. K. Rama Krishna, for motivating me to choose this branch of study. I would like to thank God for giving me hope that I can finish this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Physical Unclonable Functions (PUF) are used for authentication and key generation to obtain a unique signature and are widely used in hardware security applications. In this work, we propose Set-Reset Flip-flop (SRFF) based PUF for FPGAs. We exploit the race around condition of the SRFF to obtain a one-bit signature output which is a function of feedback path delays. In deep sub-micron technology node, delay variations on an FPGA device are significant due to manufacturing process variations. Thus, an SRFF output value is a function of its location on the FPGA device. We implement registers of various bit widths and extract signature in different locations on a device. We demonstrate that the signatures are spatially unique for sufficiently large register bit widths. We have experimented with fifteen (15) Spartan-6 FPGA devices (45 nm technology node) to study the uniqueness, uniformity, randomness, and robustness of the PUF as the Spartan-6 FPGA devices are very well known for the low power applications and good performance. We explored the Xilinx 14.7 ISE tool and used some of its in-built core tools like Chip-scope to synthesize higher bitstreams.

CHAPTER 1: INTRODUCTION

In today’s world, hardware plays a major role in every aspect of our daily life. Hardware security is very important and is possible through various methods. In this thesis, we propose a method to produce a unique identification key or a unique signature from a hardware device such as FPGA (Field Programmable Gate Array). Today, IC’s are extremely vulnerable for malicious modifications due to globalization. These vulnerabilities raise serious concerns such as leakage of confidential information in various use case scenarios such as military systems and house-hold appliances. Due to these reasons, the underlying hardware used for information processing in a computer system cannot being trusted. An adversary can introduce a Trojan designed to destroy the system at some future time or to leak secret keys covertly to the adversary. There is a need to uniquely identify a device in the field.

Physical Unclonable Functions (PUF) are used for authentication and key generation to obtain a unique signature and are widely used in hardware security applications. In this work, we propose Set-Reset Flip-flop (SRFF) based PUF for FPGAs. We exploit the race around condition of the SRFF to obtain a one-bit signature output which is a function of feedback path delays. In deep sub-micron technology node, delay variations on an FPGA device are significant due to manufacturing process variations. Thus, an SRFF output value is a function of its location on the FPGA device.

We validated the proposed idea on Xilinx Spartan6 FPGAs using the Xilinx ISE 14.7 software. The standard SR flip flop is modified with a multiplexor to incorporate two modes:

regular mode and PUF mode. In the regular mode, the SRFF PUF behaves like a normal SR flip-flop. In the PUF mode, the SRFF is first put into a race around condition and then into the latch mode.

The SR flip flop based PUF design which we proposed for this thesis work is based on previous works which actually deals with the race-around condition and the PUF signature is based on the gate delays. In this work, this PUF design is implemented on FPGA boards by mapping the SRFF to different locations using the FPGA programmable logic. We used Digilent Spartan 6 ATLYS FPGA board which actually have 16 locations on the FPGA. Each location has around 3300 slices with in them which will make a total of 54,576 slices. Each slice has 4 LUTs to which we mapped the NAND gates. In the SR flip flop when S=R=1, the race-around or the toggle condition comes into play. The flip flop will be in a condition where the bits '1' and '0' will run very fast. Using the slices, we increased the delays between the cross-coupled NAND gates because of which the output is produced as 1 or 0. This value is unique and reproducible. The major advantage is the power consumption is very low thus making the proposed PUF to play a major role in low power applications.

The rest of the thesis is organized as follows. In Chapter 2, we present the background and the literature survey. We review various PUF designs in the literature and distinguish between strong PUFs and weak PUFs. In Chapter 3, we described the proposed SRFF based PUF idea. In Chapter 4, we report the experimental results. Finally, in Chapter 5, we draw conclusions. In Appendix A, we present all the data collected from 15 FPGA boards.

CHAPTER 2: BACKGROUND AND RELATED WORK

In this chapter, we explain the concept of PUF and its applications and survey different types of PUF in the literature. We specifically focus on PUF designs implemented on FPGAs.

2.1 Introduction to PUF

The Physical Unclonable Function is a hardware implementation that generates a unique key which can be used for protecting the hardware IP. IP protection is a major problem in manufacturing the chips in today’s world. The major applications of a PUF are low cost authentication and key generation. Before we talk about these applications, let us understand how a PUF works. Figure 2.1 is a black box view of a PUF. The input given to the PUF is known as a *Challenge* and the corresponding output is known as *Response*. In general, a PUF produces several Challenge-Response Pairs (CRPs). Note that PUF is designed in such a way that the adversary cannot predict the response as the internal design is not exposed or cannot be replicated (“unclonable”).

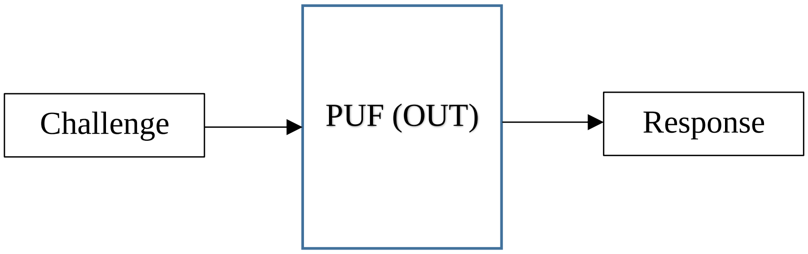


Figure 2.1: Block Diagram of a PUF

The major applications of the usage of PUF will be cost efficient and they are used for security purposes such as authentication and securing the key. PUFs can be classified as *strong* or *weak*. In these two categories, the strong ones are used for authentication purposes and the weak PUFs (or physically obfuscated keys) are used for storage of the key. We will later explain in more detail the difference between the strong and weak PUF and their functionalities. The major difference between the strong and weak is based on the number of the challenge response pairs they support. If it is a strong PUF, then they support a greater number of CRPs and if it is a weak PUF the CRP set is very small. Error correction can be applied to a PUF as there is a chance that the PUF can be affected due to noise. Examples for Strong PUFs (Optical PUF, Arbiter PUF) and weak PUFs are explained. The arbiter PUF produces the output based on the path delays when the inputs are given. Whichever path has more delay will be a slower path and whichever path has the less delay will have the faster path. This shows that this PUF (Arbiter PUF) is based on path delays and will work accordingly although the layout of both paths is the same. There is a similarity between the SRFF based PUF and Arbiter PUF: both use unbalanced paths in generating a signature bit. The major advantage is that the adversary cannot calculate the outputs properly as he/she does not know the internal delays (as each delay is independent) of the design between the gates which is helpful for the security. The major disadvantage in optical and arbiter PUFs is that they get affected by the environmental conditions (noise, temperature, supply voltage), that is they lose their stability and the error correction methods are needed to stabilize the design.

Herder *et al.* [1] explained the low-cost authentication using Strong PUFs and key generation protocols using weak PUFs in a detailed manner along with examples for the weak PUFs too. The two examples used for the weak PUF were the Ring Oscillator PUF and the SRAM PUF. Coming to the first example, the same concept of gate delays was used for the Ring Oscillator PUF too. There are some calculations based on how many bits can be extracted from the PUF. They faced the same problems which we get for the strong PUFs that are the environmental variations. But again, error correction can play a major role in this aspect. Coming to the other example of weak PUF, it depends on the Vth (threshold voltage). By varying the Vth, the transistors in the SRAM cell will result in producing propagation delays due to which we get the output as either a 0 or 1. This example also explained about metastable state and environmental variations (error correction will be useful). New PUF technologies (PPUF model and hardware) were also discussed briefly.

2.2 Weak PUF Designs

It will not change into a strong PUF if we use several instances to support a greater number of CRPs. Because the responses are directly related to component count which depends on the manufacturing variations. Environmental conditions will not affect the nature of the PUF. Explanation about SRAM and its power on state is an example for weak PUF. Key secrecy should be maintained as the PUF will be at risk otherwise.

2.2.1 Ring Oscillator PUF Design

The PUF has a great quality of unpredictability and when we use a device like FPGA it is best move for the hardware security as it gives efficient results which are unique and unpredictable. Patterson *et al.* [10] first explained the properties of PUF and how the combination of PUF design with the FPGA gives us best response pairs as the response will be random as it is generated using the on-chip PUF. This is one of the first and basic PUF designs. The Ring-Oscillator PUF is considered as a weak PUF as it is used only for key authentication. In Figure 2.2, we show a RO PUF with 256 ring oscillators which sends the inputs to the multiplexer and are used to send the input bits to the counter according to the challenge preferred. The counters play a major role in generating the output response. There will be comparison between the two counters to decide which response comes out as the output depending on which counter has the larger value. This process is repeated several times to produce the required bitstreams.

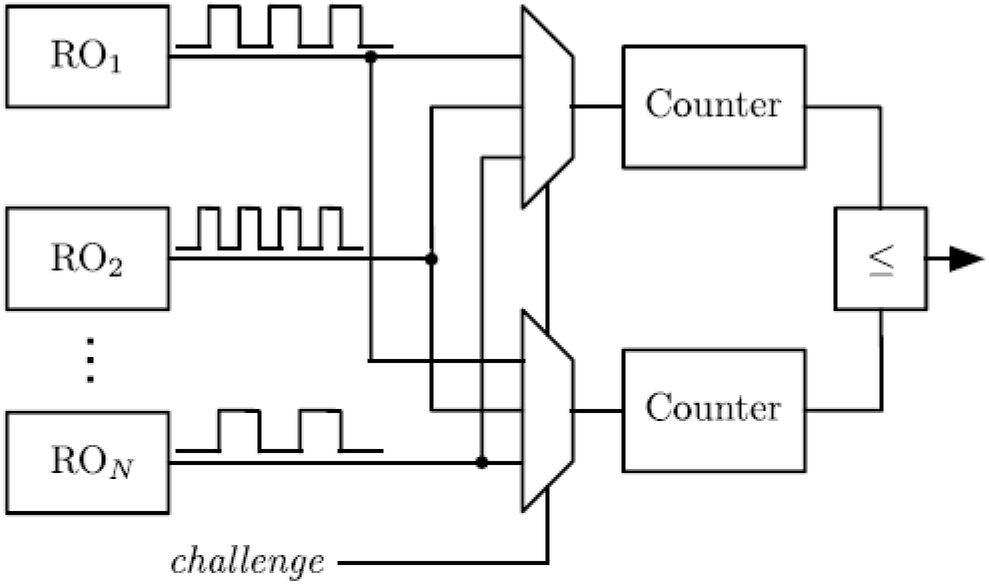


Figure 2.2: Design of a RO-PUF (Reproduced from [10])

For example, if we need a 64-bit output we need to repeat the above process 64 times. This ring oscillator design is also synthesized on FPGA using the concept of manual routing as the delays in the design will not vary due to automatic routing. The delays depend on the process variations. Generally, when we opt automatic routing the design will be optimized, and the hierarchy will not be maintained, and due to this reason, the delays will not vary. The tests were carried for 8-bit ring oscillator on Spartan-6 LX45 family FPGA. Both inter chip and intra chip variations are tested. The hamming distance is also measured and it is clear that the 8-bit responses which are generated as the output from the design shows up to 95 percent of uniqueness in the result and for every 1 bit change in the input there will be a change of 2.90 which is approximately 3 bits change in the output response. On average, when we compare two different PUF designs to the same challenge there will be a variation of 46.16% between those designs. Coming to the robustness of the PUF design, this PUF design is stable when there are temperature variations. When the temperature is varied from 10C to 65C, the output responses remains stable.

There are some drawbacks for this PUF as the frequency of the PUF is stable there might be some problems in varying the delays within the design which might produce a similar output. Another example for weak PUF is SRAM based memory PUF which is also used for key generating methods.

2.3 Strong PUF Designs

Due to a greater number of CRPs, these are used for authentication. Weak PUF can also authenticate using external hardware devices but strong PUF will not require any external hardware as they will have enough CRPs to authenticate.

2.3.1 Arbiter PUF Design

This is the first PUF design and is the best example for a Strong PUF model which is explained by Herder *et al.* [1]. We can see the Arbiter PUF design below in the Figure 2.3. The Arbiter PUF can be designed with the use of a latch, a flip-flop or a XOR gate. In this case, a latch is used for the design of Arbiter PUF. The input is sent into the device has 2 different paths which are built with the multiplexers. This connection is also similar to cross-coupled structure where the input 1 is sent as the first input for the multiplexer 1 and the same input is sent as the second input for the second multiplexer. So, it is similar with the second input as well as it works vice-versa. The switch delay elements count plays a major role in creation of input path. The multiplexers in the device are used to decide which path should send the input first and the multiplexer will alternate the paths depending on the delays to get an output response. The design shown below is 128-bit design which has 256 multiplexers finally ending into a D-Latch. The 'X' is acting as a selection line and 'Y' is the final output response. The output response will be generated based on which path is faster due to the path delays in the design. This gives a major advantage for this PUF design as the adversary cannot predict the delays in the paths between as each gate has its own delay depending on the manufacturing variations in the gates. It is highly secure as the adversary will not be able to create the duplicate design of the PUF either as the delays vary from gate to gate and this helps in maintaining the security of the hardware device.

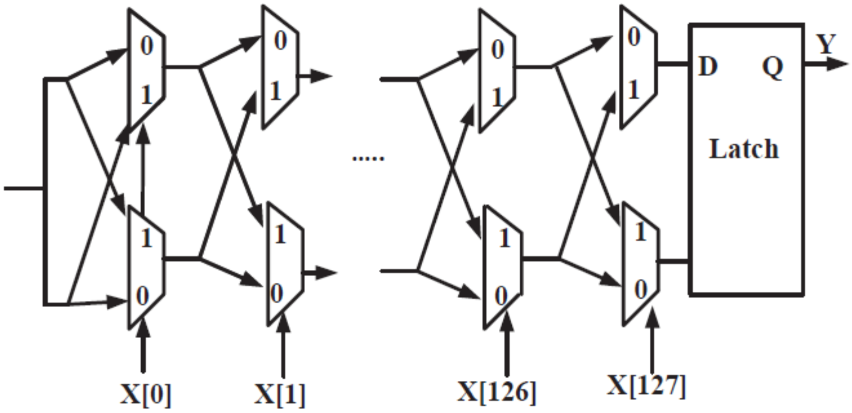


Figure 2.3: Design of an Arbiter PUF (Reproduced from [1])

But there are some of the drawbacks in this particular PUF design when it is placed on-chip that is on to an FPGA device. Generally, in the FPGA the routing of the paths is done automatically by using different slices in the different locations present on the FPGA. But because of this reason there is major drawback as the randomness which is a one of the 4 properties of the PUF will not be satisfied as the auto routing of the paths will overshadow the results which will be produced. The variation in the path delay cannot be seen as the path which is set using automated routing will not change randomly. So, this PUF design is not adaptable to the FPGA devices as there is lack of process variability for this design. This PUF also consumes more power and area which are major drawbacks.

2.3.2 Optical PUF

This PUF is the root cause for the design of the Arbiter PUF. This PUF design comes under the category of strong PUF. This design consists of a device which supports the scattering of light in certain medium. A laser is used for optical scattering and when the light is sent into the device it forms a complex pattern which cannot be recreated or duplicated which proves the name physically unclonable. The laser is sent into the device with certain angle and for each iteration there will be a change of angle of light. The complex pattern which is produced from the device as output has multiple scattering within the optical device which results in multi-bit responses.

This PUF is one of the most secure PUF designs as the output is unpredictable as it is based on the input scattering. Each input laser beam sent in might result in more than 1000 challenge response pairs which makes the output response more secure. This device is made of silica spheres in very large numbers which are used to refract the light in the device to produce a complex pattern. So, the adversary will have a hard time if he tries to create the same device as it is highly impossible to design and manufacture a device with the same number of spheres and scatter the light in the same angle to find the output responses which show that this PUF design is very secure. The process for generating output responses is complex and manufacturing this kind of devices will be very expensive.

2.4 Some more PUF Designs

2.4.1 Light weight Secure PUF

This PUF is also a primary example for a weak PUF and it is designed to introduce and combine the multiple delays which are used for the generation of bit response and works as a security (to have resistance) for the PUFs against reverse engineering and some more attacks. The goal is to make the PUFs robust and secure. As mentioned above they followed three basic principles to achieve their goal:

- The multiple paths with certain random delays
- CRPs are chosen wisely
- The outputs from the delay paths is considered and it is changed into a jumbled transformation because of this, the PUF security increases.

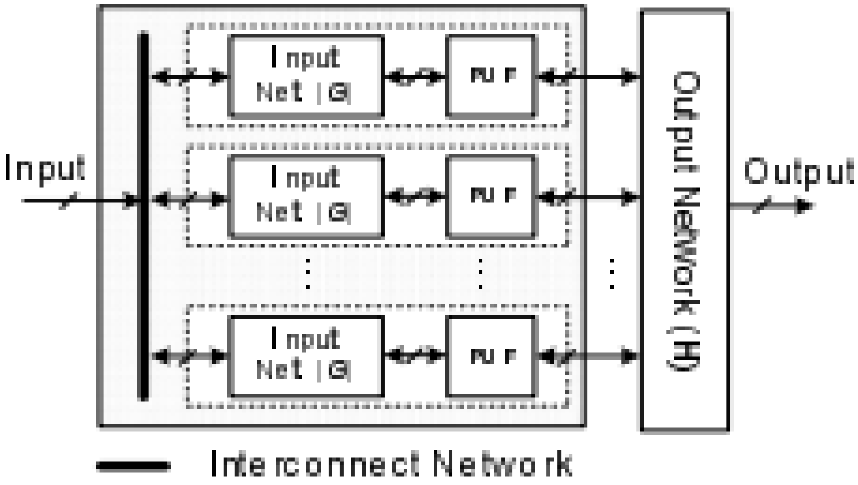


Figure 2.4: Light-weight Secure PUF Design (Reproduced from [3])

Majzoobi *et al.* [3] explained about some cases where the PUFs were vulnerable like using reverse engineering and then explained their methodology to secure the PUFs.

2.4.2 SR- Latch Based PUF

The PUF design is used for security applications. It is not easy to implement a SR latch PUF on FPGA. Ardakani *et al.* [6] explained the NAND based design and how they used it for area efficiency and introduced two methods to generate the PUF responses. The first method here shows that they used two switches which are also being used as selection lines for two MUXs and there are 4 LUTs in which they are using only 2 LUTs for generating the output response. In this Mux1 is used for multiplexing the signals which they got from the first two LUT’s and the second

Mux is used for the remaining two. They got the responses with the help of switches but alternatively varying them along with the enable. Here if they get the stable state they consider the output of the first MUX as the response and the second method here deals with the metastable state which is the case where the circuit will not be able to decide whether the output is one or zero. In this case they used a counter to count the number of oscillations and for these several responses they calculated a mean value (as this paper is on latch the outputs differ like the inputs). Coming to the results the uniqueness (one of the main factors for PUFs) is almost expected to be 45.4% (ideal value is around 50%) when the synthesis is done on several number of devices of the same kind.

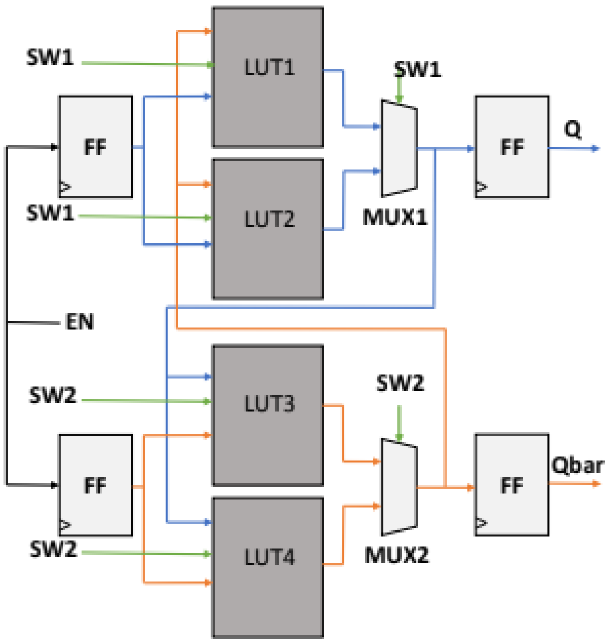


Figure 2.5: Design of a SR-Latch PUF (Reproduced from [6])

Habib *et al.* [5] explained similar work to the above one, they implemented a NAND based SR-Latch on FPGA and SOC devices. As we know there are 4 LUTs in each SLICE and 2 SLICES in each CLB. They compared the results on basis of uniqueness, robustness, temperature variations on Zynq devices and 25 different FPGA boards. The background work for this paper was based on Arbiter PUF, Ring Oscillator PUF and some more PUFs. As the Arbiter PUFs are affected by some machine learning attacks and Ring Oscillator PUFs are affected by the environmental changes or supply voltage. Similarly, the other PUFs like SRAM PUF, Latch PUF, and Butterfly PUF were studied and each of them has some drawbacks. Coming to the design methodology, a latch is used which is made of NAND gates which are placed in two different LUTs and using a control signal, the latch was introduced into a metastable state environment. In the metastable state as both the inputs (S, R) are 1 because of which the outputs will not be stable and for this the counters had been used to know the oscillation count as the output is so fast that we will not be able to see it with the human eye. So, once the latch is stable again then the value which is displayed on the counter will be stored into the block RAM (BRAM) and is displayed on the PC using an EPP. Coming to the selection of a particular latch they used a 9-bit MUX. The major drawback was that at a given time we can only configure a single latch only and that is a major reason to use only a single counter for counting the number of oscillations in a metastable state. In this research, on average the latch count that does not oscillate (zero value) is approximately 46%. Similarly, implemented the same design on Zynq 7000 chip at a voltage 1V to improve the performance and flexibility of the PUF.

2.4.3 Enhancement of PUF's Signature using RS Latches

The PUF will be more secure when it has a large response bit length and showed an example comparing 192-bit and 128-bit responses, where the first case (192-bit) is better than the second one (128-bit). But there is a disadvantage with 192-bit length as the bit length is too large and some of the bits will be inconsistent which shows that using the 128-bit will be a better idea. Yamamoto *et al.* [4] explained about the RS latch based PUF which they proposed and how it gives the random responses based on the locations. This was tested on almost 40 FPGA boards.

The boards belong to two different families. They are Spartan 6 and Spartan 3E. The error-rate is very low and the PUF also proves its reliability.

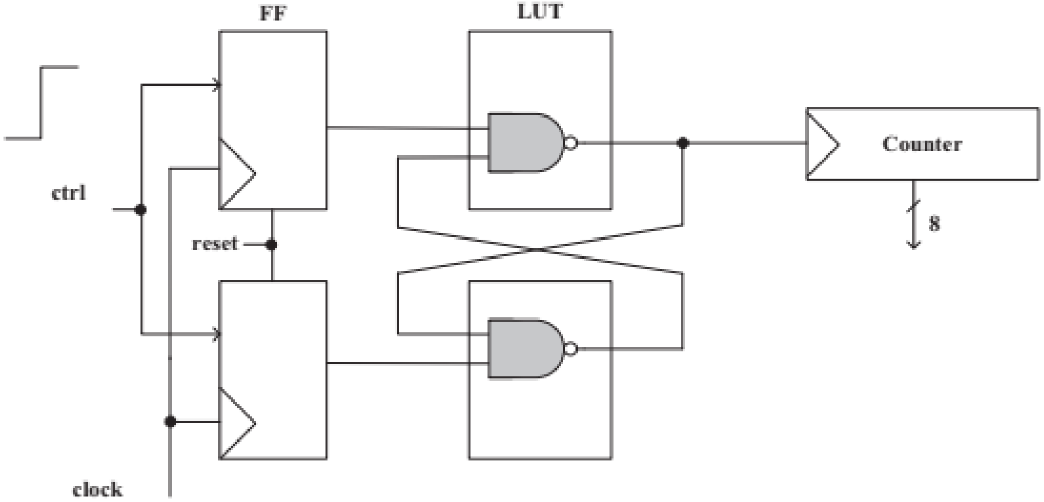


Figure 2.6: Design of a RS-Latch based PUF (Reproduced from [5])

2.4.4 SR-Flip Flop Based PUF

The SR Flip Flop [9] is designed using 4 NAND gates in cross-coupled manner. The metastable state of the SR Flip Flop was focused more as it is an inconsistent state where both the inputs (S & R) are 1. The outputs Q and Qbar will not be consistent and this state is known as race-around condition and the outputs will always be excited in this state. So, using the cross-coupled connections of the NAND gates the delays were introduced into the design. So, by changing the lengths and widths in the design using HSPICE the delays between the cross-coupled paths are varied and they avoided the inconsistency in the outputs. So, due to manufacturing differences and the process variations of the PUF should give different output for different device. Monte-Carlo simulations were performed on the PUF to check its randomness and the PUF's robustness was also tested by performing the simulations for the same PUF design in 32nm, 45nm, and 90nm technologies. This work has been done for 1-bit, 4-bit, 8-bit, 16-bit, 32-bit, 64-bit and 128-bit responses where the SR Flip Flop which were already in the registers were used without using any external support. There is a MUX in the design to send the inputs 1 and 0 using the selection line which is named ‘mode’ in this work.

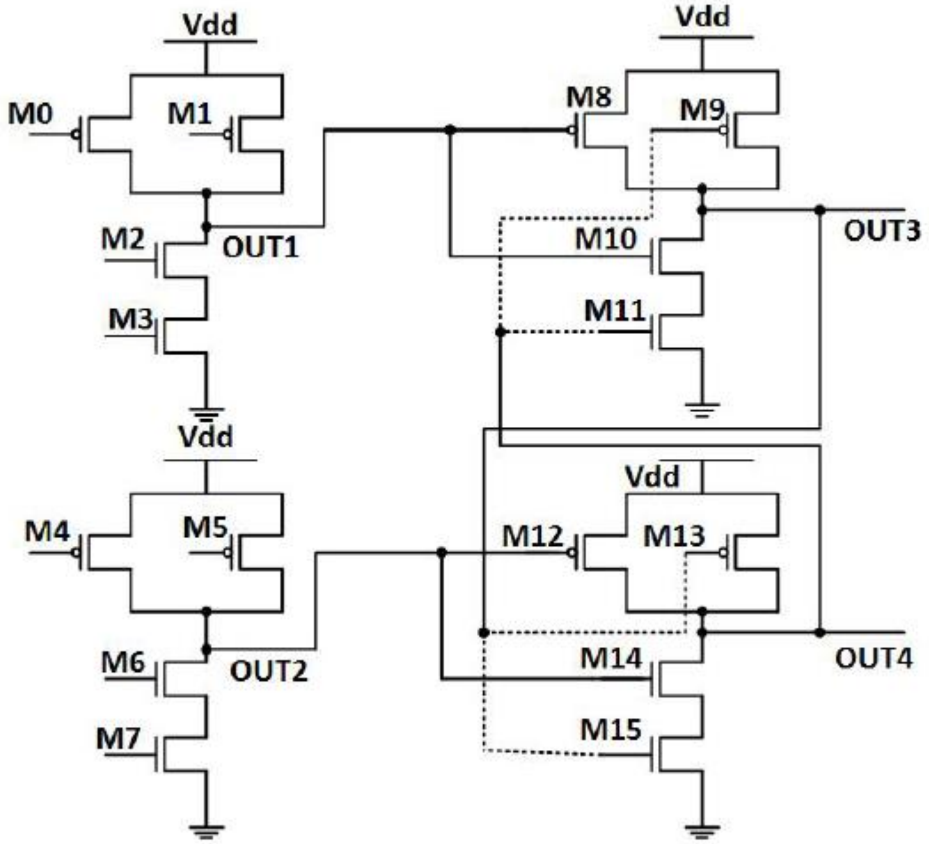


Figure 2.7: Transistor Level Design of a SR-FF based PUF (Reproduced from [9])

The design was also simplified into a centroid structure where each MUX can control up to 4 NAND based SRFFs. This design will reduce the area constraints. When this was tested across different chips this resulted in different outputs, which clearly shows that every device has its own signature response. This clearly proves the uniqueness of the PUF designed. This work was tested using 1000 iterations using 1V as the supply voltage. The PUF exhibits its reliability and also satisfies all the properties which a PUF should have.

2.5 Chapter Summary

The background study on different PUF designs is performed and every PUF design have their unique advantages and disadvantages. The RO PUF and the Arbiter PUF both are unique PUF designs with multiple advantages but the major constraint for both the PUFs is that the area consumption is very high and the aim of designing a PUF is to perform a low power and less area applications along with the hardware security. There are other drawbacks for optical PUF and memory PUF as well as it will be very expensive if the memory, we use increases. Taking these drawbacks into consideration, we propose a PUF design which we will be presented in Chapter 3.

CHAPTER 3: PROPOSED SR FLIP FLOP BASED PUF DESIGN

In this Chapter, we discussed about the work which we proposed for this thesis. It discusses about the toggle or race-around condition and how we introduced the delays between the cross coupled NAND gates using the slices in the FPGA devices.

3.1 SR-Flip Flop Design

The Set-Reset Flip Flop (SR-FF) based PUF design is a unique PUF design which is designed using cross-coupled NAND gates. The regular design of SR- Flip Flop which we know as a reference for this work is shown in Figure 3.1 below. This design is the major reference for our PUF design as this shows how the SR Flip Flop is built using cross-coupled NAND gates.

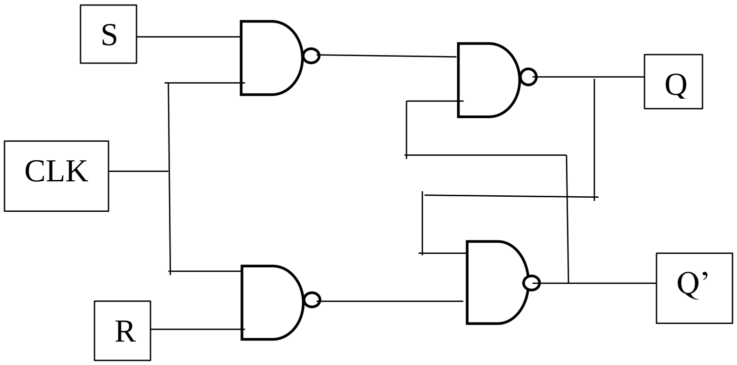


Figure 3.1: Basic SR-Flip Flop

For the above SR Flip Flop, we have a truth table which shows the behavior of the design. It shows the change of states when certain inputs are given. Here, we have ‘CLK’ which is clock which should always be as ‘1’ to produce the outputs otherwise if the clock is zero the Flip Flop design will not work. The truth table clearly shows that when the Reset (R) is 1 then the output Q' where Q will be 0. Coming to the other case, when the value of Set (S) is 1 then the output will be Q and Q' will be zero. When both the inputs are assigned to zero then the output will depend on the previous state. The last case we have is when both the inputs are assigned to one, in this case there will be a race-around condition or toggle condition. The design will be in a state where it is unable to choose a proper output that is either 1 or 0. So, the output will toggle continuously, and it is inconsistent.

Table 3.1: General Truth Table for SR Flip Flop

Clock	S	R	Q	Q'	State
1	0	0	Q	Q'	Previous State
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	1	1	Race Around

To overcome this major problem of inconsistency due to the race around condition we proposed a SR Flip Flop based PUF design which is similar to the regular SR Flip Flop with some design modifications. These design modifications are done to provide a consistent output when both the inputs S=R=1. We used the FPGA as a medium to produce outputs using the proposed PUF design.

3.2 Proposed SR-Flip Flop PUF Design

As stated earlier, the proposed design has four NAND gates with cross-coupled connection between NAND 3 and NAND 4 gates which is similar to SR- Flip Flop design which clearly shows that the output is based on the delay inserted within the cross-coupled connection of the design. We have two main inputs S, R which comes into the multiplexers which are in the design. “Mode” acts as a selection line for the multiplexers which helps both the multiplexers to select in the desired inputs. We have a “Trigger” button which is a common input for both the multiplexers present in the design. The above shown design modifications are used for construction of this reliable SR Flip Flop based PUF.

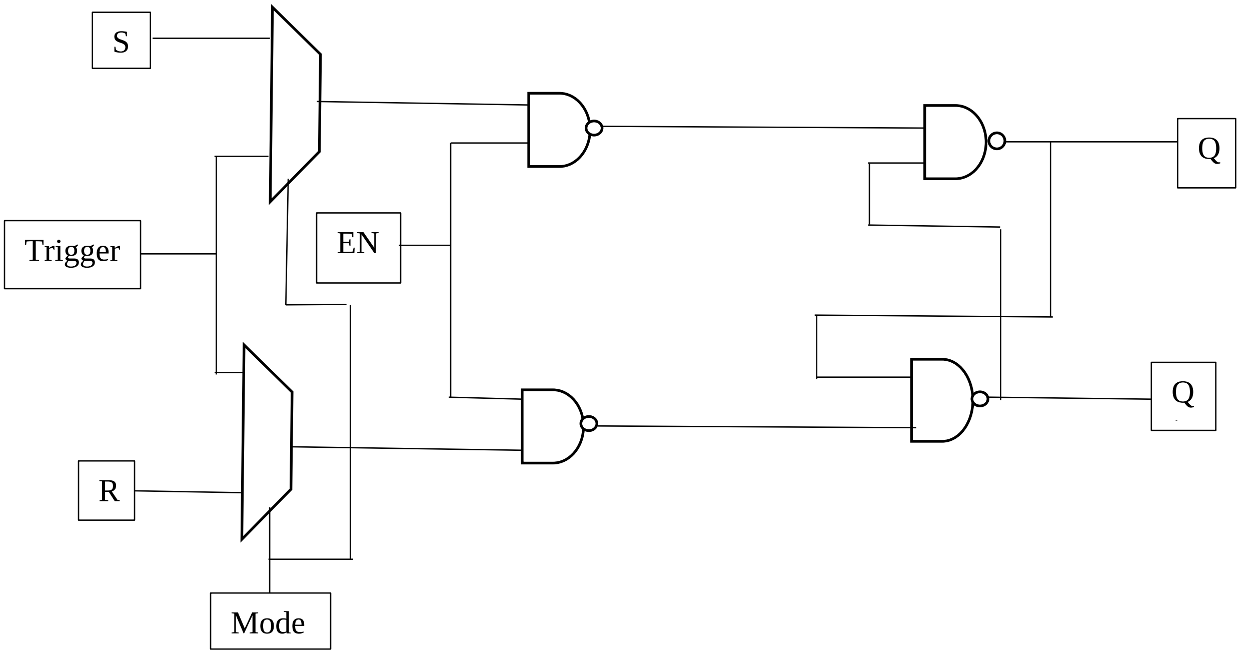


Figure 3.2: Design of the Proposed SR-Flip Flop PUF

Coming to the working of the above PUF design, which is shown in Figure 3.2, firstly, we generated a bit file which is programmable in FPGA. We use Adept to dump the bit file into the FPGA. We operate the inputs using the mode button, if the mode is selected as 0, the design is in Regular mode and when the mode is 1 then the design is in PUF mode. In the regular mode, the above design works as a basic SR Flip Flop. We operate this mode with the switches assigned to the inputs using the implementation constraints file (UCF) where we assign all the controls generally and the second mode is the PUF mode where the modified design comes into play. We have a Trigger which is assigned to a button on the FPGA. Whenever the mode is 1, the regular mode will not work, and both the inputs will be assigned to one. When S=R=1, the Trigger button is used to produce a consistent output. This output satisfies all the properties of the PUF. The construction is explained above, and the implementation is explained below in a detailed way.

3.2.1 RTL Design and Implementation

We used Xilinx ISE 14.7 tool as a medium for implementing the PUF design. There are two types of views for the design within the tool. The first view is the RTL view as shown in Figure 3.3. It consists of Enable which is always one in order to make the PUF design work. The other inputs are S and R along with Trigger button and mode as selection line. The Q and Qbar are the outputs for the design.

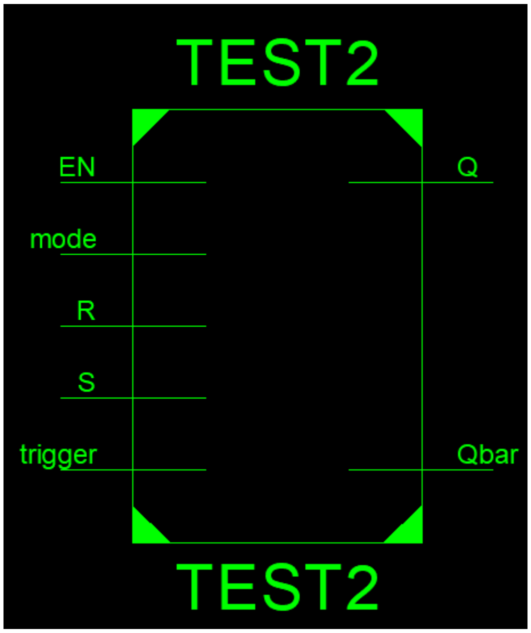


Figure 3.3: RTL View of the Proposed PUF Design

It looks like a block diagram for the design and when we go forward into the internal design of this block, we can see the gate level diagram of our PUF Design as shown in the Figure 3.4. It is the same design which we had in the Figure 3.2. But this is the image taken from the Xilinx ISE 14.7 tool. Coming to the second view which is known as Technology view as shown in Figure 3.3, where we will be able to see how all the gates are assigned to internal Look Up Tables (LUTs) and the slices.

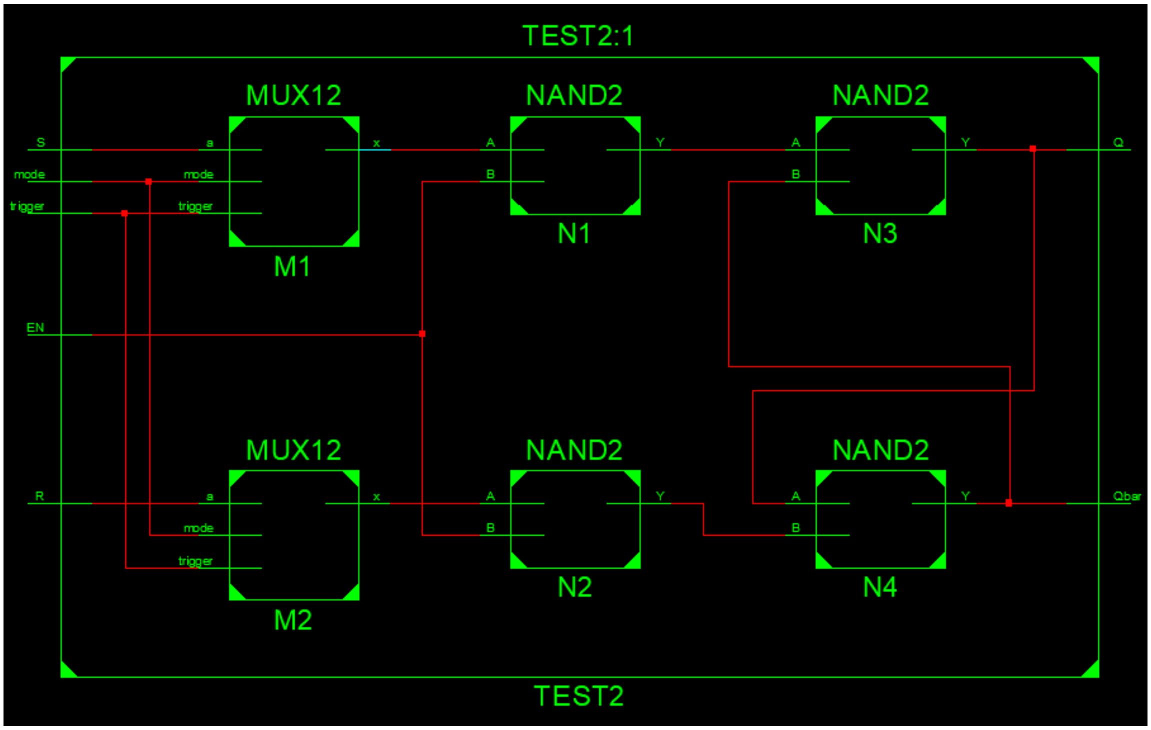


Figure 3.4: Block diagram for proposed PUF Design

Figure 3.4 shows all the internal circuit connections which will make the job of debugging easy. When we started this work there were many constraints which we came across. We used this view to debug all those problems which made our work easy. We can also have a clear idea if there are any inter-connections which are wrongly connected as we know that the cross-coupled connection is very important for this design and should be implemented correctly. There are two ways that we can use to debug the errors in the design. We explained the other way below which is shown in Figure 3.5.

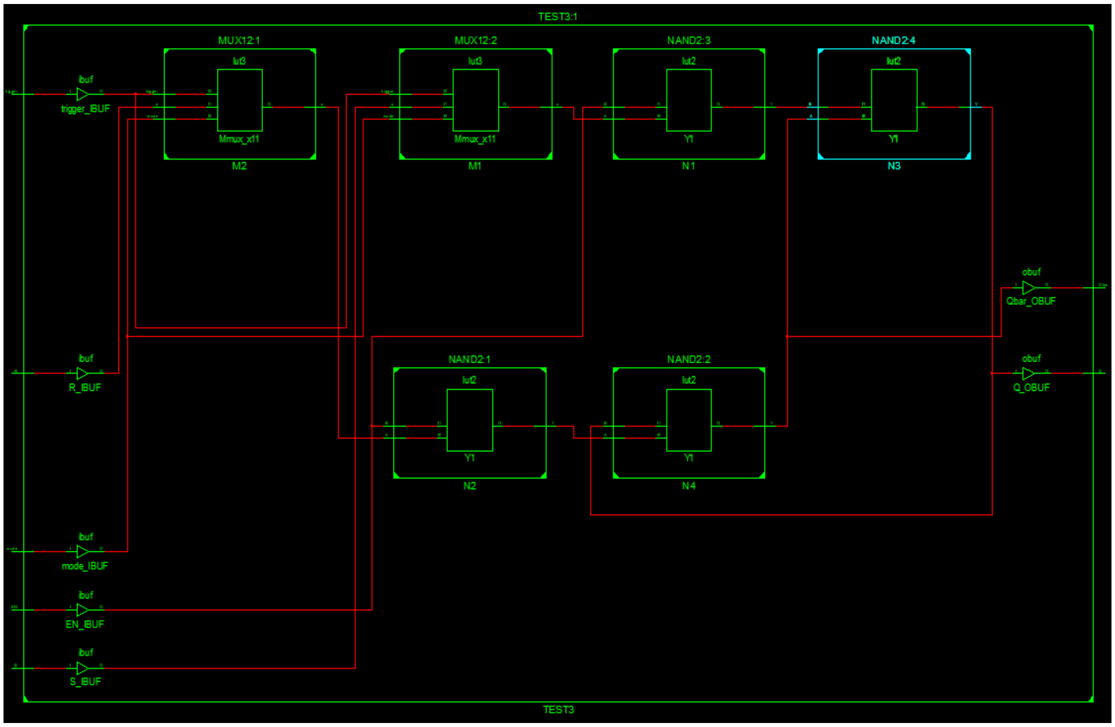


Figure 3.5: Technology View of the Proposed PUF Design

The Technology view clearly shows all the input and output buffers which are assigned to inputs and outputs, respectively. We can see those buffers and we can also see how the gates used in the design are assigned to the LUTs. As the number of registers increase there will be an increase in the usage of LUTs in addition to increase of Flip Flops. If we see in the Figure 3.4 below, the Q_OBUF and Qbar_OBUF are the buffers where we have the cross coupled connection. They are buffers where we introduced the delay using the process variations. We assigned these buffers to different slices within the different locations of the FPGA device.

3.2.2 Manual Routing

As explained above, we manually assigned the Q_OBUF and Qbar_OBUF to different slices. For this we need to explain about the Slices and the locations of the FPGA which we used for this work. We used FPGA editor which is an inbuilt tool in Xilinx to manually port all the connections.

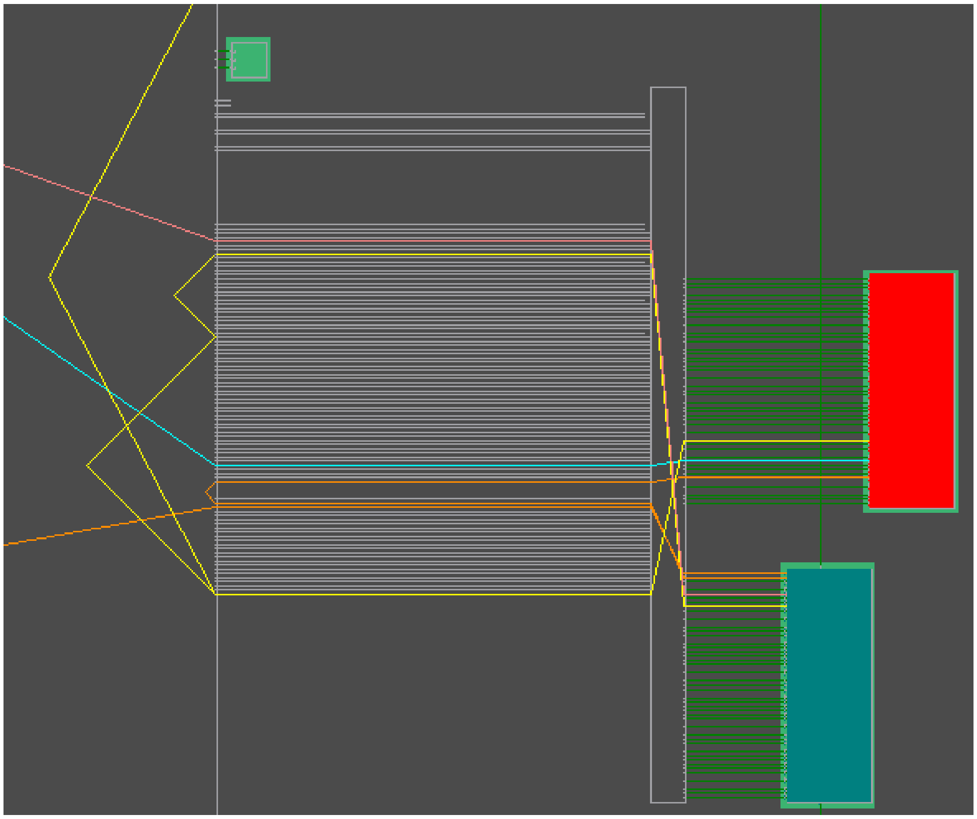


Figure 3.6: Cross-Coupled Connection between Slices

The FPGA device which we used for this work is Spartan 6 Digilent ATLYS. For this FPGA device there are 16 locations where we have 54,576 Slice registers. Out of which 27,288 which is half of the count is occupied by Slice LUTs. For example, if we consider an 8-bit PUF design, it occupies 56 Slice LUTs which are used as logic for the output generation which is just 1 percent of the Slice registers being utilized. So, it is very clear that the utilization of the Slice registers is very less. Our proposed design produces the output based on the propagation delays which are based on the LUTs and the path from the input to the output.

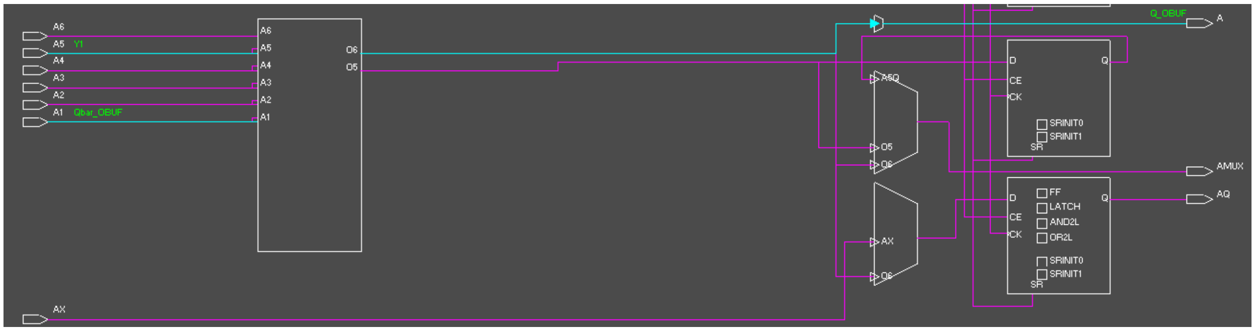


Figure 3.7: Internal View of a LUT in a SLICE

We can clearly see the LUT assigned to Q_OBUF in the Figure 3.6. The LUTs are combined automatically but they are assigned manually. We did not use any optimized instantiated primitives to maintain the hierarchy of the design. The delays vary in the PUF mode due to the cross coupled connections which is implemented manually. This produces the bitstreams according to the required output bitstream. All the delays will be in nanoseconds and will vary at each location of the FPGA maintaining uniqueness and randomness which will make the PUF design strong from the hardware security attacks. The Xilinx XPower Analyzer is the inbuilt tool in Xilinx ISE 14.7 which is used to analyze the usage of power and behavior of the PUF design in various environmental conditions like ambient temperature which are elaborated in Chapter 4.

3.3 Chapter Summary

In this chapter, we discussed about the race-around condition in FPGA and how it makes the output toggle with high speed which cannot be seen even with the human eye. Using the proposed PUF design we overcame the race-around condition by inserting the delay caused by routing the cross-coupled connections within the slices which we assigned by using the FPGA

Editor manually. We also analyzed the behavior of the PUF when the environmental conditions are changed. Table 3.2 is the updated truth table which is based on the PUF design. The experimental results are discussed in Chapter 4.

Table 3.2: Proposed SR Flip Flop based PUF Truth Table

Enable	S	R	Q	Q’	State
1	0	0	Q	Q’	Previous State
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	1	1	Output depends on the delay which is generated due to cross-coupled NAND gates.

Figure 3.8 shows the overall workflow of the PUF design and the steps which we followed when we performed the tests. It shows how the inputs were given and explained the key role of enable and trigger for generating regular and PUF based unique outputs, respectively. As shown in Table 3.2 the regular output is generated when S and R are opposite to each other and the PUF based unique output bitstreams are generated when we have S=R=1 based on the delays by exploiting the race-around condition.

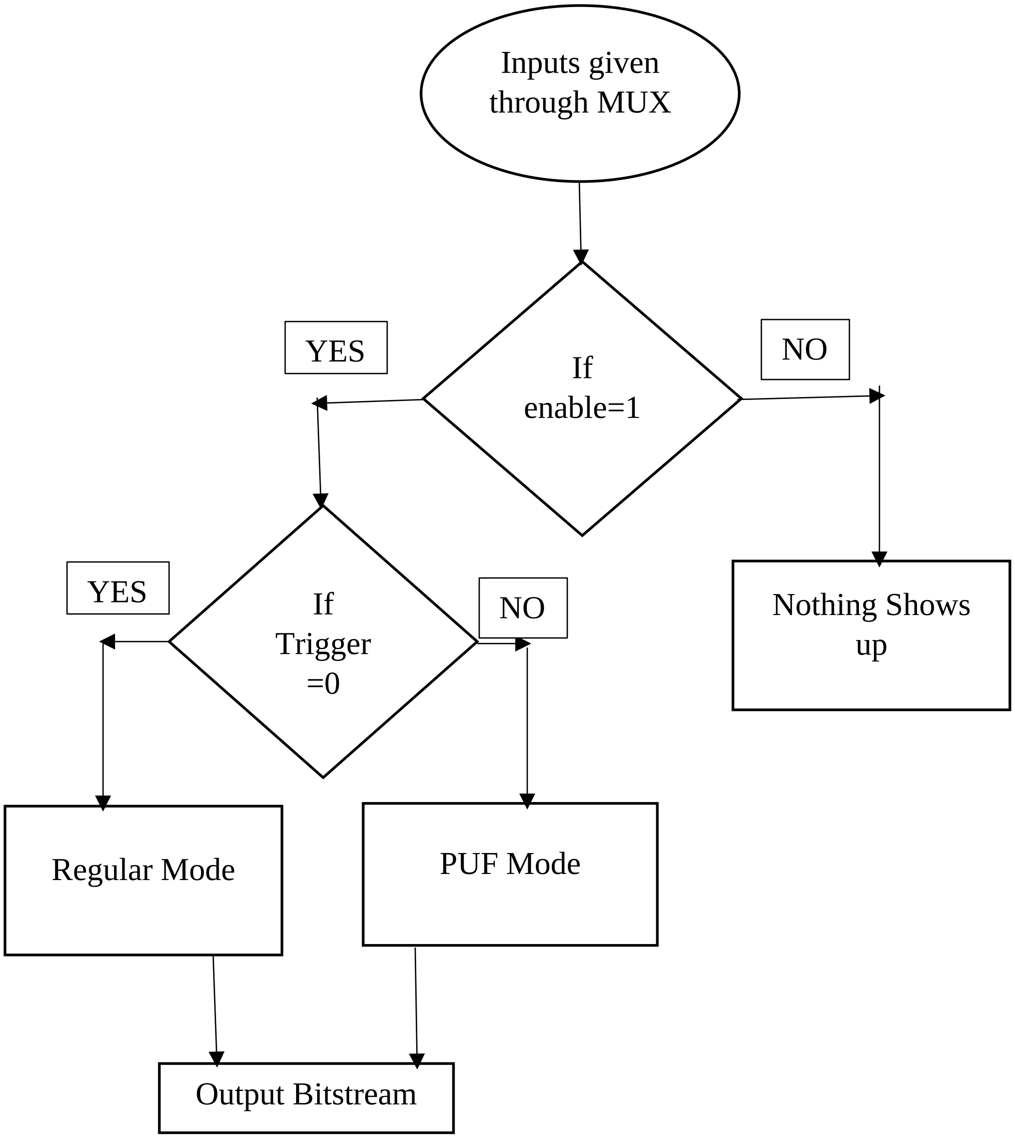


Figure 3.8: Workflow of the Proposed PUF Design

CHAPTER 4: EXPERIMENTAL RESULTS

In this chapter, we present the experimental results collected for two Digilent ATLYS FPGA boards. We present the results for 1, 4, 8, 16, 32, 64 long SRFF PUFs. The data obtained for thirteen other boards is similar and is presented in Appendix A.

4.1 Chip-scope Simulations

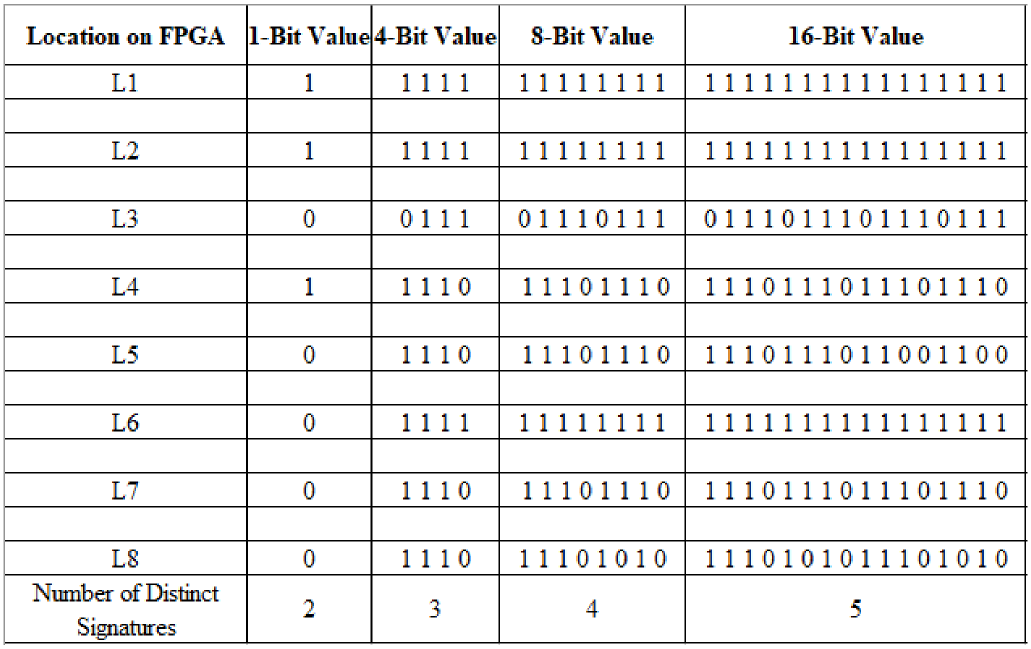
Chip-scope pro is an ILA core tool which is in built in the Xilinx ISE 14.7. It is used as a logic-analyzer and it plays a major role in debugging the FPGA devices. As the number of LED count is only 8 on the FPGA board which we used for this thesis, we opted chip-scope to automate the process. Using the chip-scope pro analyzer we can get the results for any bitstream which is higher than the 1-bit count.

4.2 FPGA Implementation and Synthesis

For each board, we present the experimental results for 1-bit, 4-bit, and 8-bit bitstreams which were tested manually on LEDS using Dip switches and Buttons which are located on FPGA Board. For the 16-bit, 32-bit and 64-bit we automated the process using Chip-scope pro which is an inbuilt tool in Xilinx ISE 14.7.

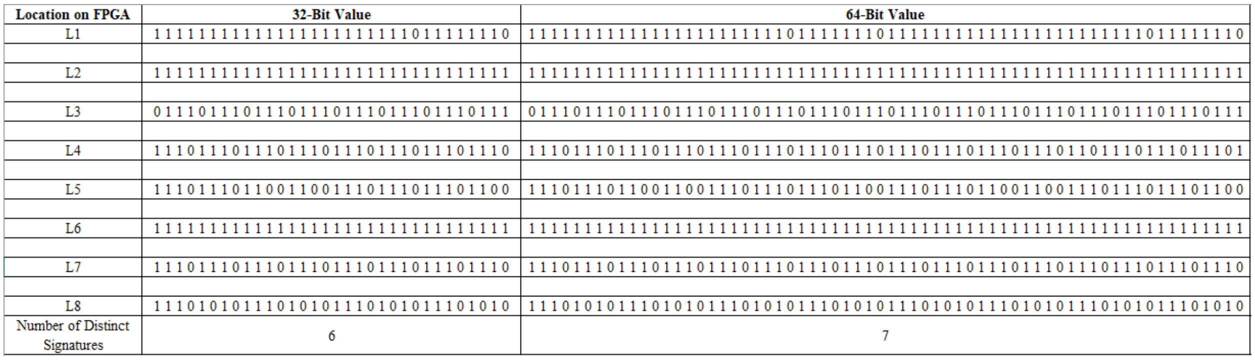
Tables 4.1 and 4.2 present the results obtained from the first FPGA board. As explained in the Chapter 3, we used slices which are in the FPGA and assigned the cross-coupled gates to certain LUTs to leverage the manufacturing variations of that device. Because of this limitation, we can have variation in the delay which we actually used for the production of the unique PUF signature which varies for each device as the manufacturing variations will not be the same for each device.

Table 4.1: Board-1 Experimental Results for 1, 4, 8 and 16-Bitstreams



Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	3	4	5

Table 4.2: Board-1 Experimental Results for 32 and 64-Bitstreams



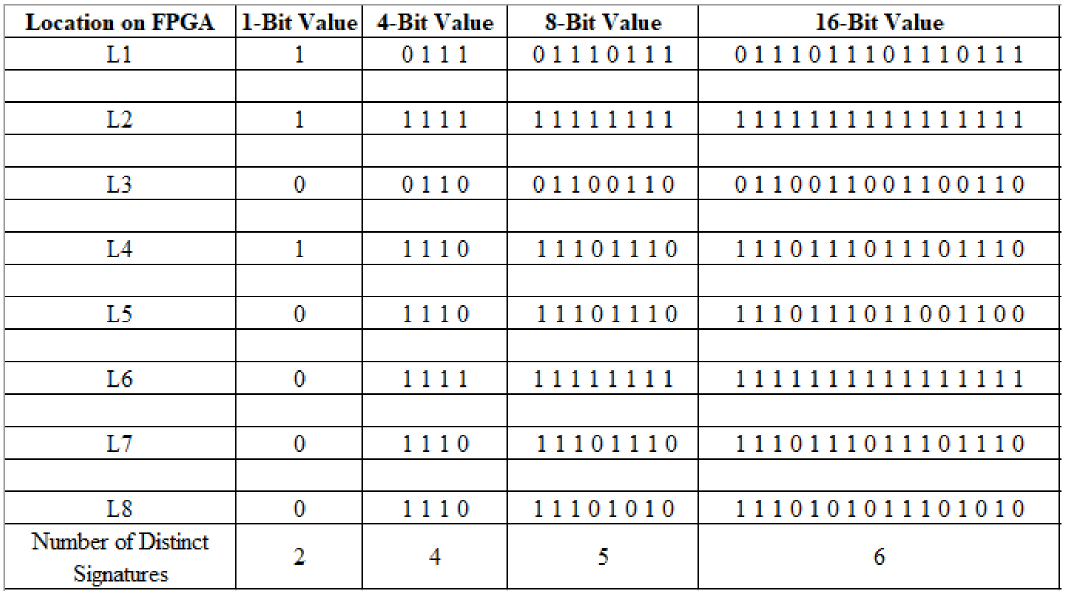
Location on FPGA	32-Bit Value	64-Bit Value
L1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	6	7

Here in the table shown above, the PUF design is tested in the 8 locations out of 16 locations present in the FPGA. We can clearly see that as the bit length increases from 1-bit to 64-bits, the variation of bitstreams among the locations is shown. The distinct pairs are also observed, and two table were showing the records of distinct signatures for all the 15 devices.

Tables 4.3 and 4.4 show the results for FPGA device 2. When we compare both device 1 and device 2, we can clearly see certain differences in the bitstreams produced on the FPGA. It clearly shows the property of uniqueness i.e., the signature remains constant after testing several times. The difference between the bitstreams occurred because of the delay between the cross-coupled NAND gates (N3 and N4).

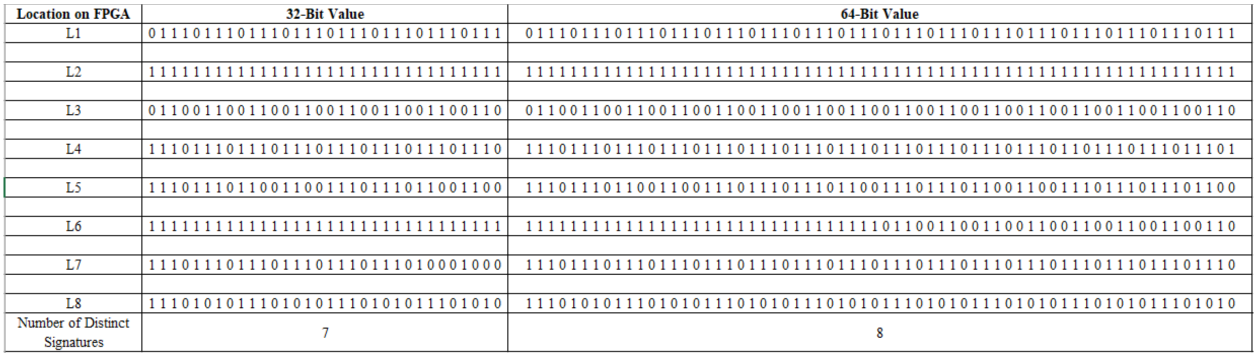
The synthesis results are tested in the room temperature and the outputs resulted for the PUF are the same and did not vary from 15C to 25C which shows the “robustness” of the PUF. The PUF proposed also proves the property of the “uniformity” as it produces the same output response after several iterations. As the response signature generated from the FPGA bit is unpredictable, the design also proves the “randomness” of our PUF design. We tested the PUF design among fifteen (15) different FPGA devices which belong to the same Spartan 6 family which produced their unique bitstream in each phase.

Table 4.3: Board-2 Experimental Results for 1, 4, 8 and 16-Bitstreams



Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 0	0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	4	5	6

Table 4.4: Board-2 Experimental Results for 32 and 64-Bitstreams



Location on FPGA	32-Bit Value	64-Bit Value
L1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	7	8

The above tables show the comparison of the results for all the fifteen (15) FPGA devices for 1, 4, 8, 16, 32, and 64-bitstreams. When we see Table 4.1, it shows the result for device 1 for 1, 4, 8, 16 Bitstreams where we used “Location on FPGA” as L1, L2, L3, L4, L5, L6, L7 and L8, respectively. Similarly, we performed these tests for the other bitstreams as well. As we know there are two modes for this SR flip flop based PUF design. The first one is the normal mode and the second one is PUF mode. When we performed the synthesis among the fifteen (15) devices in normal mode, we have a normal response which we see in Table 3.1. Coming to the PUF mode, ten (10) FPGA devices have a similar value and the remaining five (5) FPGA devices have a different bitstreams. But as we know it is very clear that all the boards have their unique bitstream responses generated according to their manufacturing variations.

Tables 4.5 and 4.6 tabulate the results for the number of distinct signatures from each FPGA device as we vary the SR register bit width (n). We can observe that for eight distinct locations (L1 – L8) we get 7-8 distinct signatures for 64-bit size. We empirically observe a logarithmic relationship between the number of distinct signatures (m) and bit size, namely, $m \approx \log_2(n)$. In order to reliably distinguish eight sites on the FPGA, we recommend choosing $n \geq 128$.

4.3 Chapter Summary

We produced the results for all the bitstreams from 1 to 64-bit registers using our PUF design and we proved that the properties uniqueness, robustness, randomness, and uniformity are satisfied for the PUF design which we proposed. We compared the bitstreams produced among the different locations on the FPGA device as well as with the other boards. We used XPower Analyzer and tested the behavior of the PUF by varying the environmental conditions for two boards and observed the power consumption.

Table 4.5: No. of Distinct Signatures for First Eight (8) Devices

FPGA	**1-bit**	**4-bit**	**8-bit**	**16-bit**	**32-bit**	**64-bit**
Device 1	2	3	4	5	6	7
Device 2	2	4	5	6	7	8
Device 3	2	3	4	5	6	7
Device 4	2	4	5	6	7	8
Device 5	2	3	4	5	6	7
Device 6	2	3	4	5	6	7
Device 7	2	3	4	5	6	7
Device 8	2	3	4	5	6	7

Table 4.6: No. Of. Distinct Signatures for Next Seven (7) Devices

FPGA	**1-bit**	**4-bit**	**8-bit**	**16-bit**	**32-bit**	**64-bit**
Device 9	2	3	4	5	6	7
Device 10	2	3	4	5	6	7
Device 11	2	4	5	6	7	8
Device 12	2	4	5	6	7	8
Device 13	2	3	4	5	6	7
Device 14	2	4	5	6	7	8
Device 15	2	3	4	5	6	7

CHAPTER 5: CONCLUSION AND FUTURE WORK

We proposed a SR Flip Flop based PUF design on to the FPGA boards and collected the response data from fifteen (15) FPGA Spartan 6 Digilent ATLYS boards. The area utilization is very less which makes it area efficient and as the Spartan 6 FPGA uses 45nm the power consumption is very less. This PUF Design is tested at 25C room temperature and all the FPGA boards we tested this design on, have their unique values which showed no changes in the output response bit signature. This PUF design satisfied all the major properties of a PUF which are robustness, uniqueness, randomness, and uniformity. As future work, we can increase the bit width of the PUF to 128-bit and 256-bit and test them on other FPGA Boards such as Digilent ANVYL and also test the PUF design for varying environmental conditions and supply voltage.

REFERENCES

[1] C. Herder, M. D. Yu, F. Koushanfar, and S. Devadas. “Physical Unclonable Functions and Applications: A Tutorial”. *Proceedings of the IEEE*, 102(8):1126–1141, Aug 2014.

[2] U. Rührmair and D. E. Holcomb. “PUFs at a glance”. In *2014 Design, Automation Test in Europe Conference Exhibition* (DATE), pages 1–6, March 2014.

[3] M. Majzoobi, F. Koushanfar, and M. Potkonjak. “Lightweight secure PUFs”. In 2008 IEEE/ACM International Conference on Computer-Aided Design, pages 670–673, Nov 2008.

[4] D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, M. Takenaka, and K. Itoh. “Variety enhancement of PUF responses using the locations of random outputting RS latches”. Journal of Cryptographic Engineering, 3(4):197–211, Nov 2013.

[5] B. Habib, J. P. Kaps, and K. Gaj. “Implementation of efficient SR-latch PUF on FPGA and SoC devices”. Microprocessors and Microsystems, 53:92 – 105, 2017.

[6] A. Ardakani and S. B. Shokouhi. “A secure and area-efficient FPGA-based SR-latch PUF”. In 2016 8th International Symposium on Telecommunications (IST), pages 94–99, Sept 2016.

[7] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls. “The butterfly PUF protecting IP on every FPGA”. In 2008 IEEE International Workshop on Hardware-Oriented Security and Trust, pages 67–70, June 2008.

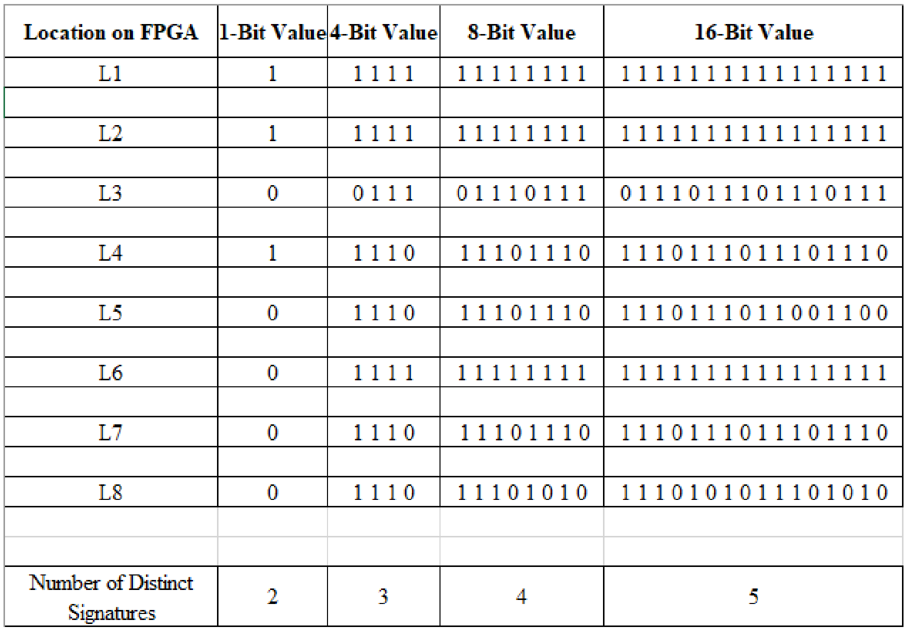
[8] S. A. Islam and S. Katkoori. “High-level synthesis of key based obfuscated RTL data paths.” In 2018 19th International Symposium on Quality Electronic Design (ISQED), pages 407–412, March 2018.

[9] R. P. Challa, “SR Flip-Flop Based Physically Unclonable Function (PUF) for Hardware Security” (2018). Graduate Thesis and Dissertations, University of South Florida.

[10] M. Patterson, J. Zambreno, C. Sabotta, S. Vyas, and A. Mills, "Ring Oscillator PUF Design and Results", 2011.

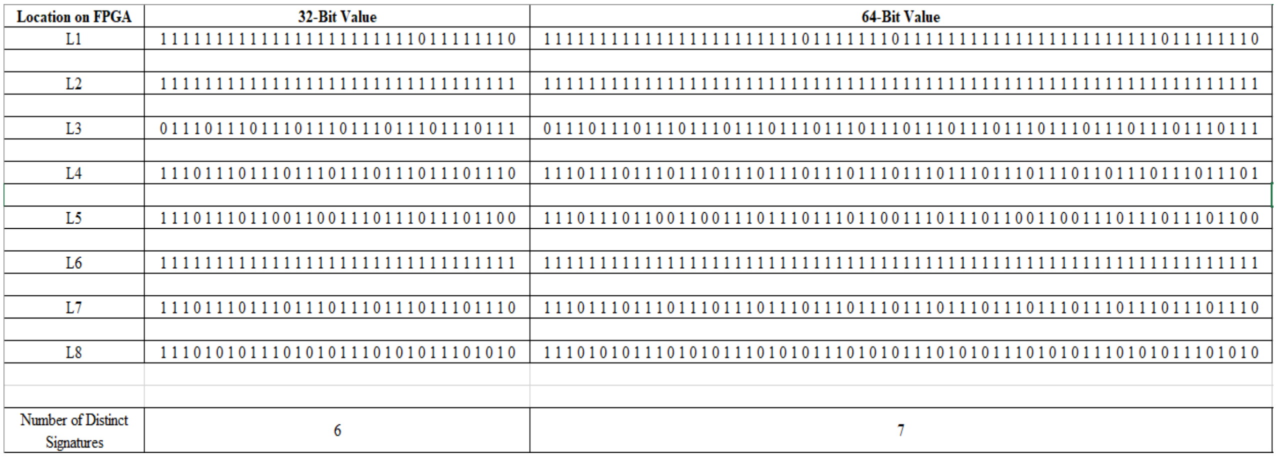
APPENDIX A

In this Appendix, we present all the raw data collected from 15 Spartan-6 Digilent ATLYS boards.



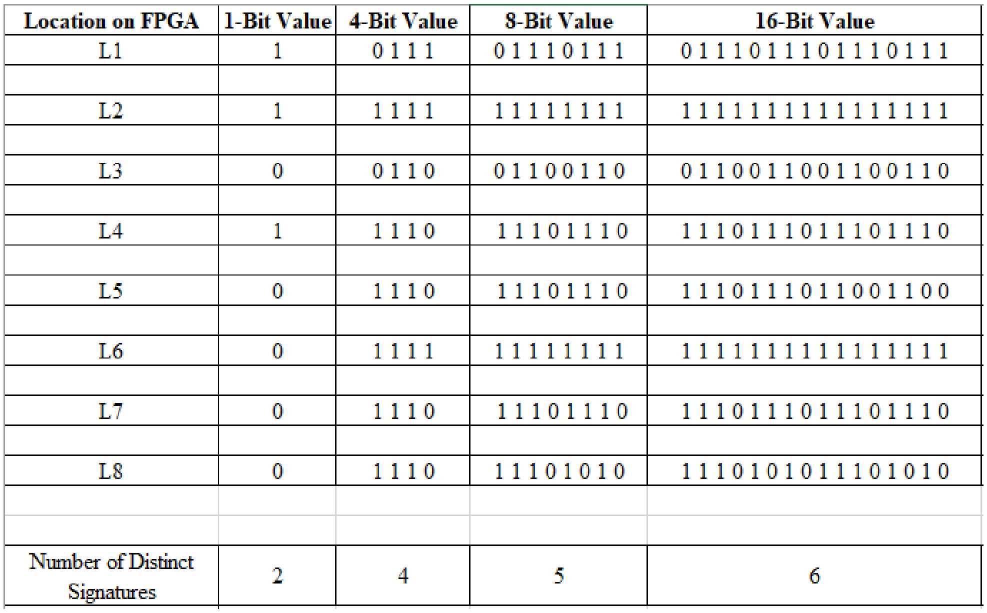
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1111	11111111	1111111111111111
L2	1	1111	11111111	1111111111111111
L3	0	0111	01110111	0111011101110111
L4	1	1110	11101110	1110111011101110
L5	0	1110	11101110	1110111011001100
L6	0	1111	11111111	1111111111111111
L7	0	1110	11101110	1110111011101110
L8	0	1110	11101010	1110101011101010
Number of Distinct Signatures	2	3	4	5

Figure A.1: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-1.



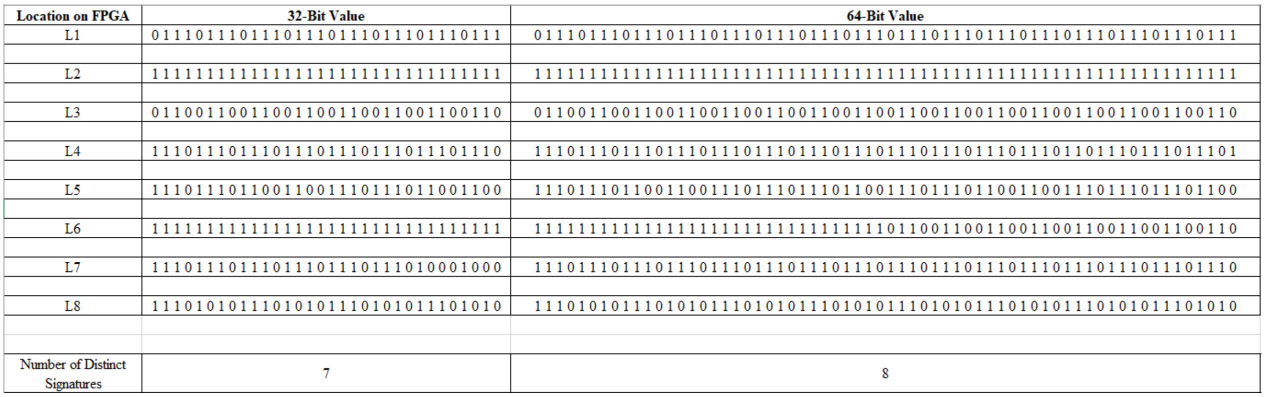
Location on FPGA	32-Bit Value	64-Bit Value
L1	11111111111111111111111011111110	1111111111111111111111101111111011111111111111111111111011111110
L2	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L3	01110111011101110111011101110111	0111011101110111011101110111011101110111011101110111011101110111
L4	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101101110111011101
L5	11101110110011001110111011101100	1110111011001100111011101110110011101110110011001110111011101100
L6	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L7	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101110111011101110
L8	11101010111010101110101011101010	1110101011101010111010101110101011101010111010101110101011101010
Number of Distinct Signatures	6	7

Figure A.2: Experimental Data for 32 and 64-bitstreams on FPGA Board-1.



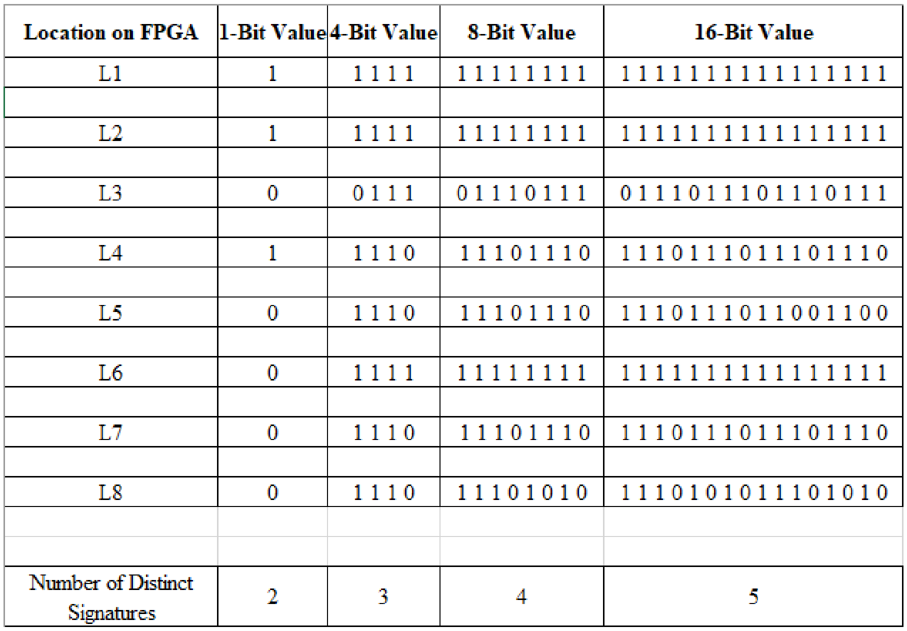
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 0	0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	4	5	6

Figure A.3: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-2.



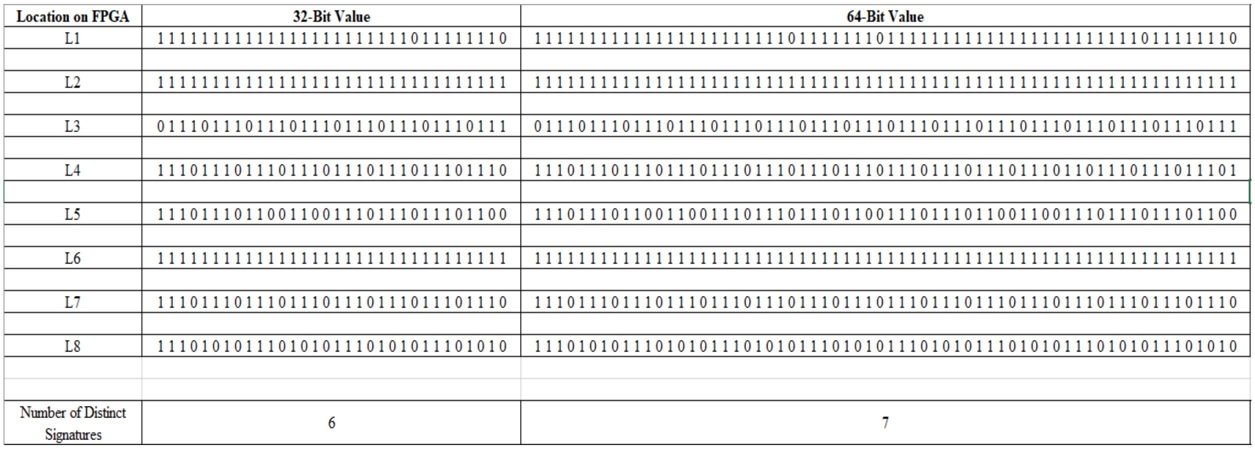
Location on FPGA	32-Bit Value	64-Bit Value
L1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	7	8

Figure A.4: Experimental Data for 32 and 64-bitstreams on FPGA Board-2.



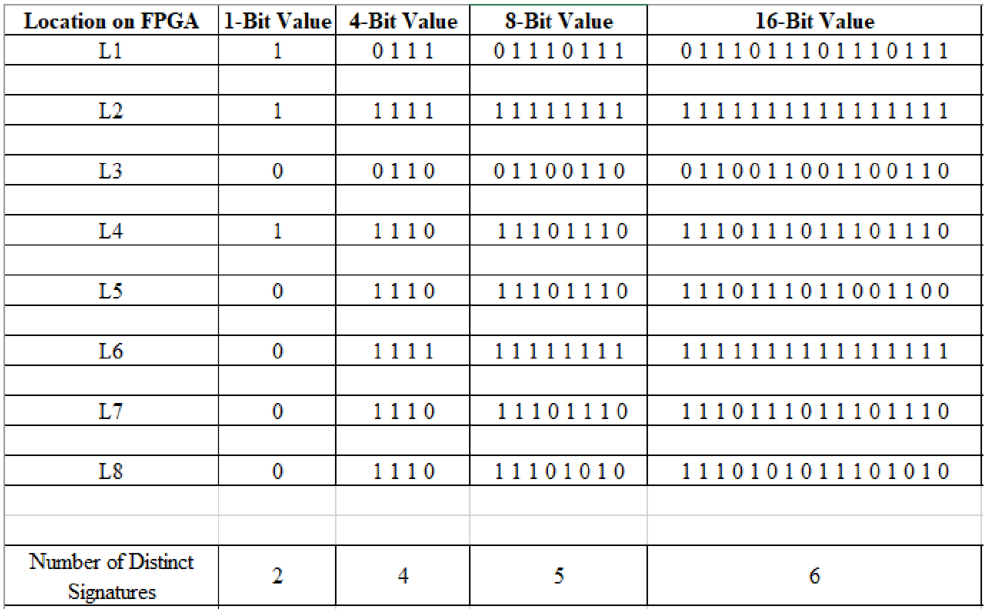
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1111	11111111	1111111111111111
L2	1	1111	11111111	1111111111111111
L3	0	0111	01110111	0111011101110111
L4	1	1110	11101110	1110111011101110
L5	0	1110	11101110	1110111011001100
L6	0	1111	11111111	1111111111111111
L7	0	1110	11101110	1110111011101110
L8	0	1110	11101010	1110101011101010
Number of Distinct Signatures	2	3	4	5

Figure A.5: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-3.



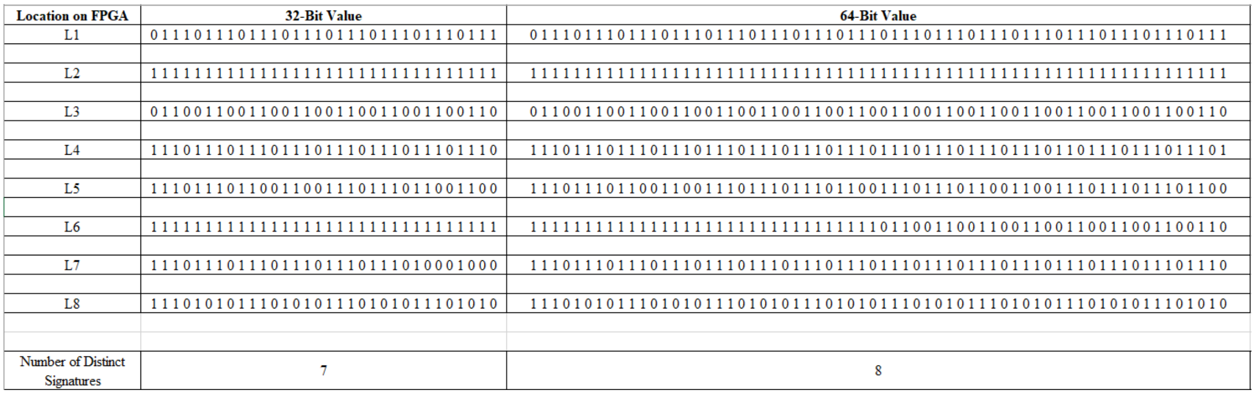
Location on FPGA	32-Bit Value	64-Bit Value
L1	11111111111111111111111011111110	1111111111111111111111101111111011111111111111111111111011111110
L2	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L3	01110111011101110111011101110111	0111011101110111011101110111011101110111011101110111011101110111
L4	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101101110111011101
L5	11101110110011001110111011101100	1110111011001100111011101110110011101110110011001110111011101100
L6	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L7	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101110111011101110
L8	11101010111010101110101011101010	1110101011101010111010101110101011101010111010101110101011101010
Number of Distinct Signatures	6	7

Figure A.6: Experimental Data for 32 and 64-bitstreams on FPGA Board-3.



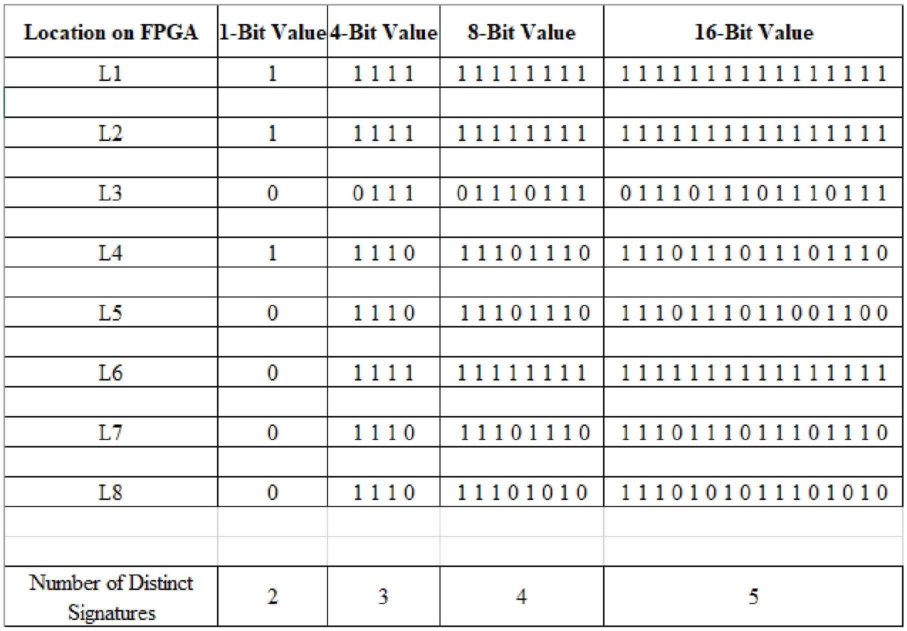
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 0	0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	4	5	6

Figure A.7: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-4.



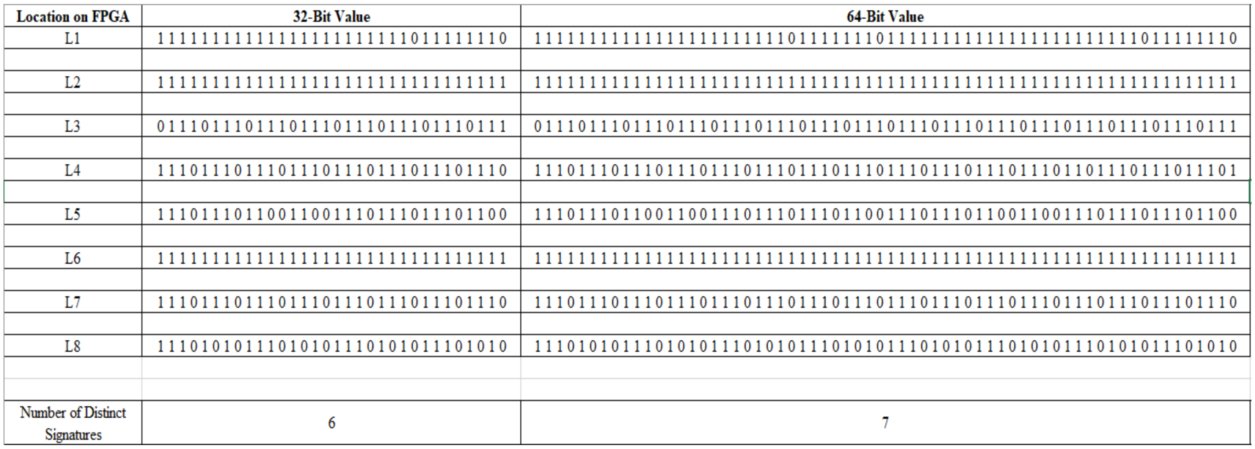
Location on FPGA	32-Bit Value	64-Bit Value
L1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	7	8

Figure A.8: Experimental Data for 32 and 64-bitstreams on FPGA Board-4.



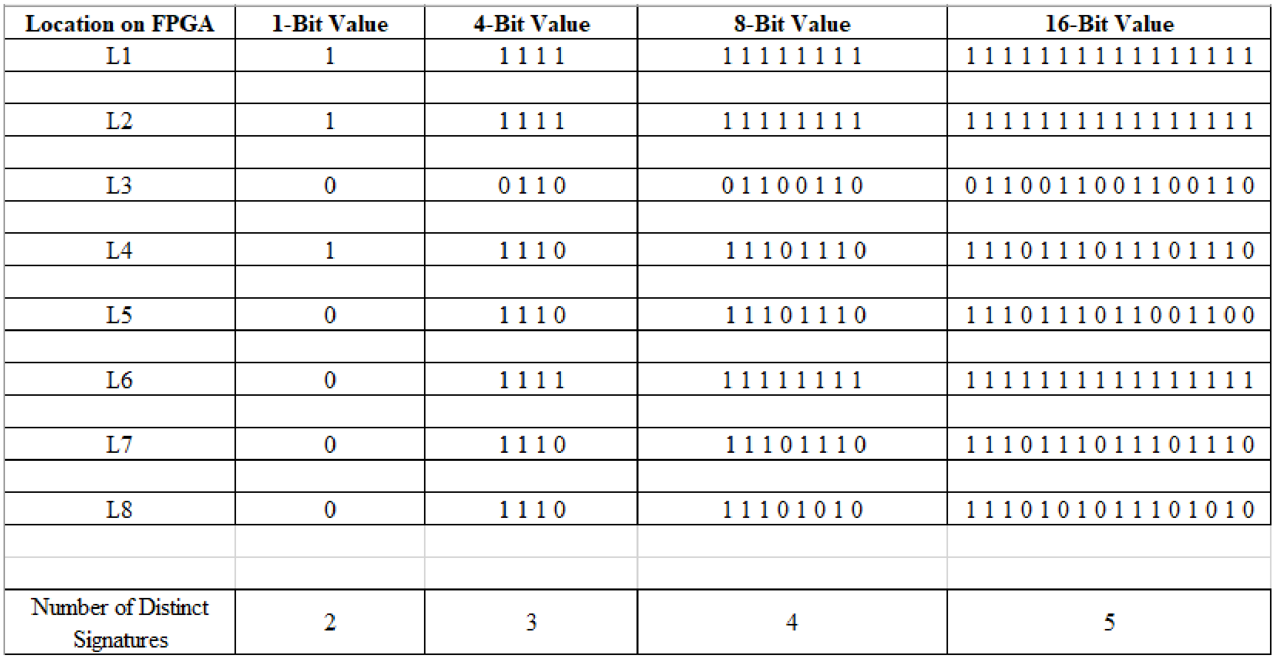
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	3	4	5

Figure A.9: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-5.



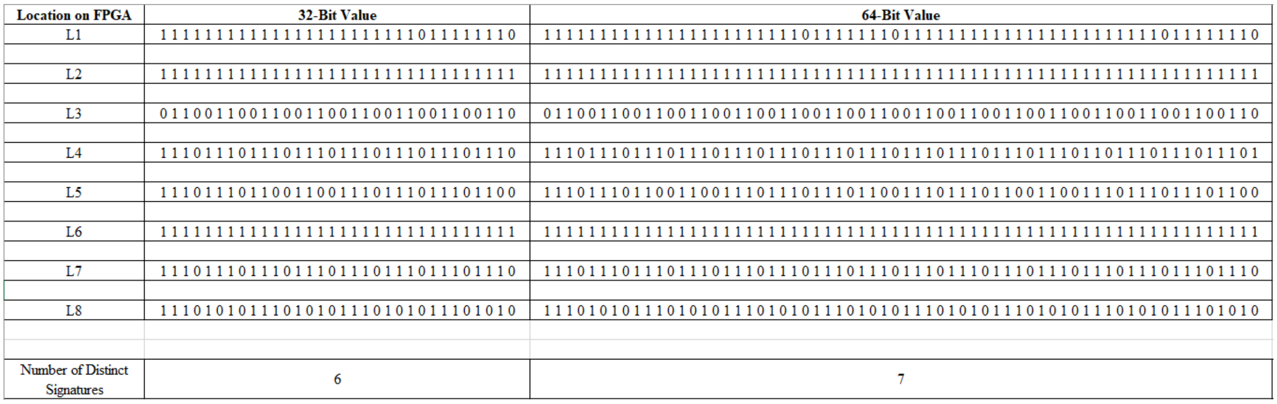
Location on FPGA	32-Bit Value	64-Bit Value
L1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	6	7

Figure A.10: Experimental Data for 32 and 64-bitstreams on FPGA Board-5.



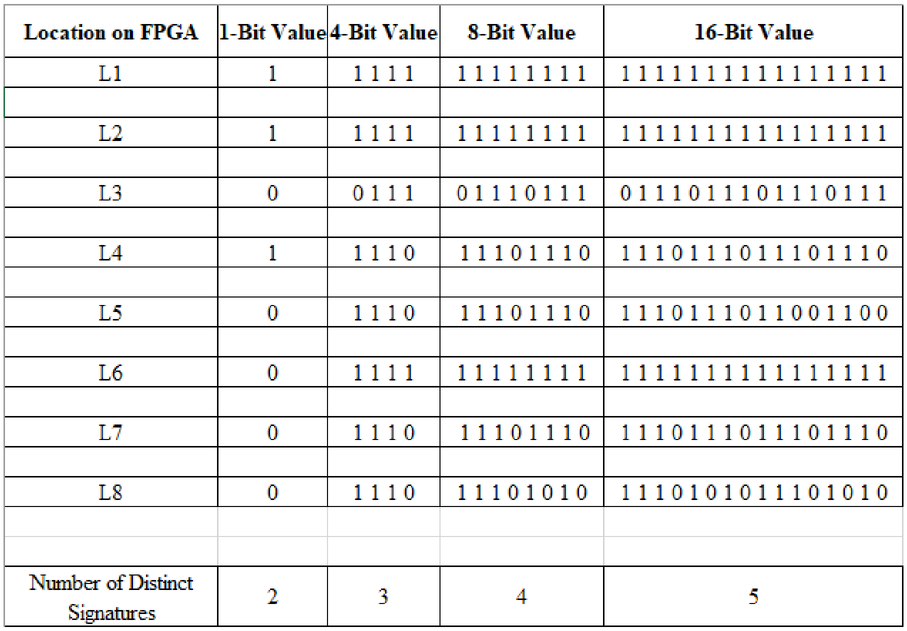
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 0	0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	3	4	5

Figure A.11: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-6.



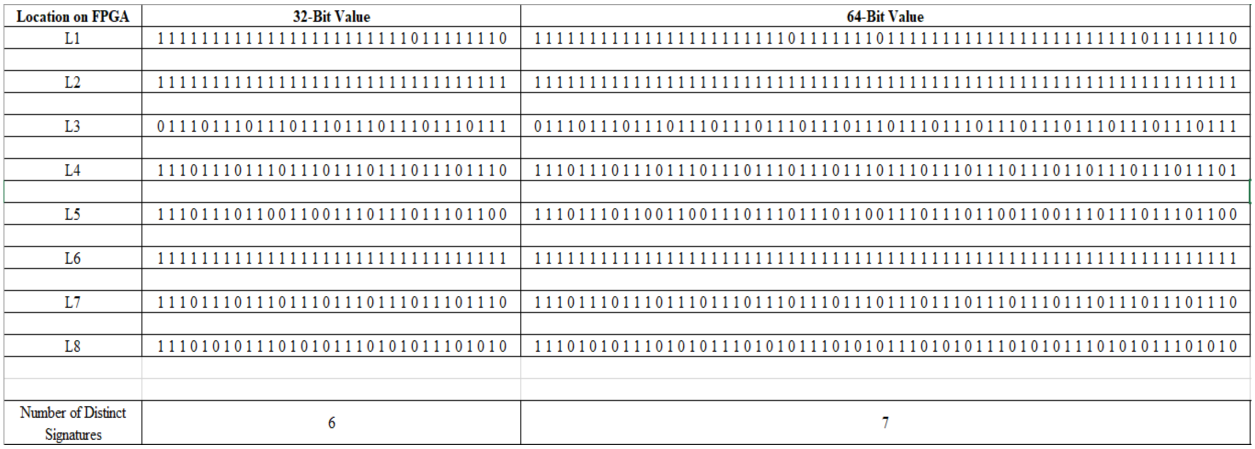
Location on FPGA	32-Bit Value	64-Bit Value
L1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	6	7

Figure A.12: Experimental Data for 32 and 64-bitstreams on FPGA Board-6.



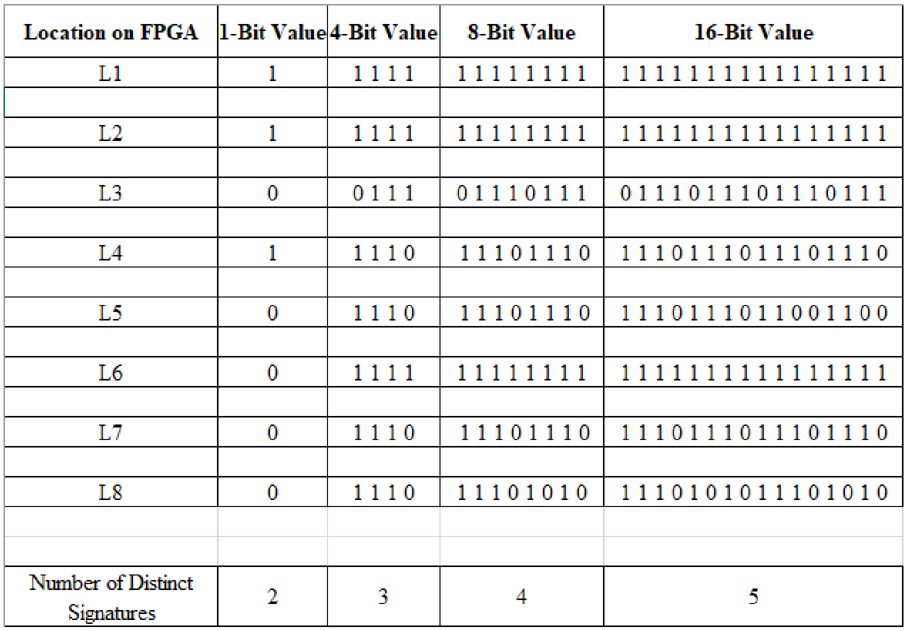
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1111	11111111	1111111111111111
L2	1	1111	11111111	1111111111111111
L3	0	0111	01110111	0111011101110111
L4	1	1110	11101110	1110111011101110
L5	0	1110	11101110	1110111011001100
L6	0	1111	11111111	1111111111111111
L7	0	1110	11101110	1110111011101110
L8	0	1110	11101010	1110101011101010
Number of Distinct Signatures	2	3	4	5

Figure A.13: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-7.



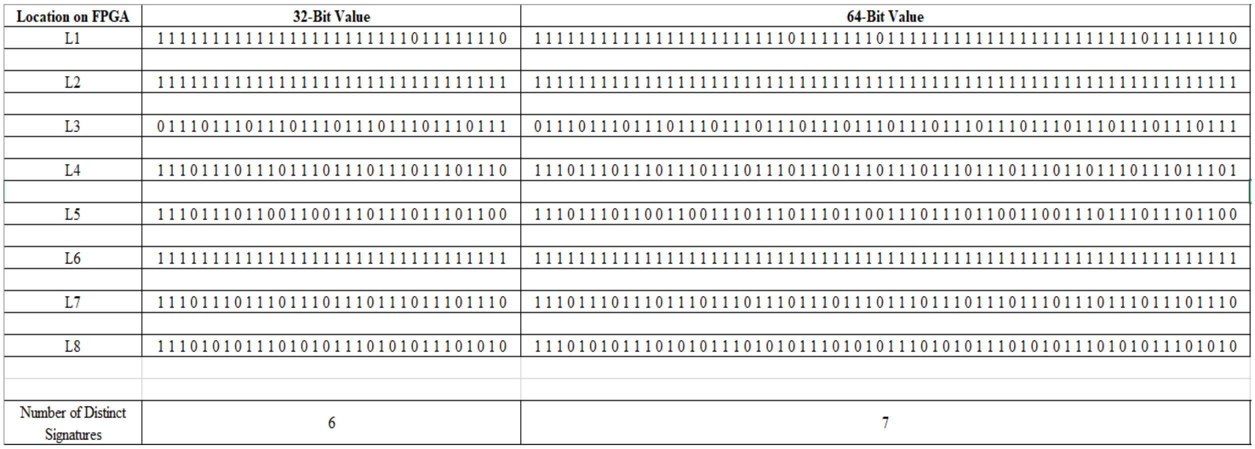
Location on FPGA	32-Bit Value	64-Bit Value
L1	11111111111111111111111011111110	1111111111111111111111101111111011111111111111111111111011111110
L2	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L3	01110111011101110111011101110111	0111011101110111011101110111011101110111011101110111011101110111
L4	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101101110111011101
L5	11101110110011001110111011101100	1110111011001100111011101110110011101110110011001110111011101100
L6	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L7	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101110111011101110
L8	11101010111010101110101011101010	1110101011101010111010101110101011101010111010101110101011101010
Number of Distinct Signatures	6	7

Figure A.14: Experimental Data for 32 and 64-bitstreams on FPGA Board-7.



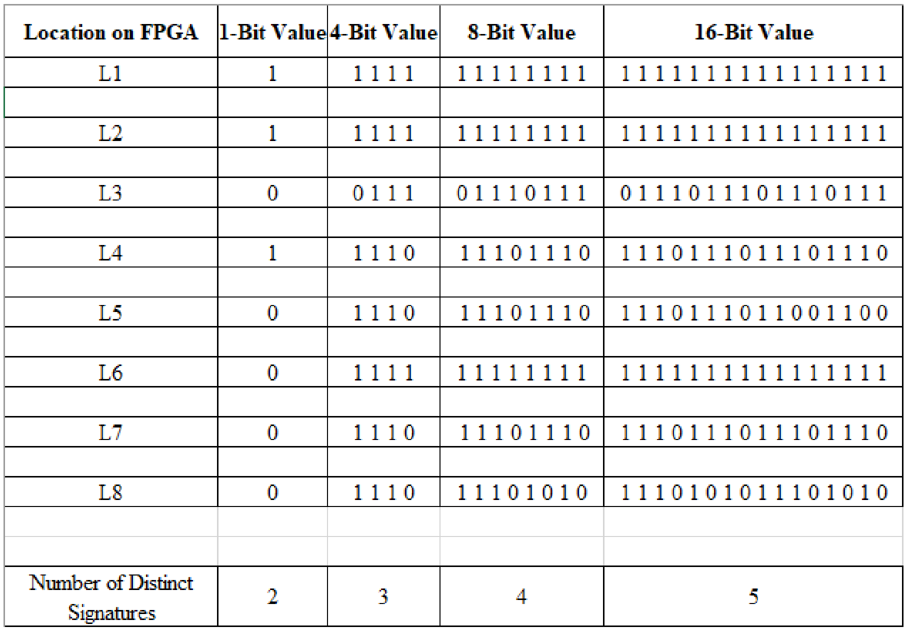
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1111	11111111	1111111111111111
L2	1	1111	11111111	1111111111111111
L3	0	0111	01110111	0111011101110111
L4	1	1110	11101110	1110111011101110
L5	0	1110	11101110	1110111011001100
L6	0	1111	11111111	1111111111111111
L7	0	1110	11101110	1110111011101110
L8	0	1110	11101010	1110101011101010
Number of Distinct Signatures	2	3	4	5

Figure A.15: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-8.



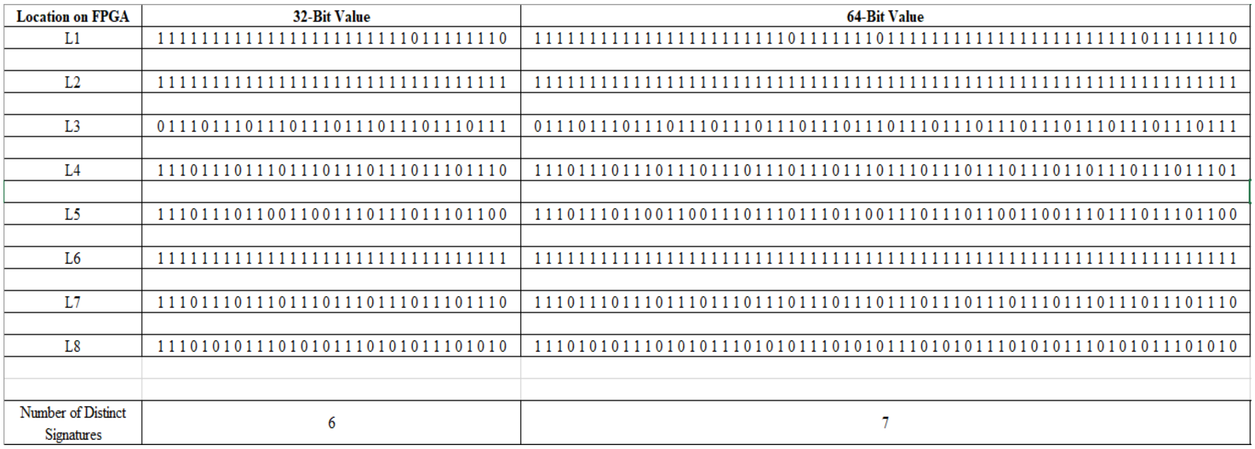
Location on FPGA	32-Bit Value	64-Bit Value
L1	11111111111111111111111011111110	1111111111111111111111101111111011111111111111111111111011111110
L2	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L3	01110111011101110111011101110111	0111011101110111011101110111011101110111011101110111011101110111
L4	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101101110111011101
L5	11101110110011001110111011101100	1110111011001100111011101110110011101110110011001110111011101100
L6	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L7	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101110111011101110
L8	11101010111010101110101011101010	1110101011101010111010101110101011101010111010101110101011101010
Number of Distinct Signatures	6	7

Figure A.16: Experimental Data for 32 and 64-bitstreams on FPGA Board-8.



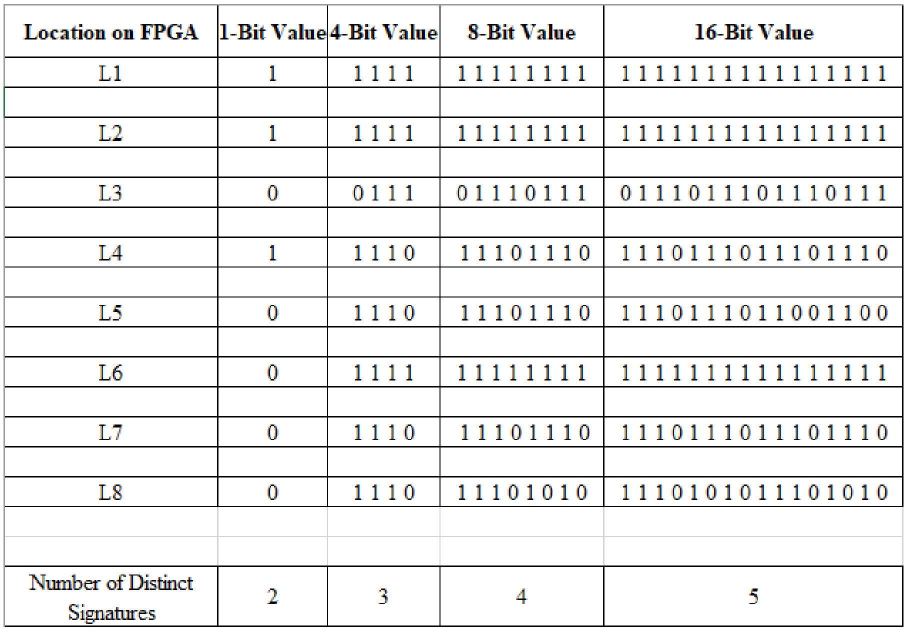
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	3	4	5

Figure A.17: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-9.



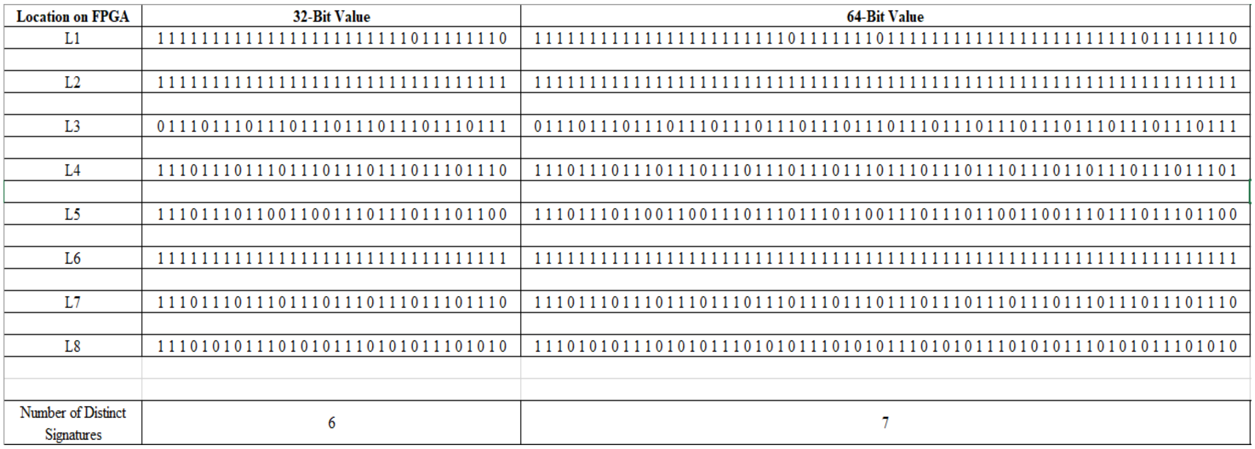
Location on FPGA	32-Bit Value	64-Bit Value
L1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	6	7

Figure A.18: Experimental Data for 32 and 64-bitstreams on FPGA Board-9.



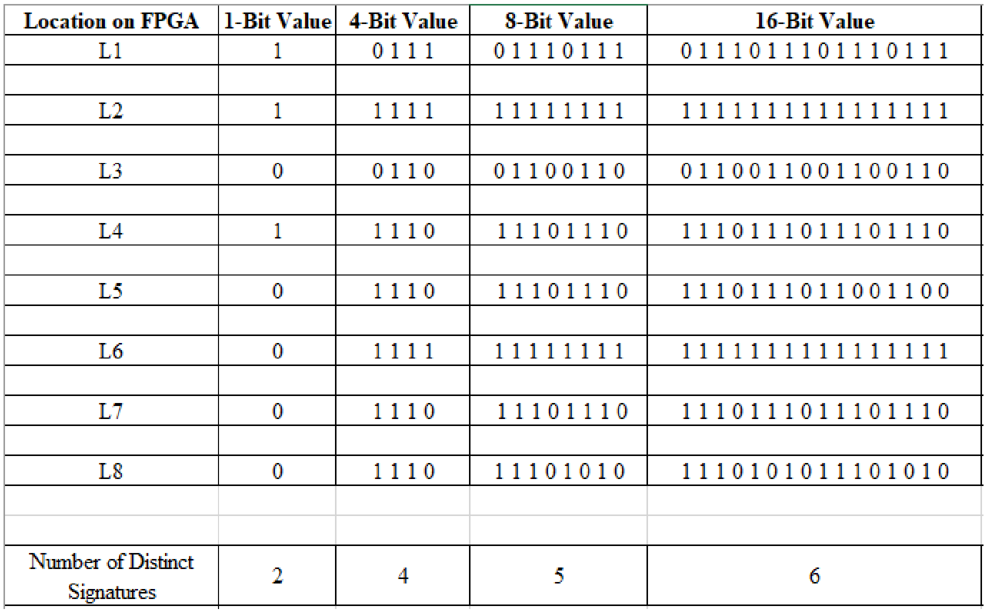
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	3	4	5

Figure A.19: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-10.



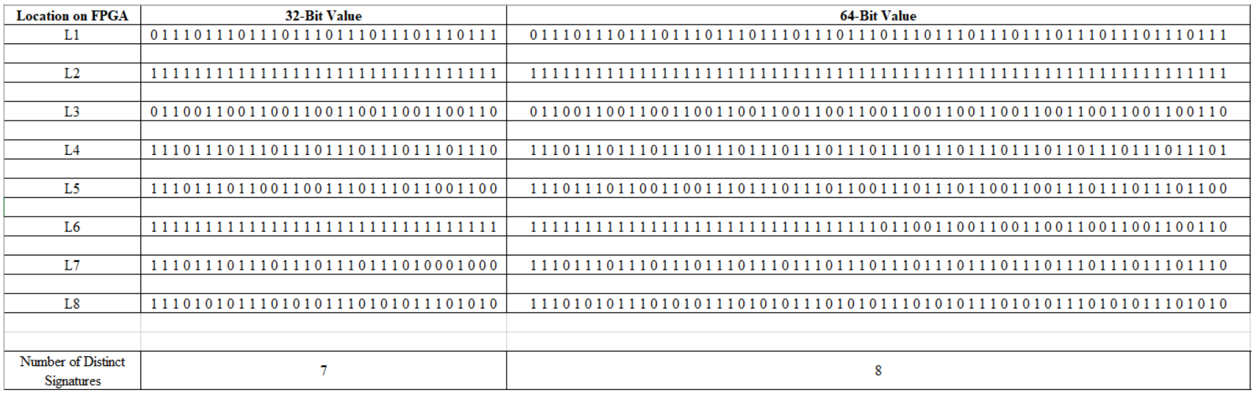
Location on FPGA	32-Bit Value	64-Bit Value
L1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	6	7

Figure A.20: Experimental Data for 32 and 64-bitstreams on FPGA Board-10.



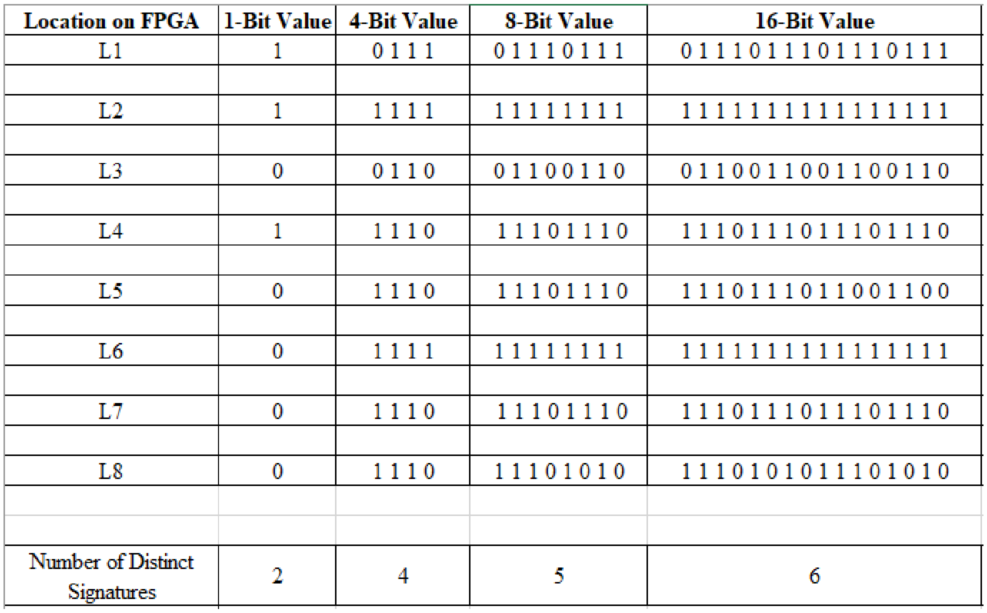
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 0	0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	4	5	6

Figure A.21: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-11.



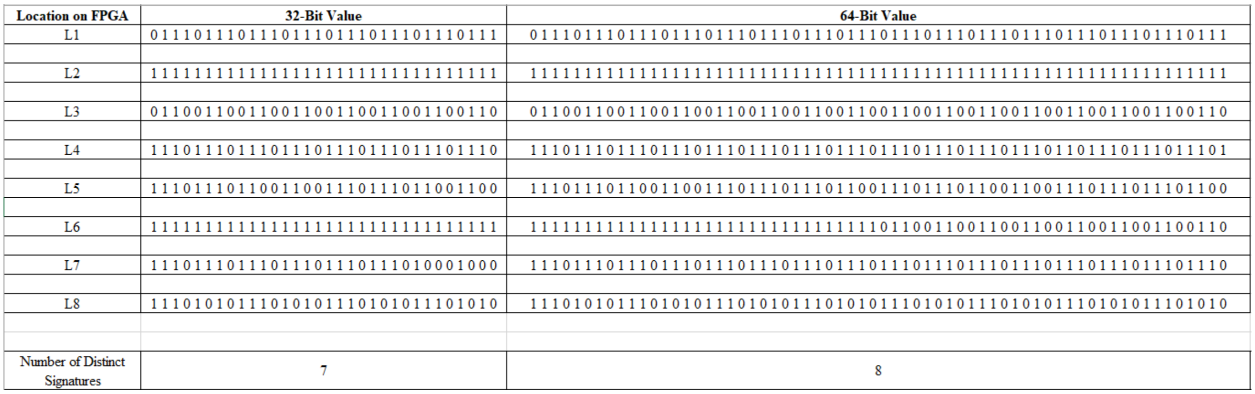
Location on FPGA	32-Bit Value	64-Bit Value
L1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	7	8

Figure A.22: Experimental Data for 32 and 64-bitstreams on FPGA Board-11.



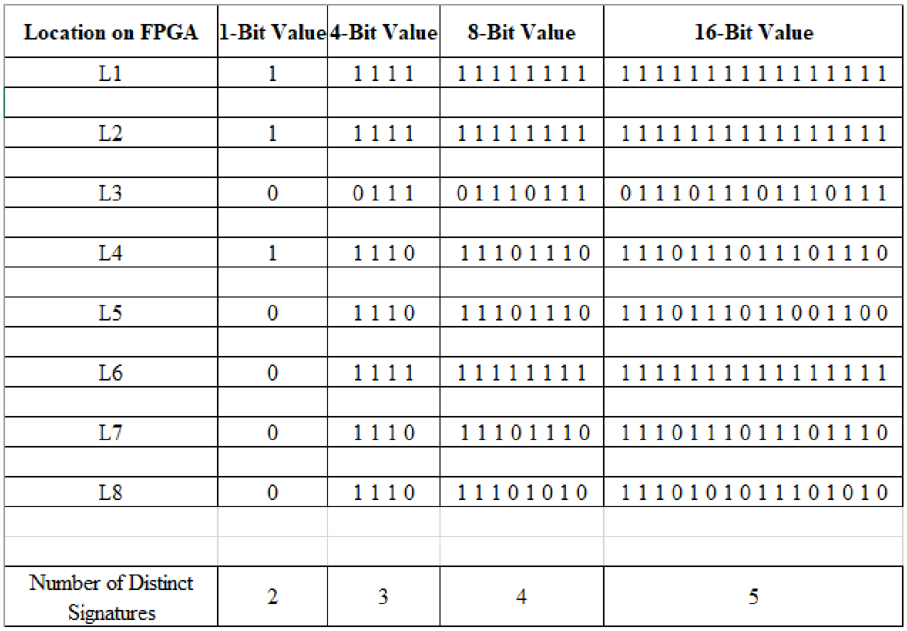
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 0	0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	4	5	6

Figure A.23: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-12.



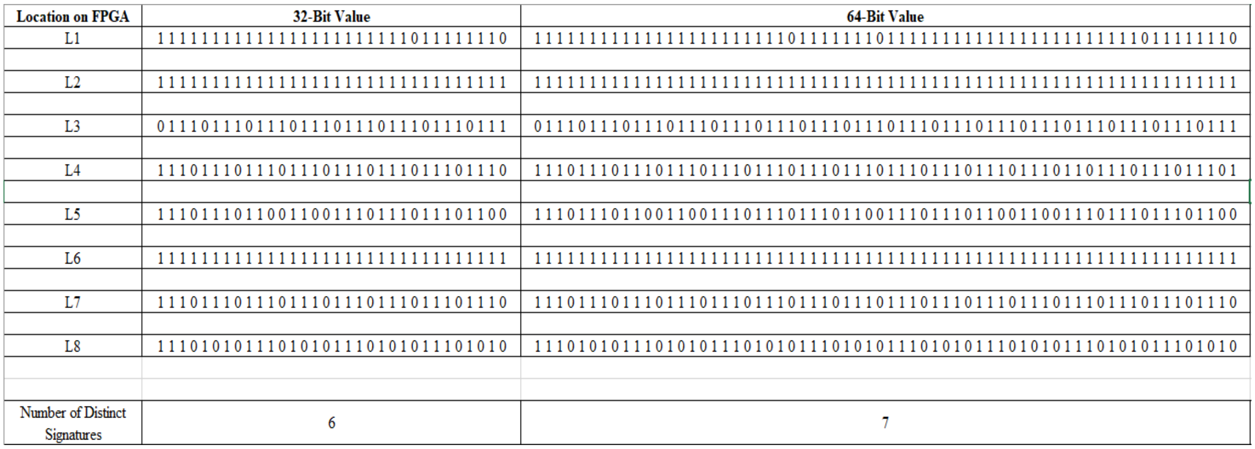
Location on FPGA	32-Bit Value	64-Bit Value
L1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	7	8

Figure A.24: Experimental Data for 32 and 64-bitstreams on FPGA Board-12.



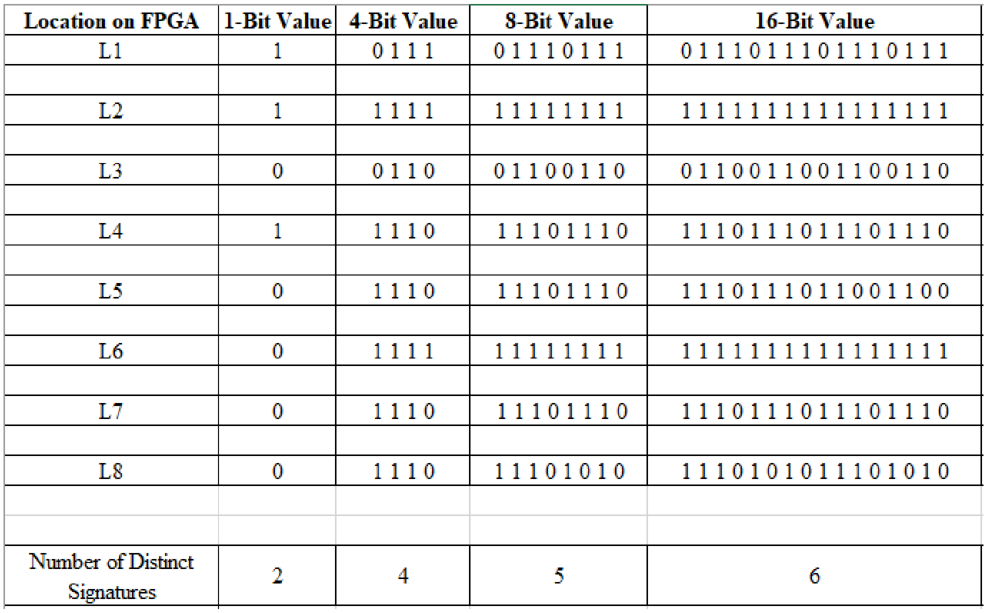
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1111	11111111	1111111111111111
L2	1	1111	11111111	1111111111111111
L3	0	0111	01110111	0111011101110111
L4	1	1110	11101110	1110111011101110
L5	0	1110	11101110	1110111011001100
L6	0	1111	11111111	1111111111111111
L7	0	1110	11101110	1110111011101110
L8	0	1110	11101010	1110101011101010
Number of Distinct Signatures	2	3	4	5

Figure A.25: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-13.



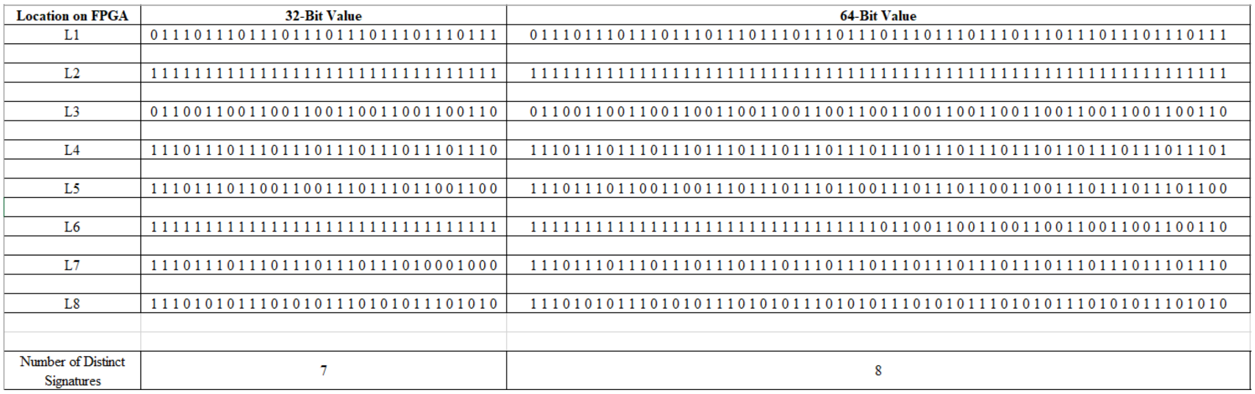
Location on FPGA	32-Bit Value	64-Bit Value
L1	11111111111111111111111101111110	1111111111111111111111110111111011111111111111111111111101111110
L2	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L3	01110111011101110111011101110111	0111011101110111011101110111011101110111011101110111011101110111
L4	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101101110111011101
L5	11101110110011001110111011101100	1110111011001100111011101110110011101110110011001110111011101100
L6	11111111111111111111111111111111	1111111111111111111111111111111111111111111111111111111111111111
L7	11101110111011101110111011101110	1110111011101110111011101110111011101110111011101110111011101110
L8	11101010111010101110101011101010	1110101011101010111010101110101011101010111010101110101011101010
Number of Distinct Signatures	6	7

Figure A.26: Experimental Data for 32 and 64-bitstreams on FPGA Board-13.



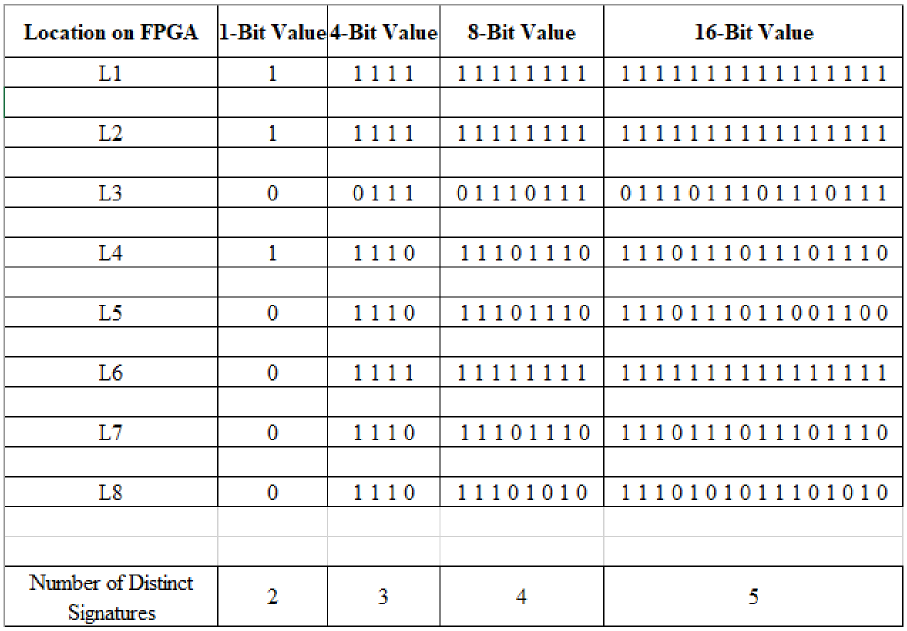
Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 0	0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	4	5	6

Figure A.27: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-14.



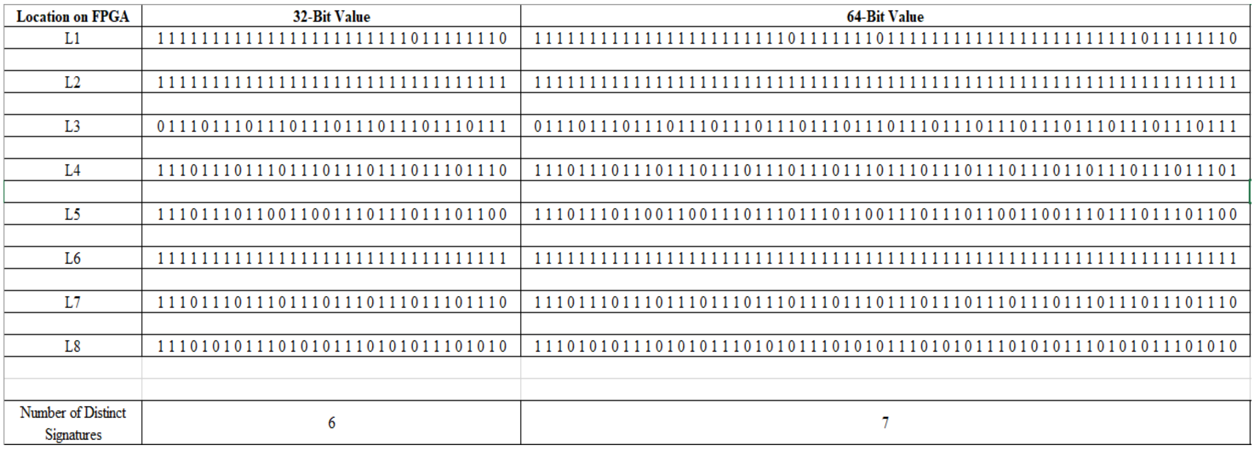
Location on FPGA	32-Bit Value	64-Bit Value
L1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	7	8

Figure A.28: Experimental Data for 32 and 64-bitstreams on FPGA Board-14.



Location on FPGA	1-Bit Value	4-Bit Value	8-Bit Value	16-Bit Value
L1	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L2	1	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0	0 1 1 1	0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L5	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0
L6	0	1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	0	1 1 1 0	1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	0	1 1 1 0	1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	2	3	4	5

Figure A.29: Experimental Data for 1, 4, 8, and 16-bitstreams on FPGA Board-15.



Location on FPGA	32-Bit Value	64-Bit Value
L1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0
L2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L3	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1	0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
L4	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1
L5	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0	1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0
L6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
L7	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0	1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0
L8	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0	1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0
Number of Distinct Signatures	6	7

Figure A.30: Experimental Data for 32 and 64-bitstreams on FPGA Board-15.