June 2018

SR Flip-Flop Based Physically Unclonable Function (PUF) for Hardware Security

Rohith Prasad Challa
University of South Florida, challa1@mail.usf.edu

SR Flip-Flop Based Physically Unclonable Function (PUF) for Hardware Security

by

Rohith Prasad Challa

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

Major Professor: Srinivas Katkoori, Ph.D.
Robert Karam, Ph.D.
Hao Zheng, Ph.D.

Date of Approval:
May 24, 2018

Keywords: Weak PUF, Strong PUF, Challenge Response Pairs, Race Condition, Process Variations, Inter Chip Variation, Intra Chip Variation

DEDICATION

I dedicate this work to my friends and family for their love and support.

ACKNOWLEDGMENTS

I begin with thanking 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 I love gave some meaning to my Master's degree. 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 constant love and support. I would also like to thank Shiekh Ariful Islam and Md. Adnan Zaman for their assistance throughout this work. I would like to thank Dr. Hao Zheng and Dr. Robert Karam for volunteering their precious time to serve as members on my thesis committee. I would like to thank my professors from Vardhaman College of Engineering, Dr. J. V. Ravindra and Dr. S. Rajendar for motivating me to choose this branch of study. I would like to thank Dr. Rekha Govindaraj for helping me to solve key points in my thesis. 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

Physically Unclonable Functions (PUFs) are now widely being used to uniquely identify Integrated Circuits (ICs). In this work, we propose a novel Set-Reset (SR) Flip-flop based PUF design. For a NAND gate based SR flip-flop, the input condition S (Set) = 1 and R (Reset) = 1 must be avoided as it is an inconsistent condition. When S=R=1 is applied followed by S=R=0, then the outputs Q and Q' undergo race condition and depending on the delays of the NAND gates in the feedback path, the output Q can settle at either 0 or 1. Because of process variations in an IC, the NAND delays are statistical in nature. Thus, for a given SR FF based n-bit register implemented in an IC, when we apply S=R=1 to all flip-flops followed by S=R=0, then we obtain an n bit string that can be interpreted as a signature of the chip. Due to process variations, the signature is highly likely to be unique for an IC. We validated the proposed idea by SPICE-level simulations for 90nm, 45nm, and 32nm designs for both intra- and inter-chip variations to establish the robustness of the proposed PUF. Experimental results for 16-, 32-, 64-, and 128-bit registers based on Monte-Carlo simulations demonstrate that the proposed PUF is robust. The main advantage of the proposed PUF is that there is very little area overhead as we can reuse existing registers in the design.

CHAPTER 1: INTRODUCTION

Many integrated circuits (ICs) require a unique identification key or ID on their die that can be read throughout the lifetime of the die. For example, mobile phones and embedded devices are becoming universal platforms for everyday tasks. Those tasks require the mobile devices to securely authenticate so that the private information is not compromised. Cryptography offers several measures for these problems but it is difficult to uphold in practice. The solution for this problem lies in the building and developing of a secure hardware which does not give access to the invaders to breach the device. Changes in the memory can serve to tighten the security of the hardware which is the current best practice for providing a secure memory or authentication source. The practice is to place a secret key in a nonvolatile electrically erasable programmable read only memory (EEPROM) or battery backed static random access memory (SRAM) and use hardware cryptographic operations such as digital signature or encryption. Such security protection mechanism requires external connectivity and power source. Therefore, this is an expensive process in terms of the design area and power consumption.

The new and promising method to overcome this problem has to be a function which takes certain challenges and produces responses based on those challenges. Physically Unclonable Functions (PUFs) are new and promising method for generating unique, reliable, and secure key for hardware authentication. These functions generate the key without the requirements of secure EEPROM and other expensive hardwares described above. The methods till now derive secret key

from the digital memory but PUFs derive secret key from the physical characteristics of the integrated circuit. This paper discusses the secret key generation from the SR flip flop by using the gate delays and generates a new unique key for each IC. A PUF is a piece of hardware which acts as a black box and transforms input to the output in a particular way. The connections inside the black box are difficult to reverse engineer.

PUFs have advantages over the current principle of generating secret keys from digital storage because they generate keys based on the physical characteristics of the IC. These primitives comprise of simple digital circuits which are easy to fabricate and they consume less power and less area on the IC. They do not require cryptographic algorithms or encryption algorithms to generate and decode a key. EEPROM needs additional circuitry for the protection and it has to be continually powered ON which is expensive to manufacture. Since PUF generates key from the physical characteristics the chip must be powered ON for the secret to reside in the digital memory. If the same design is manufactured on different ICs there will be variations in the chips because of the manufacturing processes, and this forms the basis for a PUF. To derive the secret keys PUFs use the concepts of gate delays, process variations, powered ON state of the SRAM, threshold voltages, etc.

In this work, we propose an flip flop based PUF which uses SR flip flops that are present in the memory and using process variations and by manipulating the threshold voltages gives unique ID to different chips. The proposed PUF uses the concept of race condition of the SR flip flop and using gate delays to generate a secret key which is unique. To randomize the race condition of the SR flip flop we have an additional circuitry which is described in later chapters.

The rest of the thesis is organized as follows. Chapter 2 reviews the various PUFs reported in literature and introduces PUF related terminology (weak and strong PUFs). Chapter 3 describes

in detail the proposed SR flip flop based PUF i.e., the circuitry, design approach, race condition and the proposed model of SR flip flop based PUF with the detection circuit for the random flip flops. Chapter 4 presents the experimental results in detail. Finally, Chapter 5 draws conclusion.

CHAPTER 2: BACKGROUND AND RELATED WORK

In this Chapter, we will introduce PUF related terminology and review various PUFs proposed in the literature.

2.1 PUF Related Terminology

A Physically Unclonable Function (PUF) is a *digital fingerprint* that serves as a unique identity for integrated circuits (ICs). A PUF is a piece of hardware which acts as a black box and transforms input to the output in a particular way, the connections inside the black box are difficult to reverse engineer. The inputs to the PUF are called as Challenge Response Pairs (CRPs). PUFs accept different CRPs and generate different output responses for each challenge. The classification of PUFs is based on the number of challenge response pairs. For a die, the key produced by a PUF should be unique compared to key of another die on same wafer. Inter PUF variation gives us the uniqueness of a PUF from which we can conclude that the key produced for a die is different from other keys. The key produced should be reliable and it should not change for multiple iterations on the same die. Intra PUF variation gives the information about a PUF's reliability. For a PUF, the effects of both inter-die and intra-die variations has to be performed and the uniqueness and reliability of the PUF should be determined.

Based on the number of challenges a PUF can handle, PUFs are categorized into two types. They are: i) weak PUF; and ii) strong PUF. A weak PUF can handle only a small number of

challenge response pairs (CRPs) whereas a strong PUF can handle more number of CRPs. PUFs are used in hardware security for two major applications i.e., low cost authentication and secure key generation. A weak PUF has less number of challenges and is used for key storage whereas a strong PUF is used for authentication.

The working of a PUF can be modeled as a black box where we know the input and output of the system but the internal details are hidden. In other words, if we pass an input challenge, c, a PUF returns the response $r = f(c)$ as shown in Figure 2.1. We have the output response but the internals are hidden from the user since they represent the internal manufacturing variability that the PUF uses to generate a unique challenge response set. The difference between a weak PUF and strong PUF lies in the number of unique challenges a PUF can process.

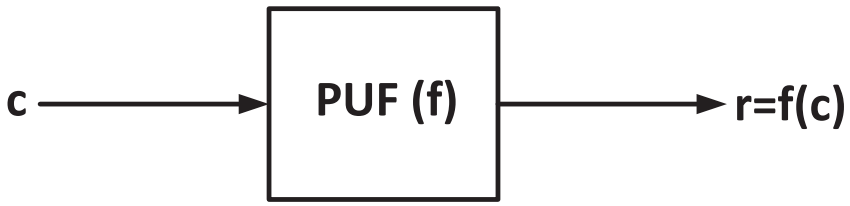

Figure 2.1: Working of a PUF

2.2 Weak PUF Architectures

Weak PUFs form an alternative method for storing secret keys in ROM, Flash, and other nonvolatile memories. Compared to strong PUFs, weak PUFs also have physical disorders which enable us to create a challenge-response mechanism which exploits this physical disorder. The characteristic features of a weak PUF are: i) it has only a few challenges i.e., for every PUF we have single challenge or some fixed number of challenges which exploit the physical disorder; ii) Weak

PUF has an access restricted pair of responses by which the adversary cannot access the PUF even with the physical possession of the device. The above characteristics reduce the scope for the use of weak PUFs. [1, 2]. SRAM PUF and Ring Oscillator PUF are two examples of weak PUFs. Below we discuss these PUFs.

2.2.1 Memory Based PUF: SRAM PUF

All weak PUFs depend on process variations and variation in propagation delay of the gates. The classic implementation of weak PUF is an SRAM PUF which has two stable states i.e., logic 0 and logic 1. The physical disorder in the SRAM cell which enables us to exploit and generate a unique key is the power ON state of SRAM cell. This is because of the positive feedback loop in the SRAM cell. The structure of SRAM cell is as shown in Figure 2.2 where it has two positive feedback loops which force the cell into one of the two states (logic 0 or logic 1). After attaining a stable state this loop prevents the cell from transitioning out of that state accidentally. In an SRAM cell for the transition toward one of the states, a write operation has to be enabled. The physical disorder of the SRAM occurs when the power is turned ON without enabling the write operation. The SRAM cell enters the metastable state where the feedback pushing the cell towards '1' state equals the feedback pushing the cell towards the '0' state. This condition locks the cell in this metastable state indefinitely. To overcome this one of the feedback should be slightly stronger than the other so that the stronger feedback dominates the weaker feedback and transitions to either logic 1 or logic 0 [1, 2]. This is naturally done due to the process variations while manufacturing the chip [3, 4, 5, 6].

This transition is independent of the external factors such as die temperature, power supply fluctuations, and common mode process variations. The measurement is differential because of the

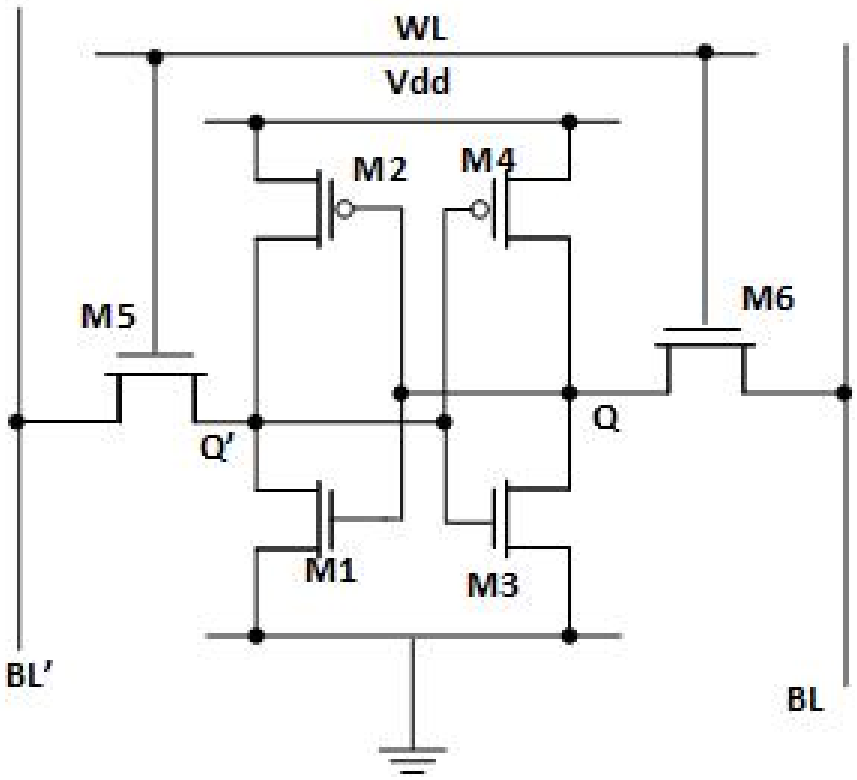

Figure 2.2: SRAM Cell

final state of the SRAM cell and depends on the delay difference between the two feedback loops.

As mentioned above, SRAM PUF leverages the threshold voltage of transistors in a SRAM cell due to manufacturing mismatch. When system is powered ON and there are no writes occurring, this mismatch results in a tendency to settle at '1' or '0'. SRAM PUF and their properties are analyzed using several studies.

The first such implementation was tested with RFID applications [7]. The authors use a custom SRAM cell which minimizes the system mismatch between two transistors. This skew would result in a given SRAM cell more likely to favor a 1 than a 0 or vice versa, this would result the same even with process variations. Using analog layout techniques the authors construct a symmetric and common centroid layout of an SRAM cell to prevent the skew. This study demonstrated that

after fabrication, within experimental errors for both layouts, equal number of SRAM cells tended towards a 1 or 0. Using this, the positioning of the SRAM cells on the wafer were de-correlated with the cells tendency to settle at 1 or 0. There were nearly 4% of SRAM cells where the mismatch was not significant and the cells did not favor 1 or 0 strongly. Due to thermal and shot noise, these cells settled at a random value. The number of such random cells will increase at temperature/voltage corners and as the chip ages.

Another study tested the functionality of SRAM PUFs on off-the-shelf RAM processor products [8]. In this study, the SRAM cell was part of another on-chip SRAM cell that was actively used for program and data transfers. This enables an end user to use existing off-the-shelf component and implement a weak PUF only using software. From this study, it can be seen that the off-the-shelf SRAM cell favors a 1 state, changing the entropy and unique identification analysis. The problem with this implementation is that SRAM should always be powered ON to generate a 1 or 0 even though when there is no write operation being performed. This results in Negative Bias Temperature Instability (NBTI) by which the threshold voltage of a transistor increases over time due to applied stress conditions of high temperatures across the gate terminal while the transistor is ON.

SRAM PUF relies on conventional security primitives to keep the key secure when the power is ON. Side channel attacks and other vulnerabilities pose a threat to the secret key output by the SRAM PUF. In weak PUFs, the key is kept secret and is not revealed. Therefore, modeling attacks used for strong PUFs which will have input/output relations are not suitable for weak PUFs.

2.2.2 Ring Oscillator PUF

A ring oscillator PUF is based on delay loops (ring oscillators) and counters as shown in Figure 2.3. The architecture comprises of N identical laid-out delay loops known as ring oscillators. Each ring oscillator is a simple circuit which oscillates with a particular frequency. Due to manufacturing variations, the delay associated with the inverters in the ring oscillators varies, and thus each ring oscillator oscillates with a slightly different frequency. In the output bit string if we want to have a fixed number of bits, a fixed sequence of oscillator pairs are selected, and their frequencies are compared to generate the output bit. For N ring oscillators there will be $N(N\text{-}1)/2$ distinct pairs and if oscillators are identical copies with the significant manufacturing variability, then we get $N!$ pairings.

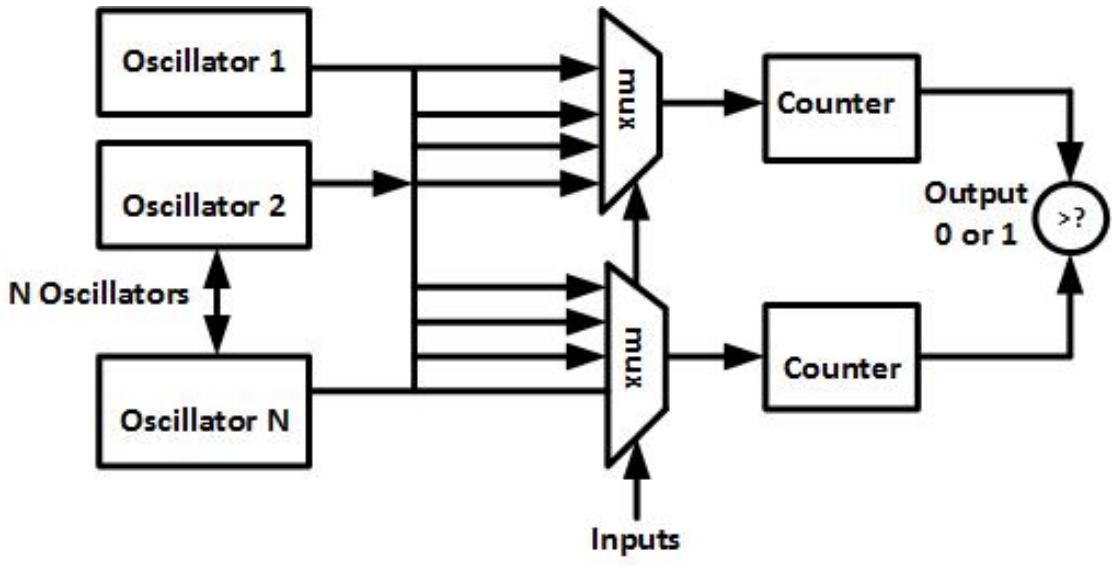

Figure 2.3: Ring Oscillator PUF

The maximum number of bits generated from the PUF will be $log(N!)$. For example, 35 oscillators can produce 133 bits, 128 oscillators can produce 716 bits, and 1024 oscillators can produce 8769 bits The ring oscillator PUF is considered to be a weak PUF because it has a single

challenge i.e., when the chip is fabricated and if frequency is set then the output bits remain constant and we get a unique key for different chips based on their oscillating frequencies. Error correction is important for ring oscillator PUF because the environmental factors such as temperature and noise effect the key generation. Another example of the weak PUF is lightweight secure PUF discussed in [9].

2.3 Strong PUF Architectures

Strong PUFs derive a more complex challenge response behavior from the physical disorder present in the device. To generate a response, unlike weak PUFs, strong PUFs have many physical components and the number of challenges applied to a PUF is very large [1, 2]. Even the physical possession of the PUF by the adversary does not allow the adversary to read out all the challenge response pairs which make it difficult to exploit.

2.3.1 Optical PUF

A *physical one-way function* is a strong PUF implementation proposed by Pappu *et. al.* in 2001 [10]. He described a device which consists of an optical scattering medium, a laser directed onto the XY plane from Z axis and an imaging device which measures the output speckle pattern of laser light exiting the optical scattering object. Inside a scattering medium, multiple scattering events occur. So, the response i.e., the speckle pattern strongly depends on the input location. Here, in this device, the input location is in the XY plane and the response is given by the speckle which is detected by the imaging device. The laser beam is directed at a selected angle and point of incidence onto the XY plane. The resulting response emerges from the complex light scattering process which is a multi-bit value.

The scattering object consists of a large number of silica spheres which act as small lens and refract the rays of the laser beam through the scattering block. It is estimated that this implementation results in 1010 different independent challenge-response pairs. This is secure because the output speckle pattern is dependent on the internal structure of the scattering object. So, we cannot manufacture two different objects with the same scattering medium. It is difficult to manufacture a similar device with same scattering block, discovering speckle patterns by the adversary. If they tend to do so then we can manufacture more stronger PUFs structures based on this electromagnetic simulation. The practice of this kind of PUFs is limited because of the macroscopic optical nature. This led to the implementation of another strong PUF called the arbiter PUF.

2.3.2 Arbiter PUF

An arbiter PUF consists of delay paths and an arbiter at the end of the delay path. The arbiter can be a latch, an XOR gate, or a basic flip-flop. In the arbiter based PUF we excite two delay paths simultaneously and make the transitions race against each other. The arbiter is used to decide which rising edge has arrived first and sets its output to 1 or 0 depending on the winner. The delay paths are generated by a switching circuit such as a multiplexer which determines the path based on the select lines given. An arbiter PUF with multiplexers generating the delay and the D latch acting as an arbiter is shown in Figure 2.4 [1, 2]. The output will thus depend on the challenge bits. We can scale the number of bits to an arbitrary value. For the security of an arbiter PUF, we depend on the assumptions regarding manufacturing capabilities of individual gates. The design of an arbiter PUF is based on the inherent variability in the manufacturing process. So, an adversary cannot create a duplicate PUF with similar variations. The gate delays which are present in the design cannot be decoded as is difficult to measure the individual gate delays. Each

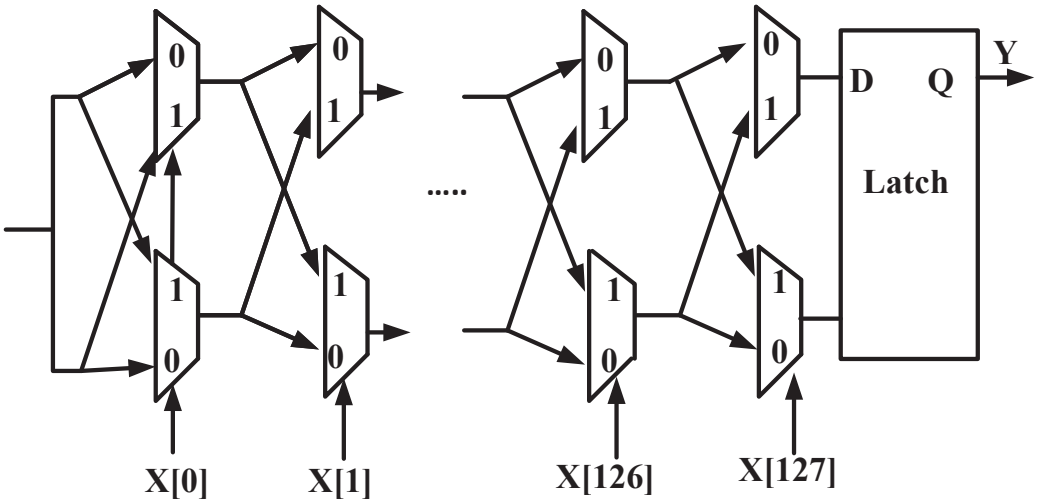

Figure 2.4: Arbiter PUF

delay is independent of the other delays, and the delays add linearly. Environmental factors such as temperature, supply voltage, aging, and random noise will affect the delay associated with each edge through the arbiter PUF.

The arbiter PUF was implemented and studied as a part of radio frequency identification IC fabricated in 0.18 μm technology [11]. This study has a 64 bit input challenge and one output challenge with a single on chip arbiter PUF. A linear feedback shift register is used to generate a pseudo random sequence based on the input challenge and constructs a k-bit challenge. The PUF is then evaluated n times using different input challenge for every iteration. An additional scrambling routine is performed to prevent learning attacks on the PUF output bits. This study uses only a single arbiter PUF because area and power consumption represent a major design constraint. This results in the majority of the chip area with standard RFID components. The advantage with this approach is that the PUF consumes only dynamic power and is small compared to that of RFID components.

Intra PUF variation is defined as the number of bits that vary when presented with identical challenge under different conditions (such as operating temperature). *Inter PUF variation* is defined as the number of bits in a PUF response that vary between devices for a set of shared challenges. Inter chip variation has to be ideally 50% and intra chip variation should be ideally 0%. This study shows that the inter chip variations are roughly 50% and intra chip variations are close to 10% which shows that the implementation has met the desired properties.

Security of an arbiter PUF depends on the difficulty of measuring the internal parameters of the PUF and difficulty of predicting PUFs behavior with respect to previous challenge response pairs. The difficulty in measuring the PUF behavior based on past CRPs is even more difficult for arbiter PUF. In RFID IC example, it is easier to predict the PUF because the output bits of a PUF pass through a digital circuit that obfuscates linear behavior of the PUF.

Few studies used machine learning tools to predict past behavior of PUFs after a considerable number of CRPs are observed. For a simple arbiter PUF, after observing 640 CRPs, there was a prediction accuracy of 95% [12]. To get 99.9% accuracy, 18,050 CRPs are to be observed. Arbiter PUF with input XORing can make the problem intractable to such learning attacks.

2.4 Some More PUF Architectures

2.4.1 Butterfly PUF

A Butterfly PUF consists of structures which behave like SRAM cell during the startup phase. A BPUF is a cross coupled circuit which can be brought to a floating state before allowing to settle in one of the two stable states [13]. Latches present in the FPGA are used to simulate a cross coupled combinational logic. Figure 2.5 shows the structure of a Butterfly PUF.

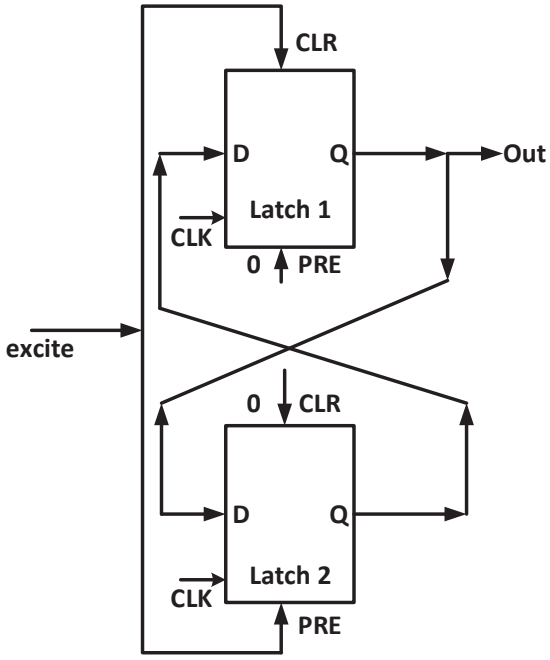

Figure 2.5: Butterfly PUF

BPUF consists of two latches, each with a preset signal and a clear signal. The PRE of Latch 1 and the CLK of Latch 2 are always set to low and the data D is transferred to the output Q when the CLK is high. CLR of Latch 1 and PRE of Latch 2 are connected and excite signal is passed onto that connection. From Figure 2.5 we can see that the outputs of the latch are cross coupled. The combinational loop is simulated by setting both CLK of both latches to always high. To start the PUF operation and to bring the BPUF circuit to unstable operating point, enable the excite signal to high. To attain one of the two possible states at the output i.e., 1 and 0 after few clock cycles the excite signal has to be set to low. The delays of the connecting wires which are designed using symmetrical paths defines the stable state which depends on slight differences in the delay between these wires. These delays depends on intrinsic properties of the IC and they differ from device to device and the positions on the FPGA.

2.4.2 Glitch PUF

The basic idea of a glitch PUF is to generate a sequence based on the glitches present between the inputs and outputs. This is explained in [14] with the help of Figure 2.6.

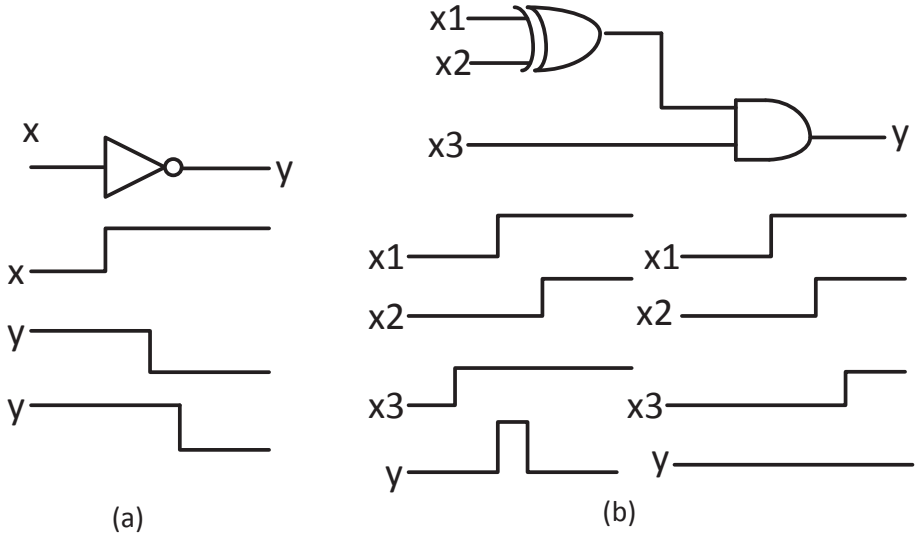

Figure 2.6: Glitch PUF

Figure 2.6 (a) shows a basic inverter in which the time difference between the output changes from an input change. It is difficult to exploit the time difference from an input change to output change because it depends on variation of gate delays, operating temperature, and voltage. In Figure 2.6 (b), we have a general circuit that perform XOR and AND to multiple inputs. Due to the delay differences in the input signals, a transient state of an output signal called a glitch is occurred. There will be a convex glitch at the XOR output if the input signals x1, x2 and x3 change from 0 to 1 from the difference of transition time between x1 and x2. If the input signal x3 reaches AND gate faster than the glitch, then the glitch passes the AND output. From [14], a glitch PUF consists of three steps: data input to a random logic, acquisition of glitch waveforms at the output

and conversion of waveforms into response bits. In Figure 2.6 (b), we have to change the inputs x1, x2 and x3 and apply to a random logic. The accompanying glitch waveform at the output y is acquired as n-bit data. By changing the inputs and iterating the steps, a bit sequence R is acquired at the end of last step.

2.4.3 Mecca PUF

Mecca PUF is a memory-cell based PUF which performs authentication by exploiting the intrinsic process variations in read/write reliability of cells in static memories. This PUF consists of a programmable delay generator and an SRAM array with peripherals. Programmable delay generator adds little overhead and most of the systems have a memory cell which we can use for this PUF construction. In the core array, inter- and intra-die variations in the device parameters cause a mismatch in the strengths of transistors which can be exploited to cause failures in cells [15].

In a memory cell, different failure mechanisms such as write-, read-, access-, and hold-failures occur. To avoid such failures, we can induce read/write collisions or using metastability in cross coupled loop to generate responses. In this PUF, by changing the word line duration, a SRAM cell is induced by a write failure. If an SRAM cell (Figure 2.7) is read and a value of "0" is observed, to change that value to "1" we can set BL to "1" and BL' to "0". The word line duration is purposely changed because at reduced length the cell may or may not be stable. For each WL durations, by selecting a set of R cells, we can obtain a signature consisting of both good cells and defective cells [15].

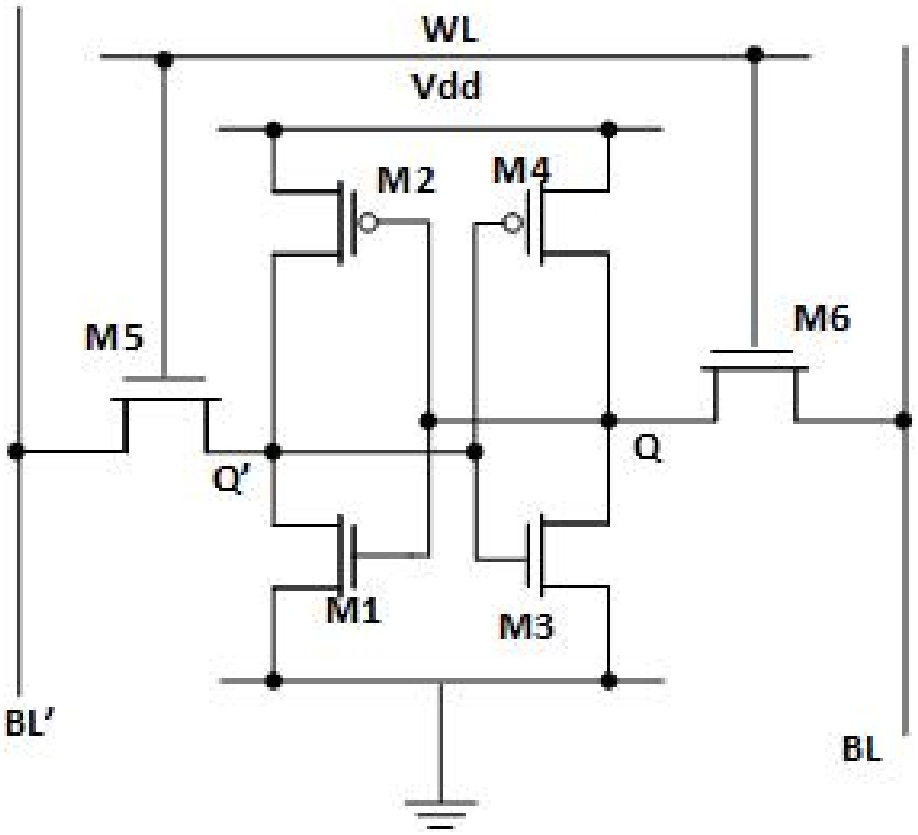

Figure 2.7: SRAM Cell (Figure 2.2 Reproduced for the Sake of Reader's Convenience).

2.5 Chapter Summary

Arbiter PUF is a good candidate for a strong PUF but the incurs significant area and power overheads due to the multiplexers and arbiters. Ring oscillator PUF accepts single challenge and it has the same disadvantage as arbiter PUF because large number of ring oscillators are required. Optical PUF generates highly unique key but is limited because of implementation complexity. SRAM PUF is a weak PUF and is effective when compared to other PUFs but its implementation is costly. The key idea of the proposed PUF is to use existing components in a design and reduce the overhead which in turn reduces silicon area and power.

CHAPTER 3: PROPOSED SR FLIP FLOP PUF

In this Chapter, we present in detail the proposed PUF design that uses an SR based flip-flop which has two logic states. SR flip-flop has an inconsistent state when both the set and reset inputs are 1 where the output is undetermined [16, 17, 18]. This state of the SR flip-flop can be changed by using the concept of gate delays. The proposed PUF leverages the race condition of the SR flip-flop and generates a unique key for every chip.

3.1 SR Flip Flop Circuit

3.1.1 Logic Level Implementation

SR flip-flop consists of cross-coupled NAND gates and has two additional NAND gates which are used to control the state of the flip flop. There are three inputs for an SR flip-flop i.e., input S and clock are given to the first NAND gate and input R and clock are given to the second NAND gate whose output is given to the cross-coupled NAND gates. The advantage of using a clock is that the output of this flip-flop can now be synchronized with other devices which share the same clock. The circuit and truth table of an SR flip-flop is shown in Figure 3.1 and Table 3.1 respectively in which the initial values of Q and Q' are set to 0. If S=0 and R=1 then the flip-flop is said to be in the reset state and the output of the flip-flop is Q=0 and Q'=1. The race condition occurs when the inputs are triggered to S=R=1 followed by S=R=0 where the state of the output is inconsistent (i.e., both Q and Q' are equal to 0).

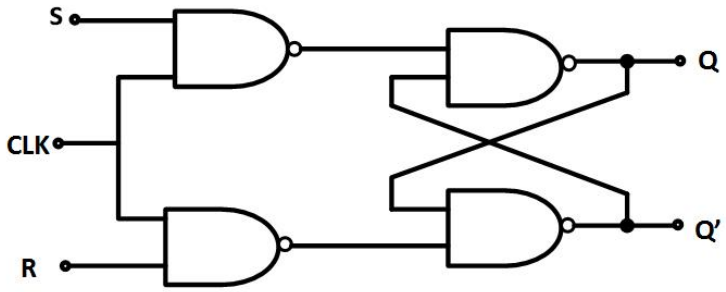

Figure 3.1: SR Flip Flop

Table 3.1: SR Flip Flop Truth Table

CLK	S	R	Q	Q'	State
1	0	0	Q	Q'	Hold (no change)
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	1	1	Inconsistent

3.1.2 Transistor Level Implementation

There are 4 NAND gates in an SR flip-flop, each NAND gate has four transistors. Figure 3.2 shows the transistor level diagram of the SR flip-flop which consists of sixteen gates (M0-M15). The first 8 transistors are used for clocking and giving inputs S (M0-M3) and R (M4-M7). The remaining gates are the cross-coupled NAND gates with the same combination as of the first two NAND gates (M8-M11) and (M12-M15). In the implementation, each transistor has four terminals i.e., gate, source, drain, and body. The inputs are always given to the gate terminal of the transistor and the global sources are given to source and drain. The fourth terminal body is connected to Vdd if it is a PMOS transistor and to GND if it is an NMOS transistor.

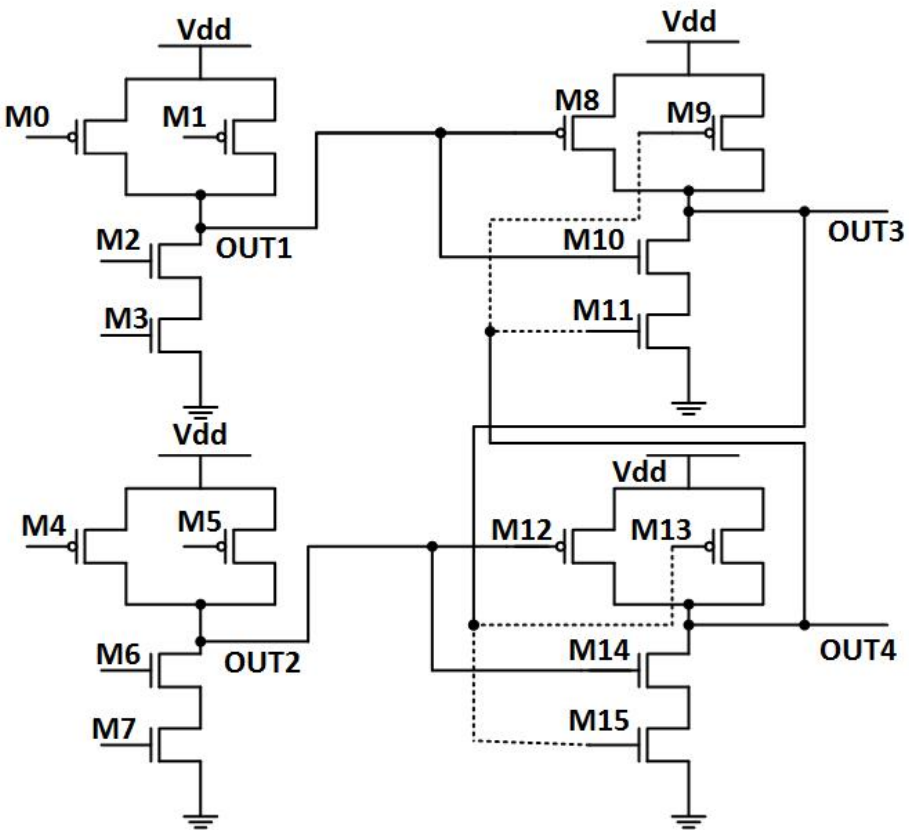

Figure 3.2: SR Flip Flop-Transistor Level

3.2 Race Condition

A timing related occurrence, race condition, is common to SR FF. When the inputs S and R are 1, followed by S and R are 0, the output of SR FF is said to be in inconsistent state [16, 17]. To reduce the possibility of race condition, we can vary transistor widths/lengths of one of the output NAND gates to increase the delay in that gate. The other NAND gate can then win the race and determine the output. From the transistor level diagram, if we want to propagate the inputs from NAND4 quickly, the transistors (M12-M15) can be sized to reduce the delay. Similarly, transistors (M8-M11) can be sized such that NAND3 wins.

3.3 PUF Construction

In this work, we use SR flip-flops already present in a design to create a unique ID for the device. We incorporate process variations (length and temperature) in Monte-Carlo simulation to determine the inter-die and intra-die performance for the proposed PUF. Inter-die process variations give the uniqueness of the PUF, ideally, the uniqueness of the PUF has to be 50% whereas intra-die can give the PUF's reliability in terms of multiple responses and ideally it has to be 0 [19]. All the previous approaches use additional circuitry to construct a PUF which in this case is omitted as we are using built-in registers to construct the PUF.

We can construct an SR Flip Flop based PUF by forming an n bit register using a scan chain mode as shown in Figure 3.3. In order to obtain the signature, we apply a pulse to the register. A built-in pulse generator is used to apply logic 1 to both inputs S and R followed by logic 0 (the falling edge of the pulse). Each flip-flop generates a single bit of key. If we have n register bits in the design, then we can obtain a signature with maximum size of n. All or some of the flip flops in a design can be used in the construction of this PUF.

In a traditional Glushkovian model, an RTL design consists of interacting datapath and controller blocks. We can automate the signature extraction process by incorporating an extraction state in the controller FSM as shown in Figure 3.4. After design reset, the controller goes into ID Extraction phase during which a pulse is applied to the PUF register and then the signature can be serially read out.

The proposed PUF, according to the theory, can be considered as a weak PUF but, the key generated by this PUF is based on the number of registers and hence, the probability of the uniqueness of the key increases with the number of flip flops. Table 3.2 shows the number of registers

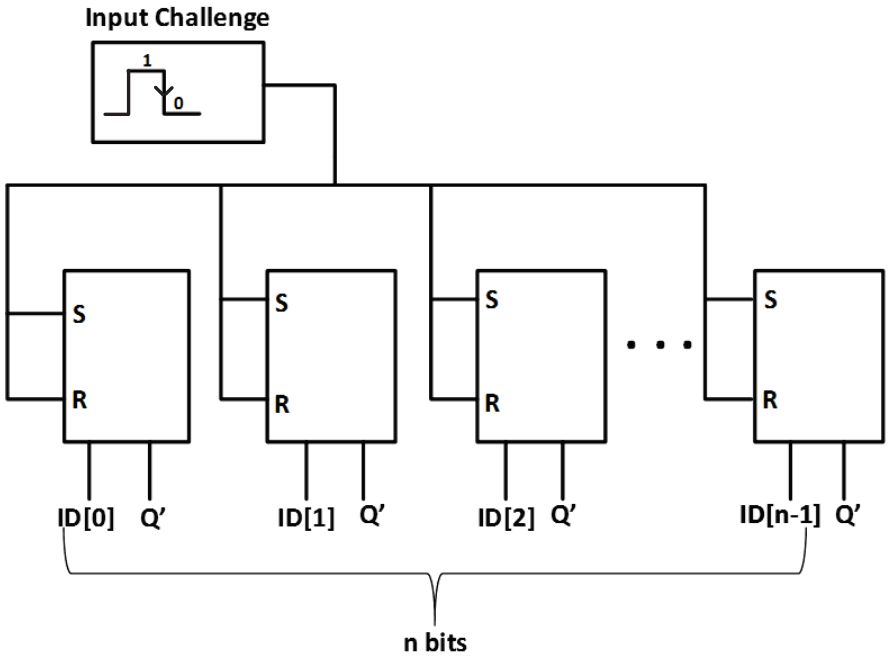

Figure 3.3: Proposed PUF

for processors belonging to four different design categories. For example, consider DES core which has 3,714 registers, here the generated key can be 3,714 bit wide which will be unique for different processors of the same type and this makes the PUF more reliable.

3.4 Chapter Summary

The input condition S=R=1 is applied followed by S=R=0, must be avoided as it is inconsistent condition for which the outputs Q and Q' undergo race condition and depending on the NAND gates in the feedback path, the output Q can settle at 0 or 1. In this work we leverage this state of the flip-flop by using the concept of gate delays and generate a unique key. Most of the PUFs presented in the literature use additional circuitry, in this work we use the flip flops which are already present in the design to create a unique ID for the device.

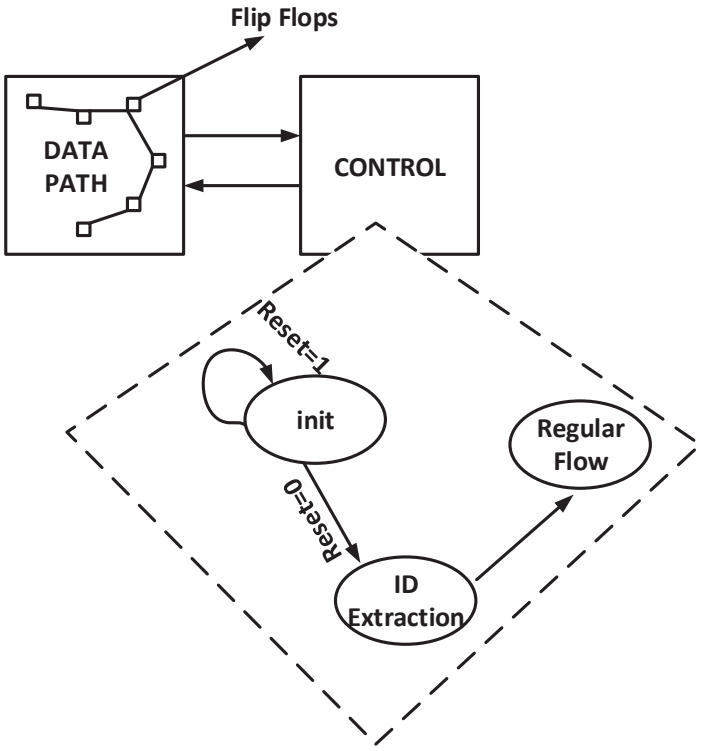

Figure 3.4: Design Controller

Table 3.2: Number of Registers Bits in Various Processor Cores

Design Category	Design	No. of bits (approximately)
DSP	Cannyedge Detector	2,027
	Low-Power IIR Filter	1,017
Crypto Core	AES128	1,072
	DES core	3714
Processor Core	Arm4	2,676
	8-bit uP	386
Communication Core	ADAT Receiver	888
	16550 UART Core	578

CHAPTER 4: EXPERIMENTAL RESULTS

To check the uniformity and reliability of the PUF, we simulated a design at SPICE level with process variations and determined the inter-die and intra-die Hamming Distance (HD). The effect of process variations are measured using Monte-Carlo analysis for 1000 iterations for a 16-, 32-, 64-, and 128-bit keys in 90 nm, 45 nm, and 32 nm CMOS technology nodes (PTM models).

4.1 Monte Carlo Simulation Flow

To perform Monte Carlo simulation, we need to follow the flow as shown in Figure 4.1.

The first step is to choose the variable which has to be varied. Here, in this case the variable which has to be varied is length (L) of a transistor. In the next step we need to determine the values of mean (ρ) and variance (σ) which gives us the percentage of variation required. There are two types of variations available i.e., local variation and global variation. In this case, we are using local variation of 33% for inter chip and 15% for intra chip. The outputs for multiple iterations are obtained and are XORed with each other to calculate the Hamming Distance (HD).

4.2 Inter-die Variations

In this case, the length of each transistor is varied about 33% of nominal length. Figure 4.2 shows the HD for 16-, 32-, 64-, and 128-bit keys. We can observe that the HD is more than 50% of the key length, which is ideal. The maximum HD for 16-, 32-, 64- and 128-bit registers is 10, 20,

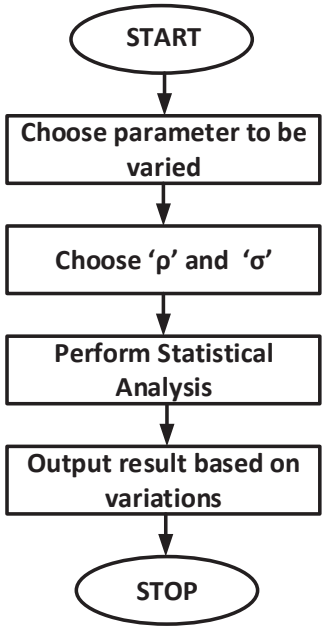

Figure 4.1: Monte Carlo Simulation Flow

53, and 52 respectively. Figure 4.2 shows that the key produced is unique for each die. The same study is performed in 45nm and 32nm technology nodes and the corresponding HD plots are shown in Figures 4.3 and 4.4. Table 4.1 gives the information about maximum and minimum HD for 16-, 32-, 64- and 128-bit keys. The Max. Hamming Distance % for 128-bit key is approximately equal to 50% because the number of iterations for which the experiment is carried out is 1000. This result will be improved if the experiment is carried out for 10000 iterations.

4.3 Intra-die Variations

The reliability of a PUF can be measured by its intra-die HD which should be close to 0 for all possible CRPs of a PUF. Intra-die HD is measured by XORing the responses of the PUF at different temperatures (0C to 80C) with 15% variation in the length of a transistor for 1000 iterations. Figure 4.5 shows the intra-die hamming distance for 16-, 32-, 64-, and 128-bit registers.

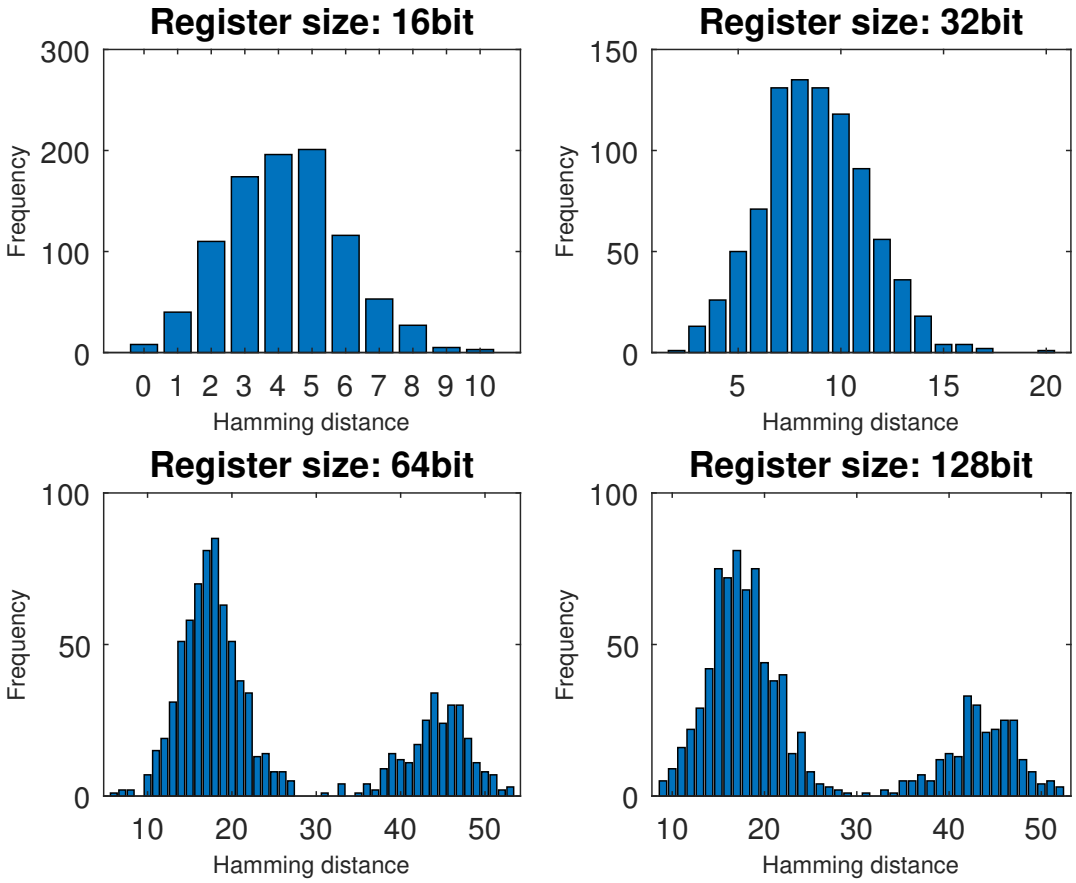

Figure 4.2: Inter-die Hamming Distance (90nm)

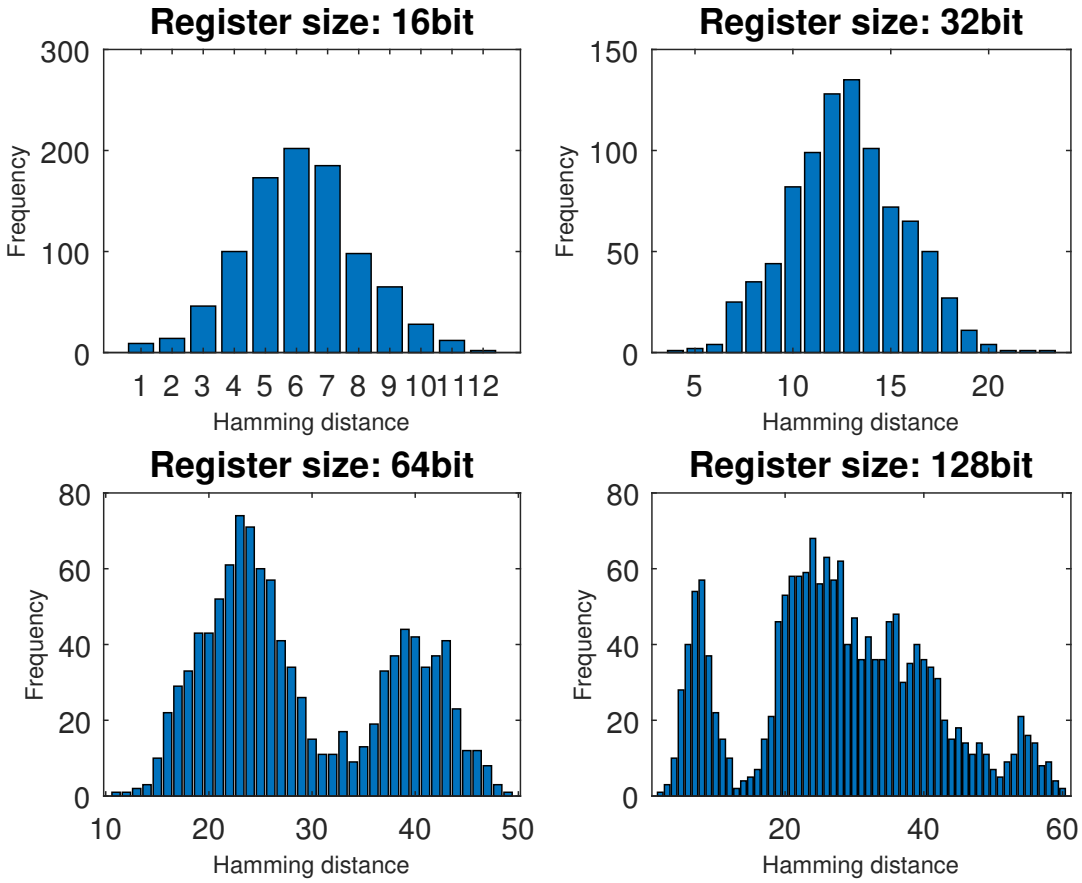

Figure 4.3: Inter-die Hamming Distance (45nm)

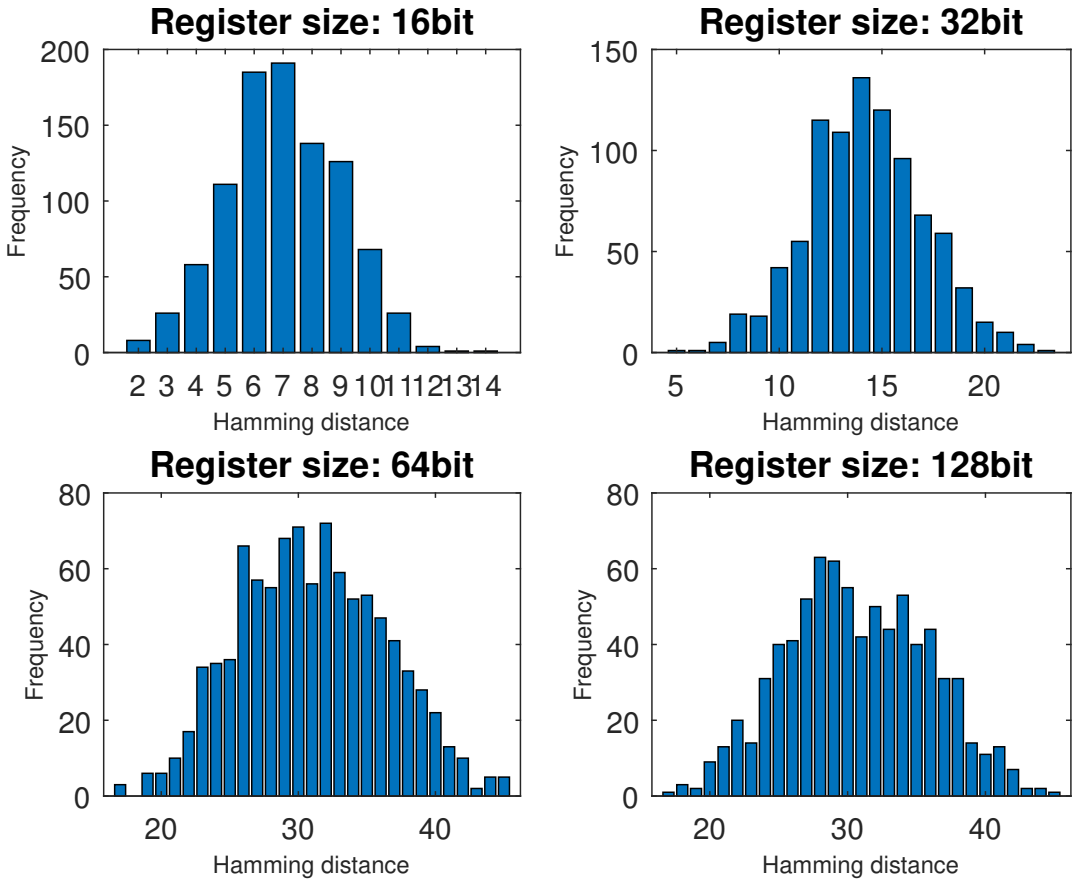

Figure 4.4: Inter-die Hamming Distance (32nm)

It can be seen that the frequency for which the HD is 0 for 16-, 32-, 64- and 128-bit registers are 92.3% , 92.2%, 90.7%, and 92.7% respectively. The same study is performed in 45nm and 32nm technology nodes and the corresponding PUF uniqueness is demonstrated in Figures 4.6 and 4.7. From these Figures, we can see that the intra-die variation is close to 0%. Table 4.2 shows the percentage of iterations for which the key has a HD of 0 for 16-, 32-, 64-, and 128-bit.

Table 4.1: Inter-Die Variations

Register Size	Technology node (nm)	Min. Hamming Distance	%	Max. Hamming Distance	%
16	90	2	25.00	10	62.50
32	90	7	21.80	18	50.00
64	90	11	17.10	52	81.25
128	90	13	10.10	51	40.00
16	45	2	25.00	12	75.00
32	45	8	25.00	21	65.60
64	45	15	23.40	50	78.00
128	45	10	7.80	60	46.80
16	32	2	25.00	12	75.00
32	32	8	25.00	22	68.75
64	32	20	31.25	47	73.40
128	32	20	17.18	49	39.00

4.4 Chapter Summary

Inter chip and Intra chip variations are performed on the proposed PUF which shows that the PUF is reliable and unique. SPICE level simulation is performed and the uniqueness of the key is over 50% for all simulations. The reliability of the PUF is close to 0 which means the key generated remains the same for multiple iterations on same die.

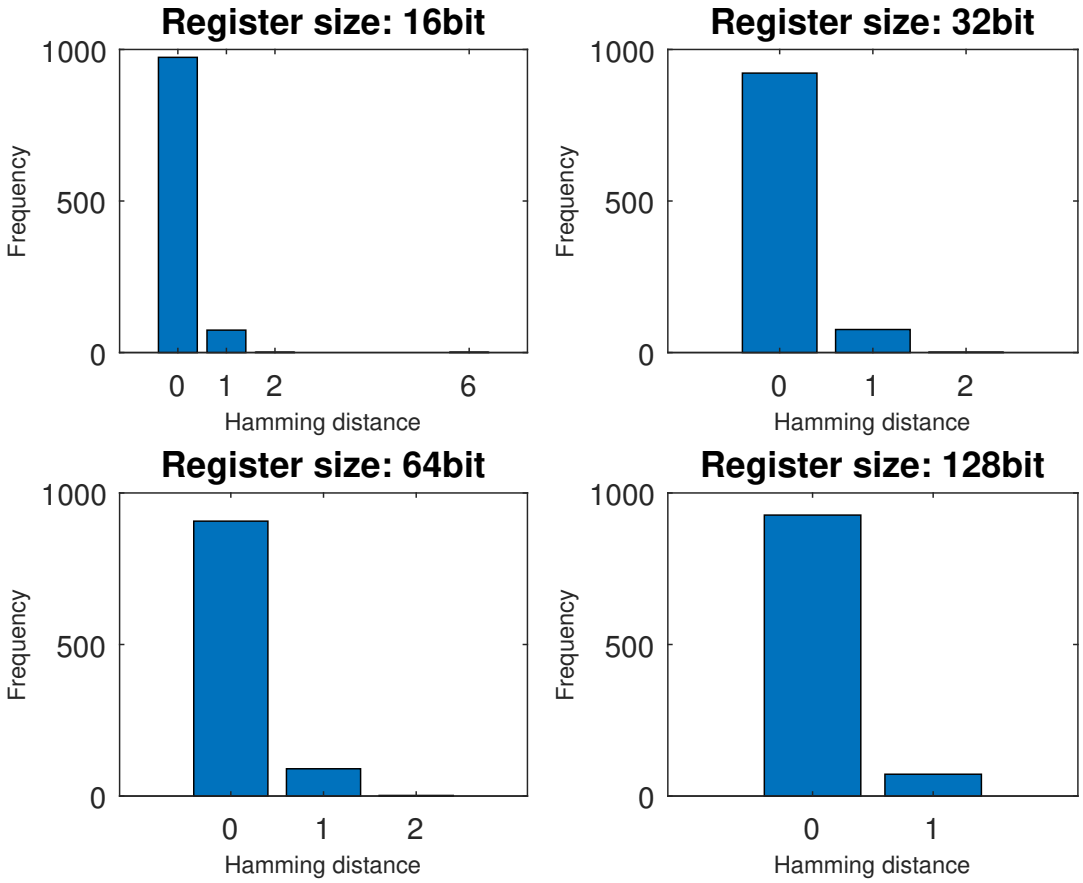

Figure 4.5: Intra-die Hamming Distance (90nm)

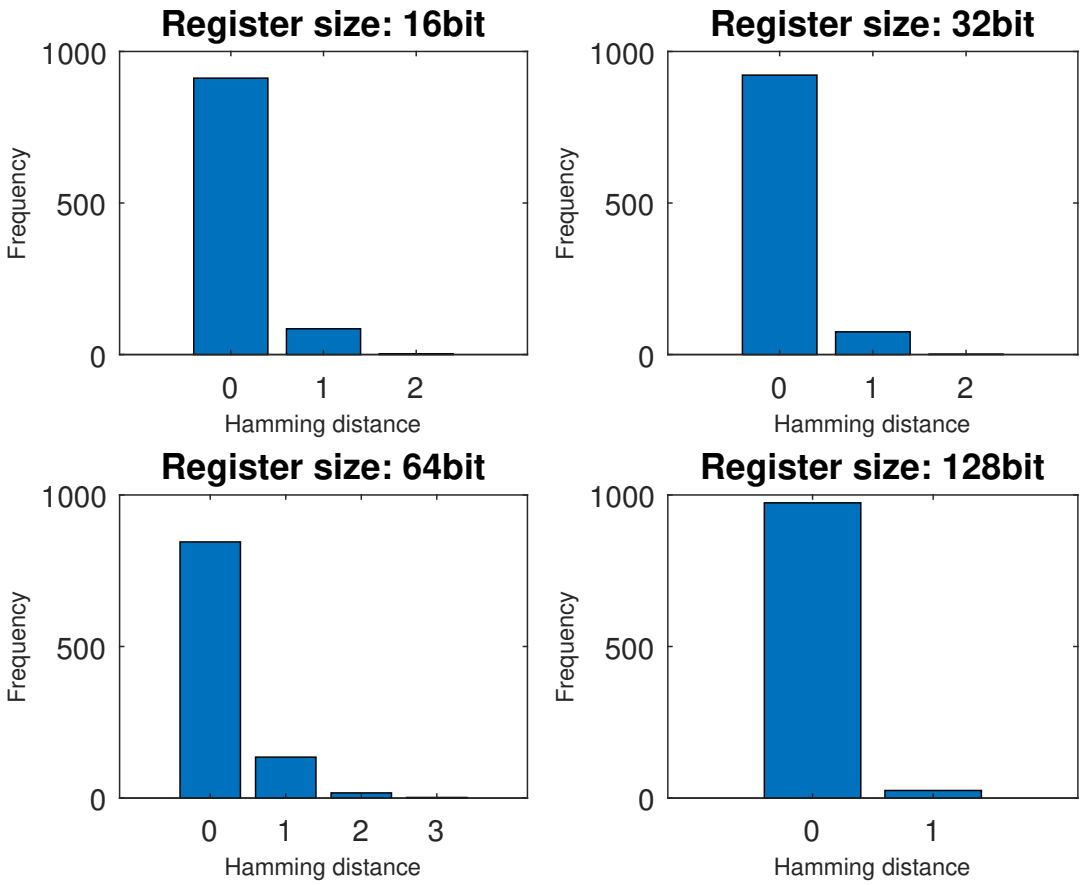

Figure 4.6: Intra-die Hamming Distance (45nm)

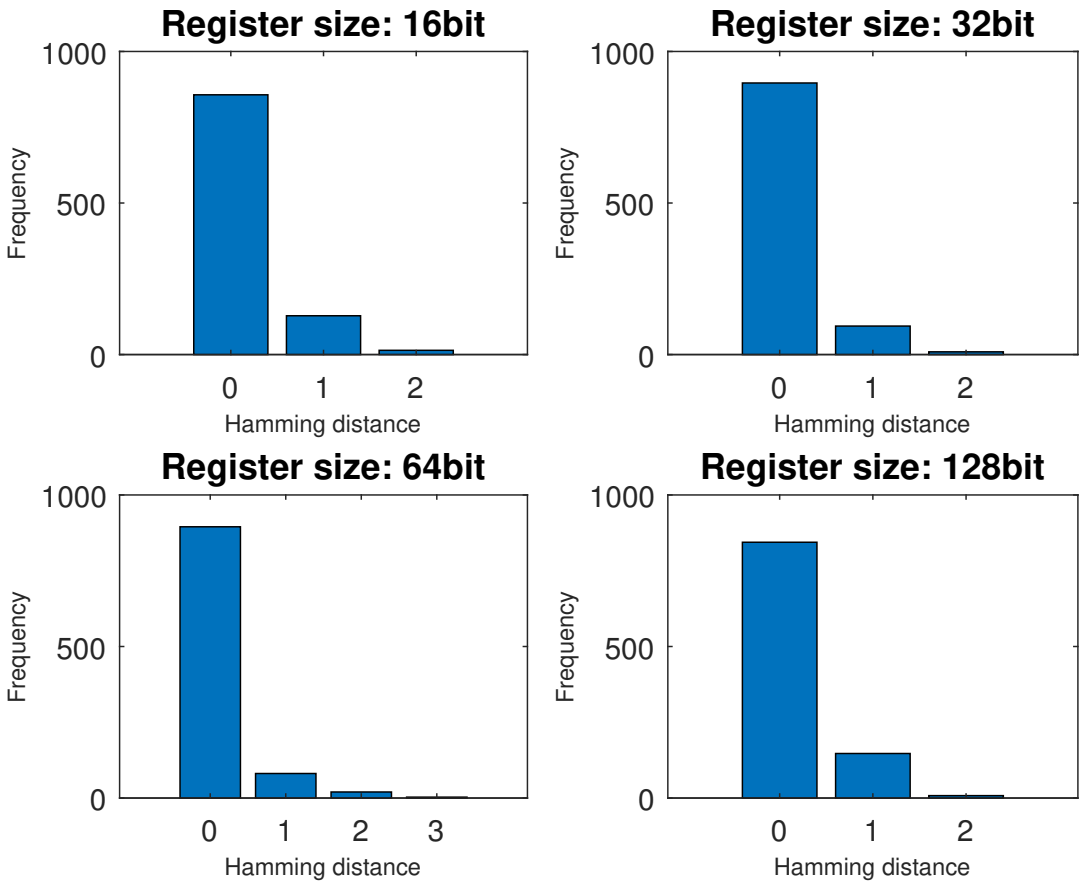

Figure 4.7: Intra-die Hamming Distance (32nm)

Table 4.2: Intra-Die Variations

Register Size	Technology node (nm)	Iterations with 0 HD	%
16	90	974	97.40
32	90	922	92.30
64	90	903	90.30
128	90	927	92.70
16	45	912	91.20
32	45	922	92.20
64	45	845	84.50
128	45	977	97.70
16	32	857	85.70
32	32	896	89.60
64	32	895	89.50
128	32	844	84.40

CHAPTER 5: CONCLUSION

We have proposed a new PUF architecture using the SR flip-flop which can be used for hardware security and authentication. As this PUF uses existing flip-flops in a design, there is no additional overhead on the system, which improves delay and requires low-power. The proposed PUF response is evaluated by systematic evaluation methodology which demonstrates above 85% of iterations have 0 intra-HD for most of the responses and inter-die HD of approximately 50% with sufficient randomness in the response.

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] J. Kong and F. Koushanfar. "Processor-based strong physical unclonable functions with aging-based response tuning". *IEEE Transactions on Emerging Topics in Computing*, 2(1):16–29, March 2014.

[4] H. Chang and S. S. Sapatnekar. "Statistical timing analysis under spatial correlations". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(9):1467–1482, Sept 2005.

[5] Y. Su, J. Holleman, and B. P. Otis. "A digital 1.6 pJ/bit chip identification circuit using process variations". *IEEE Journal of Solid-State Circuits*, 43(1):69–77, Jan 2008.

[6] Y. Lao and K. K. Parhi. "Statistical analysis of MUX-based Physical Unclonable Functions". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(5):649–662, May 2014.

[7] Y. Su, J. Holleman, and B. Otis. "A 1.6pJ/bit 96% stable chip-ID generating circuit using process variations". In *2007 IEEE International Solid-State Circuits Conference.*, Feb 2007.

[8] D. E. Holcomb, W. Burleson, and K. Fu. "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags". Proceedings of the conference on RFID Security January, 2007.

[9] 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.

[10] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. "Physical One-Way Functions". *Science*, 297(5589):2026–2030, 2002.

[11] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal. "Design and implementation of PUF-based "Unclonable" RFID ICs for anti-counterfeiting and security applications". In *2008 IEEE International Conference on RFID*, pages 58–64, April 2008.

[12] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S Devadas, and J. Schmidhuber. "Modeling attacks on Physical Unclonable Functions". In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM.

[13] 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.

[14] D. Suzuki and K. Shimizu. "The glitch PUF: A new delay-PUF architecture exploiting glitch shapes ". In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 366–382, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[15] A. R. Krishna, S. Narasimhan, X. Wang, and S. Bhunia. "MECCA: A robust low-overhead PUF using embedded memory array". In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, pages 407–420, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[16] 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.

[17] 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.

[18] 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.

[19] A. Maiti, V. Gunreddy, and P. Schaumont. *"A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions"*, pages 245–267. Springer New York, New York, NY, 2013.