A PUF based Lightweight Hardware Security Architecture for IoT

A Dissertation Proposal Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Mesbah Uddin

May 2020

© by Mesbah Uddin, 2020
All Rights Reserved.

Dedication

This research is dedicated to my parents who have been supportive in my whole life,

Md Mohi Uddin

and

Shahina Akter

Acknowledgments

I would like thank to my Advisor, Dr. Garrett Rose for his immense support throughout my Ph.D. career. He has always given me guidance, encouragement, financial support and has always put his trust in me.

I would like to extend my gratitude to Dr. Nicole McFarlane, Dr. Jinyuan Stella Sun and Dr. Andy Sarles for finding time to serve on my Ph.D. committee. I would also like to thank Dr. Syed Islam for his guidance as a Professor and as a guardian for Bangladeshi students at UT.

I am very grateful to the EECS department and all its staff especially Dana Bryson and Melanie Kelley for helping me in any academic issues. I would also like to thank my lab-mates Md Badruddoja Majumder, Gangotree Chakma, Sherif Amer, Md Musabbir Adnan, Sagarvarma Sayyaparaju, Ryan Weiss, Md Sakib Hasan and all others for their cooperation and friendship.

I am very grateful to my wife, Anika Zaman for always being there for me. I have always had the deepest support and encouragement from my parents, Md Mohi Uddin and Mrs Shahina Akter, my brother Md Meftah Uddin and my sister Maliha Ibnat. I am also grateful for the Bangladesh student community of Knoxville and all its members and the various organizations of UT like International House, RUF International, Bridges International etc. for making my life at Knoxville easier and enjoyable during my Ph.D.

Abstract

With an increasing number of hand-held electronics, gadgets, and other smart devices, data is present in a large number of platforms, thereby increasing the risk of security, privacy, and safety breach than ever before. Due to the extreme lightweight nature of these devices, commonly referred to as IoT or 'Internet of Things', providing any kind of security is prohibitive due to high overhead associated with any traditional and mathematically robust cryptographic techniques. Therefore, researchers have searched for alternative intuitive solutions for such devices. Hardware security, unlike traditional cryptography, can provide unique device-specific security solutions with little overhead, address vulnerability in hardware and, therefore, are attractive in this domain.

As Moore's law is almost at its end, different emerging devices are being explored more by researchers as they present opportunities to build better application specific devices along with their challenges compared to CMOS technology. In this work, we have proposed emerging nanotechnology based hardware security as a security solution for resource constrained IoT domain. Specifically, we have built two hardware security primitives i.e. physical unclonable function (PUF) and true random number generator (TRNG) and used these components as part of a security protocol proposed in this work as well. Both PUF and TRNG are built from metal oxide memristors, an emerging nanoscale device and are generally lightweight compared to their CMOS counterparts in terms of area, power, and delay. Design challenges associated with designing these hardware security primitives and with memristive devices are properly addressed. Finally, a complete security protocol is proposed where all of these different pieces come together to provide a practical, robust, and device-specific security for resource-limited IoT systems.

Table of Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Research Goal and Summary . . . 2

1.3 Original Contributions . . . 2

1.4 Dissertation Overview . . . 3

2 Background **5**

2.1 Internet of Things (IoT) . . . 5

2.2 Emerging Nanoscale Technology: Memristor . . . 7

2.2.1 Emerging Devices . . . 7

2.2.2 Memristor . . . 10

2.3 Hardware Security as IoT Security . . . 13

2.3.1 Traditional Security: Cryptography . . . 13

2.3.2 Introduction to Hardware Security . . . 15

2.3.3 Physical Unclonable Function (PUF) . . . 15

2.3.4 True Random Number Generator (TRNG) . . . 24

3 Design and Analysis of Fundamental Hardware Security Components **27**

3.1 Memristor Crossbar PUF . . . 27

3.1.1 Introduction to HfO2 Memristor . . . 27

3.1.2 Environmental Modeling of a HfO2 Memristor . . . 28

3.1.3 Working Principle of XbarPUF . . . 32

3.1.4 Clock Frequency and Other Parameter Selection . . . 34

3.1.5 Choice of Load Resistance . . . 35
3.1.6 Shift Back to Fixed Challenge Scheme . . . 36
3.1.7 Reverse Read Scheme . . . 38
3.1.8 Sense Amplifier . . . 38
3.2 Specialized Peripherals for Memristor based Circuits . . . 39
3.2.1 Current Compliance . . . 39
3.2.2 Forming Circuitry . . . 40
3.2.3 Memristor Read-Write-Form Circuitry . . . 41
3.2.4 Sense Amplifier for Memristor Crossbar Circuits . . . 46
3.3 Design of Twin Memristor TRNG . . . 61
3.3.1 Motivation . . . 61
3.3.2 Our Design . . . 61
3.3.3 Effect of Environmental and Process variation on Existing Single Memristor TRNG . . . 63
3.3.4 Variation Modeling for Our Designed TRNG . . . 67
3.3.5 Results and Analyses of Designed Twin-Memristor TRNG . . . 69
3.3.6 Output Correction . . . 71

4 Design and Robustness Analysis of XbarPUF 73
4.1 Design Analysis of XbarPUF . . . 73
4.1.1 Noise Margin Analysis . . . 73
4.1.2 Security Analysis of the XbarPUF . . . 75
4.1.3 Detailed Reliability Analysis . . . 79
4.1.4 Performance Overhead Analysis . . . 82
4.1.5 Overhead analysis for Other Memristors . . . 83
4.1.6 On-Chip Design . . . 87
4.2 Robustness Analysis Against ML (Machine Learning) based Attacks . . . 87
4.2.1 Development of High Level Behavioral Model of Memristor . . . 89
4.2.2 Development of High Level Behavioral Model of XbarPUF . . . 90
4.2.3 Introduction to Machine Learning Attack Models . . . 91

4.2.4 Experimental Setup . . . 92
4.2.5 Modeling Attack on an XbarPUF . . . 92
4.2.6 Mitigation of ML Attacks . . . 93
4.2.7 Results and Discussions . . . 94

5 Lightweight Security for IoT/Embedded System 98

5.1 Motivation . . . 98
5.2 Security vulnerability of IoT processor . . . 99
5.3 Our Security Solution . . . 99
5.4 Our Security Protocol . . . 102
5.5 System Implementation . . . 107
5.5.1 One Time Pad Implementation . . . 107
5.5.2 True Random Number Generator (TRNG) . . . 110
5.5.3 Redesigned XbarPUF for IoT . . . 110
5.5.4 RRAM for Non-Volatile Storage . . . 111
5.5.5 Time Multiplexing of PUF to Lower Power Consumption . . . 112
5.5.6 SRAM as Part of Reliability Enhancement Block . . . 112
5.5.7 Sneak-Path based Integrity Checking . . . 113
5.5.8 Control Circuit Design . . . 113
5.6 Initial Design without Reliability Correction . . . 117
5.6.1 Overview . . . 117
5.6.2 Results for Initial Prototype . . . 118
5.7 Reliability Enhancement Technique . . . 120
5.7.1 All-Agree Voting/Veto Technique . . . 120
5.7.2 Majority Voting . . . 122
5.7.3 Chosen Reliability Enhancement Technique . . . 124
5.8 Working Principle . . . 125
5.9 Possible Attack Scenarios . . . 126
5.9.1 Malicious read . . . 126
5.9.2 Malicious Write . . . 127

5.9.3 Readout or Alteration of the PUF Challenge or Secure Tag 128
5.9.4 Modeling Attacks 129
5.10 Results: PUF Security Analysis 129
5.11 Overall Security Evaluation 134
5.11.1 Malicious Read 134
5.11.2 Malicious Write 135
5.11.3 Modeling Attacks 136
5.11.4 Readout/Alteration of Secure Information 137
5.12 Overall Performance Analysis 138

6 Conclusion and Future Plan **144**
6.1 Future Works 144
6.2 Summary 145

Bibliography **147**

Appendices **158**
A Calculation of Optimal Load Resistance for XbarPUF 159
B Memristor (HfO_2) Model Used in This Work 160
B.1 Verilog-A Model for HfO_2 Memristor: 160
B.2 MATLAB Model of a Memristor for Quick Behavioral Simulation: . . 169
C Behavioral XbarPUF Model Used for Response Generation and Power Calculation 171
D How to Perform Tests on the Chip 179
D.1 Setting up the Probepad Connection 179
D.2 PSOC Microcontroller and Source Meter to Generate Inputs 179
D.3 Setting up the Connection with Probepads 182
D.4 Performing Simple Functionality Test 182
D.5 Oscilloscope to View and Collect the Output 183
E Probability of Error with Majority Voting Technique 185

Vita **187**

List of Tables

3.1 Mean and standard deviation for device level parameters of a HfO_2 memristor considered in this work [69] 31
3.2 Temperature coefficients for different memristor parameters [69] 31
3.3 Comparison of our designed SA with different circuits from different topology [73] 55
3.4 Percent yield of our designed SA for different transistor sizes and supply voltage [73] 57
3.5 Sensing delay (nS) of our designed SA for different transistor sizes and supply voltage [73] 57
3.6 Energy(Joule) per cycle of our designed SA for different transistor sizes and supply voltage [73] 57
3.7 Percentage of 0's to the sum of 0's and 1's for different operating conditions [67] 71

4.1 Impact of 2% variation of load resistance on the security properties of a 32×2 XORed XbarPUF [69] 80
4.2 Performance of the XORed XbarPUF with variable temperature (10^oC-100^oC) and with different cycle-to-cycle variation for the memristor parameters [69] 80
4.3 Power estimation of a HfO_x memristor based XbarPUF from MATLAB and cadence [71] 84
4.4 Switching Parameters for Metal-Oxide Memristors [71] 86
4.5 Power Consumption during different phases of a 32×2 XbarPUF [71] 86

4.6 Results from modeling attacks on basic XbarPUF (no mitigation) for different machine learning algorithms . . . 93
4.7 Column swapping logic implemented in this demonstration [71] . . . 94
4.8 Results from modeling attacks after mitigation techniques applied for different machine learning algorithms . . . 96

5.1 Switching Parameters (mean value) for TiO_x Memristors [48] . . . 115
5.2 Performance of modified XbarPUF used in this design [74] . . . 119
5.3 Overhead in different phases of the system [74] . . . 120
5.4 Security properties of the tag generation method . . . 135
5.5 Results from modeling attacks using different machine learning algorithms against TiO_2 memristor based XbarPUF [70] . . . 138
5.6 Total area in terms of transistor count for different components of our security architecture [70] . . . 141
5.7 Power consumption of the security architecture in different stages (State 1 and 2 involve a reset of the whole memristor crossbar, 64×64 at once and thus have large static current) [70]. . . . 141
5.8 Delay overhead of the system in different stages of operation . . . 142
5.9 Performance comparison with state-of-the-art lightweight hardware security techniques [70] . . . 143

List of Figures

2.1 The Internet of things (IoT) [68] 6
2.2 Memristor symbol (left) and its IV curve showing hysteresis (right) 11
2.3 Arbiter PUF 17
2.4 Two input/two output switch (left two) and its simple implementation with two MUXes (right) 18
2.5 Ring oscillator: a chain of an odd number of inverters 19
2.6 A ring oscillator PUF 19
2.7 Standard metrics used for the evaluation of a PUF 21
2.8 TRNG sources: clock jitter, noise, metastability 26

3.1 Illustration (left) and TEM (transmission electron microscopy) image (right) of a hafnium-oxide memristor embedded between the M1 and M2 layers. In addition, the Illustration depicts a seamless integrated 1 memristor 1 transistor (1M1T) structure where the transistor acts as the current limiting device [8, 69]. 28
3.2 IV measurements of a memristor (left) and simulated IV characteristics using our memristor model (right) [69]. 30
3.3 Simulated effect of (worst case) aging on both HRS and LRS of a memristor over time [69]. 32
3.4 Schematic of a $N \times M$ XbarPUF circuit (with XORing) [72, 71]. 33
3.5 Two resistance in series, one is the equivalent memristance of a crossbar column and another is the load resistance [69]. 36

3.6 Read-monitored write approach to gradually check memristance of a pair of memristors until one finishes switching. 37
3.7 An NMOS transistor with appropriate gate voltage to control the current through a memristor during set. 40
3.8 Forming circuitry of a HfO_2 memristor with 65nm CMOS technology 41
3.9 Schematic of the the read, write and forming circuitry for the HfO_2 memristor used in this work 42
3.10 Layout of the designed read, write and forming circuitry for the HfO_2 memristor ($18.31\mu m \times 22.60\mu m$) 43
3.11 Successful forming of three different memristors with our designed memristor read-write-form circuitry. The forming can be easily identified by the presence of a sudden increase in current in the IV plots. The gate voltage to control the current compliance transistor is also varied to demonstrate forming at different levels which is why there are multiple forming-like spikes in these plots 44
3.12 The current before (left figure) and after forming (right figure) is performed. The current is very low ($\sim$pA) before forming but was high ($\sim$50-70μA) after forming. 44
3.13 This plot shows a successful reset of a memristor with our read-write-form circuitry. The plot clearly shows how current increases linearly with increasing reset voltage voltage i.e. there is no change in memristance or resistance. After when reset voltage (-0.5V) is reached, memristors finally resets to a high resistance where there is a sudden and noticeable drop in current. . . . 45
3.14 Plot shows the stable current through the memristor when formed using several different current compliance control voltage (Vref). These several different current levels prove the ability to form these memristors at different LRS values with confidence. 45
3.15 Schematic diagram of the our designed sense amplifier circuit suitable for memristor based crossbar architectures [73] 51
3.16 Application of the our designed sense amplifier for the XbarPUF [73] 52

3.17 An equivalent input-output circuit for our sense amplifier circuit when used in an XbarPUF [73] . . . 52
3.18 Waveform for transient simulation of proposed SA for input bit-line voltages of 150mV and 155mV with VDD = 0.4V [73] . . . 56
3.19 Reliability across multiple evaluations of our sense amplifier for different transistor size and differential voltage . . . 59
3.20 Layout of the designed sense amplifier (8.049μm×13.03μm) . . . 60
3.21 On-chip test results from our designed sense amplifier . . . 60
3.22 Conceptual circuit diagram of (left) existing switching time based single memristor TRNG [26], and (right) our designed twin memristor TRNG based on HRS difference [67]. . . . 62
3.23 Histogram of bit bias with process variation for (top) existing single memristor design and (bottom) proposed twin memristor design [67] . . . 72
3.24 Histogram plot showing the trend for shift in distribution with temperature and voltage variation for both single memristor TRNG (left two figures) and our designed twin memristor TRNG (right two figures). Top two figures represent variation due to temperature for single and twin memristor TRNG, respectively while bottom two figures are for voltage variation. Existing single memristor TRNG displays a large variation while our design almost unchanged [67]. . . . 72

4.1 Relationship between noise margin and the load resistance [69]. . . . 74
4.2 Relationship between maximum achievable noise margin and the number of rows (N) [69]. . . . 75
4.3 Relationship between the maximum achievable noise margin and HRS/LRS ratio [69]. . . . 76
4.4 Security performance of XORed XbarPUF for different crossbar size [69]. . . . 77
4.5 Dependence of uniqueness (left) and bit-aliasing (right) with HRS/LRS ratio and the absolute values of HRS and LRS. The base LRS was 30KΩ for both of these plots [69]. . . . 78

4.6 Reliability of the XbarPUF with respect to increasing LRS [69]. 81
4.7 Reliability of the XbarPUF with age [69]. 81
4.8 Dependence of average power on HRS/LRS ratio and the absolute values of HRS and LRS [69]. 83
4.9 Area comparison of XbarPUF with Arbiter PUF [69]. 84
4.10 Relationship between average static power consumption and size of the XbarPUF [71] 87
4.11 Schematic of the designed 1-bit memristor crossbar PUF 88
4.12 Layout of the designed 1-bit memristor crossbar PUF (19.59μm×33.63μm) . 88
4.13 Simulated IV characteristics of a metal-oxide (HfO_x) memristor for 200 set/reset cycles for a 100kHz clock. [71] 90
4.14 An implementation of column swapping circuit [71]. 94
4.15 XbarPUF with the inclusion of column swapping logic [71]. 95
4.16 Modeling accuracy of LR vs. the size of dataset [71]. 97
4.17 Modeling accuracy of Gaussian Native Bayes vs. the size of dataset [71]. . . 97

5.1 An embedded processor (either regular or energy harvesting) backs-up data for data-forwarding but the data left on NVM is unprotected. 100
5.2 A unique random challenge is applied to a PUF to generate a cryptographic key for secure back-up and restoration of data in an embedded processor. . . 100
5.3 Control flow graph for our secure back-up and restore protocol [70] 108
5.4 System block diagram showing different components of our designed security protocol [70] 109
5.5 High-level conceptual block diagram of the initial security system [74] 118
5.6 Block diagram of the initially designed prototype system [74] 118
5.7 'All-agree' reliability enhancement technique for different number of evaluations for different bit-flip probability. It is more efficient at detecting highly unstable bits with increasing number of evaluations [70] 123
5.8 Effectiveness of majority voting for reliability enhancement technique for different number of evaluations and with different bit-flip probability [70] . . 124

5.9 Waveform for a complete secure backup and restore cycle showing how different states control different operations of the system. Counter keeps track of the number of valid or legal key bits and stops each time a bit error occurs 127
5.10 This figure shows average uniqueness results for 500 different chips [70]. Even for different challenges, this value is very close to ideal value of 0.5. 131
5.11 Summary of results for bit-aliasing for all 32 bits from 500 different chips. The minimum and maximum value for each bit are also shown [70]. 131
5.12 Summary of results for average uniformity, diffuseness and steadiness from 500 different challenges and for 25 different chips [70]. 132
5.13 Detailed reliability results generated from 500 different cycles for 25 different chips and 25 different challenges [70] 133
5.14 Probability of success in a spoofing attack with number of trials. The more trials an attacker can perform, the higher the chance that the data matches the tag. No. of effective trial is 1 in this protocol since the tag is updated on each cycle [70]. 136
5.15 Figure shows the average number of bits to produce 16 'good' bits from 50 different chips. The results are generated for three different cycle-to-cycle variations and a 50^0C temperature change [70]. 139
5.16 Probability of bit-flip for a same chip for 10 unique challenges. Plots showing the bit-flip probabilities for all 32 bits from a PUF response, evaluated for 500 cycles. Different challenges cause different bits to flip i.e. there is no single set of globally unreliable bit [70]. 140

1 Partial snapshot of the layout of the chip where it contains a 12×2 probepad for the sense amplifier circuit. 180
2 12×2 probepad connection to a test circuit inside of a probe station, the image is taken from a microscopic view. 181
3 Connection between the inputs generated by the PSOC microcontroller and wires coming from the probepad 183

4 Output shown on a oscilloscope during testing of a sense amplifier circuit from a fabricated chip. 184

Chapter 1

Introduction

1.1 Motivation

Traditional cryptography has been providing security solution for decades and are mathematically robust but they introduce a large overhead to a system. For a large system, this overhead may be negligible but with the emergence of smart devices and small embedded systems i.e. Internet of Things (IoT), there is a growing need of providing security for these extremely resource limited systems. As traditional cryptography or software based security is impractical due to their heavy overhead, innovative solution using hardware based security are becoming more and more important. Physical unclonable function (PUF) [53] is a hardware security primitive that has been a highly researched topic in the past 20 years (since 2001). PUF can generate random, unique hardware-specific signatures that can be used in many security applications and thus researchers have proposed hundreds of different PUFs. On the other hand, with Moore's law [50] neared to its end since early 2000s, several new emerging technologies are being explored extensively. Among those, metal-oxide memristors [63] are very promising because of their non-volatility, CMOS integration, low read/write energy, high on-off ratio, usability in both logic and memory circuits. Thus the future is going towards post-CMOS era where these novel nano-devices like memristors would be in use in a lot of electronic designs and hardware based security solutions would be necessary for these billions of smart small devices. Thus there is a growing need for development of hardware security primitives like PUFs using emerging nano-devices like memristors. These

novel devices and hardware specific security measures introduce different forms of challenges and thus the research scope is huge in this domain.

1.2 Research Goal and Summary

The goal of this work is to develop a very lightweight hardware security architecture for resource constrained IoT domain leveraging the solutions and opportunities provided by emerging nanoscale devices.

The research is specifically focused greatly on the design and analysis of a hardware security primitive called the PUF. The PUF itself is built from memristors. The memristor itself is an emerging nano-device with a high manufacturing variability which is used as the source of entropy for the memristive PUF circuits that we have used in this work. The initial work is focused on improving a practical model of HfO_2 memristor by including environmental effects and aging on its characteristics. Then a detailed device and circuit level parameter exploration based on security and overhead performance of memristive PUF is performed. A machine learning based modeling attack is also implemented on this PUF and further circuit level modification is done to mitigate this vulnerability. Different peripheral circuits, especially memristor's read-write-form circuity and a practical sense amplifier for memristor PUF is also designed. Besides, another hardware security primitive called true random number generator (TRNG) is also designed from memristor and is shown to be robust against environmental changes compared to existing designs. Finally, a complete security architecture is built from memristor based PUF, TRNG, and RRAM (resistive random access memory). Memristor PUF along with our designed reliability enhancement technique is used to provide secret and unique key to a resource-limited embedded or IoT system to provide robust security for its back-up data at the absence or scarcity of power.

1.3 Original Contributions

The original contributions of my doctoral work until now are listed below:

- Environmental modeling of HfO_2 memristors, specifically inclusion of temperature change, aging, and also stochasticity in its characteristics
- Security analysis of memristor crossbar PUF against varying device parameters, size and temperature
- Scalability and overhead analysis of memristor crossbar PUF
- Design of abstract model for memristor and memristor crossbar PUF and application of machine learning based modeling attack on PUFs
- Successful demonstration of modeling attack on memristor crossbar PUF and circuit redesign to improve robustness against machine learning based attacks
- Design of a practical sense amplifier for memristor based PUFs and read-write-form circuitry for memristors
- Comprehensive theoretical analysis of memristor based true random number generator (TRNG) and designing an improved twin memristor based TRNG
- Design and analysis of a run-time reliability enhancement technique for memristor crossbar PUF
- A complete security protocol and transistor level architecture design for resource-constrained IoT or embedded processors

1.4 Dissertation Overview

This dissertation is divided into six chapters. Chapter 1 or this chapter provides some context about the research and lists out original contributions of my doctoral work. Chapter 2 provides background on emerging nanoscale devices like memristors, and hardware security primitive like PUF and TRNG. Chapter 3 introduces circuit design works for memristor XbarPUF, different peripheral circuits (e.g. sense amplifier) for XbarPUF, the TRNG and analyzed against manufacturing process variation and environmental changes. Chapter 4 provides a thorough design analysis of XbarPUF and also analyzes its robustness or

vulnerability against machine learning based algorithms along with mitigation techniques against these attacks. Chapter 5 uses all the designed components and knowledge so far to provide a robust and lightweight hardware security protocol and architecture to secure the non-volatile memory of an embedded processor or IoT device. Finally, chapter 6 provides a summary of the works done for this dissertation and future work extensions are discussed.

Chapter 2

Background

2.1 Internet of Things (IoT)

The internet of things or IoT has been a buzzword for the past few years as the world has seen a rapid growth in this domain. IoT generally refers to embedded devices, sensors etc. which can gather data, send information via a channel or the internet to somewhere else to make a decision based on its gathered data. Smart home appliances, smart medical devices, smart automotive, sensor devices that can communicate, connected embedded processors. etc. all can be considered as IoT. In most cases, IoT devices are interconnected in an environment and share data with each other to ease making decisions. For example, in a smart home, many different such devices like smart smart bulb, thermostats, refrigerator, microwave oven, fans etc. can be controlled via a smartphone which ease the way of our life. This is illustrated in Figure 2.1 [68, 58]. IoT is the fastest growing field in the world right now, with global spending expected to cross 1.4 trillion by the end of this year (2020). In 2019, there were more than 26 billions IoT devices worldwide and the number is expected to reach 75 billions within the next 6 years, by 2025 [17].

With all the ease and comfort IoT brings to us, there are some serious potential risks involved with it. IoT devices often store our data to help make decisions for ourselves but that data might be sensitive if it leaks out. Since there are billions of IoT devices out there, with our data being stored and processed on a number of such devices, with communication via internet or other potentially less secure networks, IoT are posing a huge security, privacy,

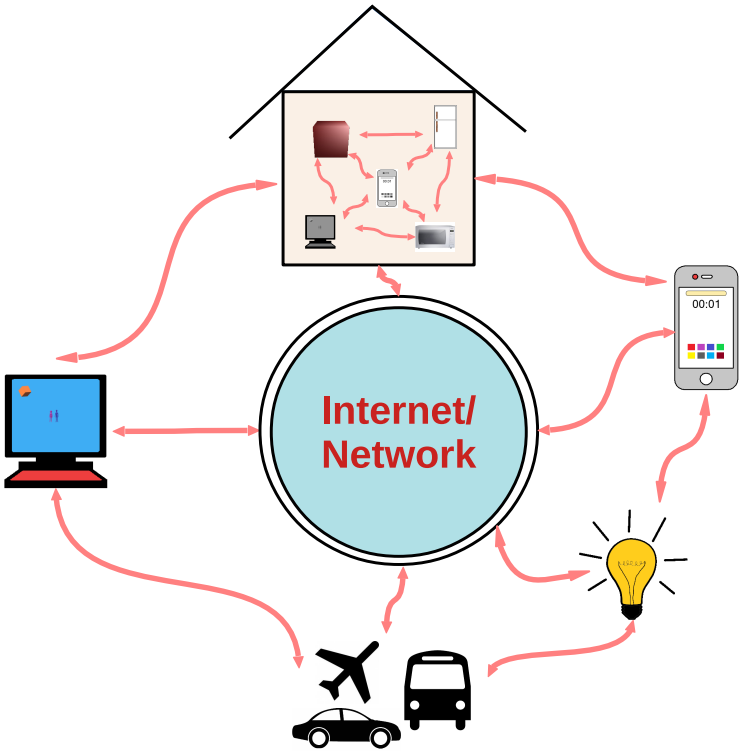


Figure 2.1: The Internet of things (IoT) [68]

and safety risks. Specifically smart medical implants and automotive devices can cause safety concerns while private data storage media and smartphones can create privacy risks. Among all these, security could be the main concern as because of its fast growing nature along with its very low cost production, security was not thought of as an essential feature. At the early stages of IoT, many argue that maybe these devices may not need security at all since the information they carry are so little or insignificant like someone's preferred room temperature or state of a fan or bulb. However, with the rapid increase of IoT, these devices find their way into many applications in our everyday life and process our data. Now that billions of these devices are everywhere with little to no security in them despite containing a huge amount of data, makes them the main target for an attacker. Therefore, governments and industries alike are concerned about the security of these devices and this is one of main research topics now.

One of big difficulties applying security mechanism is the resource constrained nature of such devices. Most of these devices are tiny with a very small silicon footprint, and

consuming a very little amount of power to run their operations. Some of these devices are employed in places where it requires them to strengthen their life as much as possible by draining as less energy as possible from their batteries. Pacemaker is one such example where you don't want it to run out of energy of often, requiring another tiresome and difficult heart operation very soon to replace that device. Many sensor devices are employed in field where they can be deployed for years and thus may depend on the harvested energy for most of their life. Embedded processors in such devices often employ aggressive energy saving techniques, going to a low power mode i.e. sleep or hibernation to save power. Employed embedded processors might spend most of their life in such sleep or hibernation to extend their battery life or to minimize the amount of power they harvest from different energy sources like solar, WiFi, wind etc. Therefore, The processor usually shuts off power to most of its components while using a little (or even zero) amount of energy to maintain its state.

Since IoT devices are often resource limited this way, this makes it very difficult to employ any rigorous security since they might require a good amount of silicon area and power that these devices can't provide. Any traditional cryptographic solution are thus often impractical for most devices in this domain. Lightweight cryptographic techniques are being developed but even those can be very resource heavy for a low power embedded processor or a small smart sensor. Moreover, employing the same security techniques with billions of such devices out there might make all of them vulnerable at once on the event of an attack, This are some of the main reasons why researchers are looking at alternative techniques like hardware security which would provide the required *practical* level of security, while complying with the limited resources of such devices.

2.2 Emerging Nanoscale Technology: Memristor

2.2.1 Emerging Devices

With the slow-down of Moore's law as we are approaching the physical limit of CMOS, researchers have developed a number of different post-CMOS devices. Among these, some technologies like NAND or NOR flash are more mature and have already found real-world

applications. However, there are a number of emerging devices especially nano-devices which are being researched with the goal of replacing conventional CMOS. International Technology Roadmap for Semiconductors (ITRS) has published and listed the most promising emerging technologies in their report in 2013 [2] and in 2015 [3]. These devices offer various advantages over traditional CMOS while but there is a need for a lot more research to overcome their limitations and use them effectively in current infrastructure.

Memory based on ferroelectric properties of matters have been known for many years and FeRAM (Ferroelectric RAM) is already being considered a mature technology. However, to overcome the shortcomings of FeRAM like destructive read and low read stability, other devices built from ferroelectric materials are being considered. Ferroelectric transistors or FEFET [3, 14] is one such device which is very similar to conventional except the oxide layer is replaced by a ferroelectric layer which helps to retain its state using residual polarization even when the power is disconnected. The polarization of the ferroelectric can be controlled by applying voltage at the metal gate contact. By applying voltage or electric field of sufficient magnitude, the magnetization in the ferroelectric layer is altered. The current or capacitance of the this ferroelectric FET shows a typical hysteresis loop if the applied voltage at the gate terminal is swept through a range. Even if the electric field is removed, a sufficient residual polarization remains in the ferroelectric and, therefore, this system can be used as a non-volatile 1-bit memory. Doped HfO_x is the most promising material that offers ferroeletricity. FEFET offers high speed, low power, non-volatility, full CMOS compatibility but has relatively high switching voltage and low endurance. Ferroelectric tunnel function or FTJ is another emerging device built from ferroelectric layer sandwiched between two metal layers which is ultra-thin and thus displays tunneling electroresistance (TER) even at room temperature for some complex ferroelectric oxides [2].

Phase change memory (PCM) [19] is another emerging device which displays resistive switching behavior. Its basic mechanism is Joule heating which transforms the internal state of a matter between crystalline and amorphous states. The resistivity of a PCM is high when the underlying material is in amorphous state and is low when it is in a crystalline state. Chalcogenide materials display this phase change property at relatively low temperature of around 600^0C. Both of these phases are stable at room temperature, thereby enabling them

to function as non-volatile memories. The physics of PCM is very well-understood compared to many other emerging devices. It offers high speed, high on/off ratio, long endurance but suffers from high switching current as well as limited scalability mainly due to the size of the access device required.

Spin-transfer-toque RAM or STT-RAM is another promising emerging memory technology based on traditional MRAM (magnetoresistive RAM). The memory cell of an MRAM consists of a magnetic tunnel junctions (MTJ) along with a regular MOSFET as the access transistor. A thin tunnel insulator, such as MgO (magnesium oxide) is sandwiched between two ferromagnetic layers to create an MTJ. One of the two layers has a fixed magnetic polarization, called as the 'fixed layer'. The other layer is called the 'free layer' and its magnetization can be easily rotated. When the polarization of these two magnetic layers are in the same direction i.e. parallel, the resistance of the cell is low and is considered a binary value of '1'. When the magnetic orientation is anti-parallel between these two layers, the resistance is higher. Write is performed by passing a current through both of these layers to change the magnetization of the free with respect to the fixed layer. The writing current and energy of a traditional MRAM is very high, thereby limiting their scalability. STT-RAM tries to improves upon this problem by using a low energy spin-torque action for write. It induces a spin-aligned current to control the magnetization of the free layer, reducing the current density compared to MRAM. MRAMs have very high switching speed, data retention, and endurance. However, STT-RAM suffers from small on/off resistance ratio and sensitive to fabrication process.

Redox (reduction-oxidation) based RAM or resistive RAM (RRAM), also known as memristors are a class of metal oxides which can display resistance switching among different states [19]. The basic structure of a ReAM is a thin oxide film between two metal electrodes. RRAM can be created from a variety of materials and composition. Perovskite oxides or binary oxides both can work as RRAM. CB-RAM or conductive-bridge RAMs are another class of RRAM where reative electrode supplies mobile ions to migrate across the dielectric to form a conducting path during on state. However, typically considered RRAM operate differently, by creating oxygen vacancies and in the oxide layer. They can be bipolar or non-polar (current in both or one direction for switching). There are even volatile memristors

which when programmed spontaneously reset back to its high resistance state. RRAM have fast switching speed, low energy, good endurance, retention time but suffers from reliability and larger process variation. Also, the device physics of RRAM is also not well-understood yet. Many RRAMs require an initial forming step where an initial high electric field is applied to initiate this switching behavior. Researchers are also trying to reduce this required forming voltage and create 'forming-free' RRAM which would reduce the circuit design complexity.

Besides these technologies, there are carbon based memory memory devices like carbon nanotubes and graphene based FET, Mott memory, molecular memory and so on [2]. All of these different technologies provide some advantages over existing techniques while showing some challenges in some others, thus there is a need for further research in device engineering, infrastructure building, and finding suitable applications.

2.2.2 Memristor

Introduction

Memristor or RRAM is one of the most promising nanoscale devices of the last decade or so that is being explored extensively due to its several advantages over traditional CMOS. Although the term could mean a number of different resistance switching devices as mentioned earlier, we would only mean metal-oxide memristors, specifically those from transition metal-oxides (TMO) throughout this book when we use this term. Memristors are usually non-volatile where the resistance of an insulating oxide can be altered between high and low resistances states via the application of an external field. The physics of metal-oxide memristors are different compared to its close counterpart like CBRAM (conductive-bridge RAM) or phase-change memory. In a memristor, the number of oxygen vacancies created inside the insulating dielectric oxide is the source of conduction. Memristors display typical hysteresis in their I-V curve as shown in Figure 2.2 below along with its common symbol notation. Researchers have shown the use of memristors in many different applications ranging from memory devices, logic circuits [61, 20], neuromorphic computing, hardware security [57, 26] and so on.

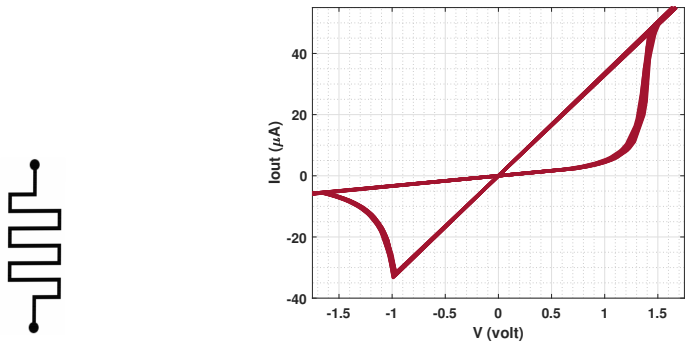


Figure 2.2: Memristor symbol (left) and its IV curve showing hysteresis (right)

History

The term 'memristor' is coined by Professor Dr. Leon Chua in his 1971 paper [15] when he predicted the presence of a fourth fundamental circuit element alongside resistor, capacitor, and inductor. He predicted this from the relationship among voltage, current, flux, and charge among the fundamental devices and later also projected some other characteristics of memristors or memristor-like devices [16].

The relationship between the flux linkage $\delta\phi$ and δq is expressed as the memristance (M), the property of a new fundamental circuit element, memristor. It can be implied that at a certain time, the memristance defines a linear relationship between voltage and current just like a resistor. Thus, memristor can actually be represented as a variable resistor. This relationships are shown in these equations below:

$$M = \frac{\Delta\phi}{\Delta q} \tag{2.1}$$

$$= \frac{\Delta\phi/\Delta t}{\Delta q/\Delta t} = \frac{\Delta v}{\Delta i} \tag{2.2}$$

$$\implies \Delta v = (M)\Delta i$$

Although, metal-oxide based switching devices are well-known for many years, memristors only have gained attention after a research team from Hewlett-Packard lab published a paper where the tie the characteristics of a TiO_2 MIM (metal-oxide-metal) device to that of an

Chua memristor in 2008 [63]. After that, memristors have gained a massive attention and a lot of research effort is ongoing for its device engineering, fabrication techniques, and a broad range of applications.

Memristor Characteristics

As mentioned earlier, we are considering transition metal-oxides which are non-volatile, and bi-polar. They show different resistive or memristive states i.e. memristors show hysteresis in their I-V characteristics as shown in Figure 2.2. Binary memristors have two stable resistive states, called the high resistance state (HRS) and the low resistance state (LRS). HRS and LRS are also known as OFF and ON states, respectively as well. Each of these two states can be reached by applying a voltage of appropriate magnitude and duration across a memristor. The minimum magnitude and duration of voltage applied across a memristor to reach LRS from HRS are called the set voltage or positive threshold voltage (V_{thp}), and positive switching time (t_{swp}), respectively. On the other hand, the minimum magnitude and duration of voltage required to go to HRS from LRS are called the reset voltage or negative threshold voltage (V_{thn}), and negative switching time (t_{swn}), respectively. These six parameters HRS, LRS, V_{thp}, V_{thn}, t_{swp}, and t_{swn} define the high level characteristics of a binary memristor.

Metal-oxide memristors are usually CMOS (complimentary metal-oxide semiconductor) compatible, can be fabricated on the back-end of the silicon together with CMOS on front. The size of a memristor is also very small (a few nanometers) which makes them an attractive choice for many applications.

The physics of memristors are not fully understood yet as memristive properties vary a lot depending on the type of oxide as well as the conductive materials used. The thickness and area of these materials, current limiting devices etc. control the memristive parameters. Thus different memristors reported in literature vary a quite a bit from each other in their resistive states, operating voltages and speed of operation. TiO_x, HfO_x, TaO_x are among of the most promising memristor devices because of their high on/off ratio, speed, retention time, endurance, and CMOS compatibility. Moreover, their switching process is stochastic and thus, memristors display cycle-to-cycle variation in their characteristics which makes it

challenging for circuit designers. Techniques are being explored to confine the characteristics of a memristor and make their switching operation much more consistent to reduce variability.

2.3 Hardware Security as IoT Security

Usually for any system, security choices can be divided into two broad categories, software and hardware. Software security usually refers to some algorithmic implementation of mathematically rigorous encryption or hash functions. They algorithms are usually very robust and hard to break, providing practically the best security for a system. However, these algorithms usually take many clock cycles for their implementation and thus also consume a larger amount of power. Therefore, implementing them for a small system where area, power and delay are very limited, can be impractical. As these systems have limited power budget, small in size and sometimes real-time in nature (and thus small delay), any security add-on must have very low overhead but these rigorous algorithms fail to fulfill that criteria. That's where hardware security comes.

2.3.1 Traditional Security: Cryptography

Cryptography is the technique of securing a data communication in the presence of an adversary. Usually using some kind of a key, cryptography allows a message to be encoded in such a way that only the intended receiver with the correct key would be able to decode the message. Cryptoghaphy has been used in human history for a long time including Caesar cipher, polyalphabetic cipher, Vigenére cipher etc. In the computer era, modern day cryptography is much more secure and based on rigorous mathematical analysis where a brute force attack would take an incredibly huge amount of time which is not possible in practice. Cryptography can be classified into two broad categories: symmetric key and public key cryptography. In a symmetric key cryptography, the same key is used for both encryption and decryption, easing the hardware implementation while being very secure. AES (advanced encryption standard), DES (data encryption standard), triple DES are examples of symmetric key cryptography. Public key or asymmetric key cryptography, on the other hand, relies on two keys: one public key and one private key. One key is known

to all and anyone can use to encrypt his/her message. The private key is only known to the intended receiver and only he/she can decrypt that message using this key. RSA (Rivest-Shamir-Adleman) is the most well-know public key cryotographic alogorith. While this is very secure, main disadvantage is the associated huge resource overhead. Thus while AES are used everywhere for secure data communication, RSA is only used for secure key exchange or digital signature.

Besides securing data communication, cryptography is also used to verify the authenticity of received data by means of a hash. A hash function is any algorithm can convert a large message to a fixed-length token of that message. Simple hash includes checksum, parity check etc. which can find bit-errors and thus verify the authenticity of the data. Cryotographic hash functions are one-way functions that fulfills some crytographic requirements like: pre-image and second pre-image resistance, collision resistance, despite the hash being easy to computer. The basic idea is that it should be very difficult to find two messages with the same hash given either the hash or the message and almost half of the bits of the hash should change if just one bit of the data is changed. Popular cryptographic hash algorithms include MD5, SHA-1, SHA-2, SHA-3 etc. All these hash algorithms are computationally expensive.

Algorithms like AES, RSA etc. are based on discrete mathematics with brute force attack resistance in billions of years of computing resources. However, when these algorithms are implemented on an embedded platform, the hardware can leak information like power consumption, delay in different stages of the algorithm runs. In fact, researchers have shown that these side-channels like power, timing, sound, EM (electromagnetic emission) can be used to perform so-called side-channel attack to break the security of cryptographic algorithms. For example, using simple (SPA) and differential power analysis (DPA) of side channel power, security of AES implemented on an embedded microcontroller can be broken [30, 31]. Fault injection is another very effective technique to attack otherwise secure cryptographic implementations [5]. Probing attacks [76] or EM attacks [1] can also be performed to read the stored secret keys to break a cryptographic algorithm as well.

Thus, although traditional cryptography is theoretically secure, their security can be compromised using the vulnerability of the underlying hardware implementations. Moreover, because these algorithms are mathematically rigorous, their resource requirement i.e. power,

area, and delay overhead are high. Therefore, researchers have worked on lightweight implementations of these algorithms as well as different new lightweight algorithms like SIMON, Piccolo, KATAN, PRESENT etc. [43] to tackle this issue. However, in the era of IoT, there are now billions of devices being produced each year and most of those are extremely resource limited, running on a battery or harvested energy alone. Therefore, traditional cryptographic algorithms would be either impractical or too resource hungry for these of types of devices.

2.3.2 Introduction to Hardware Security

We now know that even mathematically secure algorithms can be broken by exploiting the vulnerability in their hardware implementations. Moreover, IP piracy, reverse engineering, counterfeit etc. cause losses on the order of billions of dollars each year. Software based security solutions are not enough to prevent an adversary from copying a design and reproduce illegally. Therefore, researchers have been working on different hardware level modifications which improves the security of an IC and restrict overproduction, recycling, or counterfeit products on the market. Hardware security refers to any such technique that uses the hardware and modifies it to provide security solutions. Usually these techniques are specific to some particular security threat. For example, logic encryption and IC camouflaging techniques significantly reduce an adversary's ability to reverse-engineer a design, thereby mitigating piracy and counterfeit. Thus, hardware security provides innovative solutions to improve security of a device which may otherwise be vulnerable. Two very common hardware security primitives are physical unclonable function and true random number generator. They can generate hardware specific truly random keys that is otherwise impossible to generate from software alone.

2.3.3 Physical Unclonable Function (PUF)

Physically unclonable function or PUF is a hardware security primitive that exploit tiny variations among the chips with same functional implementation and use that variation to generate hardware specific signatures. These intrinsic variations originate from

uncontrollable manufacturing process variation and PUFs try to enlarge these variations to generate keys unique to that hardware only. Thus PUFs are usually represented as a challenge-response system where the challenge (C) is the input to the PUF that is used to extract process variation from a hardware and response (R) is the output key generated from that hardware using that PUF.

In this sense, PUF is analogous to a mathematical one-way function. One-way function is any function which is very easy to evaluate in one direction, but very hard to compute in reverse. For example, multiplication of two large prime numbers to get a composite number is easy, however factoring that composite number to get its prime factors is not straightforward and usually time-consuming. Thus applying a challenge to a PUF to generate its response is easy but determining the tiniest process variations that cause a particular PUF to generate a particular response is not very difficult to determine. Just like a software based security like encryption or hash algorithms use mathematical one-way function to implement the algorithms and thus ensure security, PUF based hardware security are essentially a low cost and natural way of implementing the same concept. Pappu *et.al.* first demonstrated a physical one-way function using variation in scattering pattern in [53] which later known as physical unclonable function or PUF. A PUF has to fulfill these requirements: (1) the PUF response must be raondom, (2) the process variation must not uncontrollable, and (3) PUF functionality or underlying process variation is unclonable.

Previous Relevant Works on PUF

Traditional cryptographic algorithms depend on some mathematical one-way function to provide security. However, most commonly used one-way functions are either based on unproven conjectures or have practical vulnerabilities in their implementations. To overcome these issues of algorithmic one-way functions, Pappu *et.al.* first proposed a physical one-way function (POWF) in 2001 [53] which relies on simultaneous multiple scattering from inhomogeneous structure to implement the one-way function.

Later in 2002, Gassend *et.al.* proposed a silicon physical random function and coined the term 'PUF' [21]. It was the first PUF implemented in silicon using electrical properties which is path delay in this case. Since then, many researchers have worked on many variations

and implementations of PUF. Arbiter PUF or APUF can be considered the first electrical PUF and the most well-studied PUF [34, 35], proposed in early few years after the POWF was established. APUF utilizes variation in propagation delay due to manufacturing process variation of two identically laid out circuits to implement PUF functionality [66]. Figure 2.3 shows a high level block diagram of an arbiter PUF. Here, two-input/two-output switches are chained together to create a circuit with two identical but separate path for a signal to propagate through. Process variations dictate that a single input signal will propagate through one path faster than the other. An arbiter or flip-flop is then used to compare that delay differences and a response is generated.

For an n-challenge arbiter PUF, each path consists of n such identically laid out two input/two output switches. Each switch has one selector input - one bit of the challenge. If the challenge bit is 0, then the path is straight-through and if the challenge bit is 1, the path is criss-crossed. Figure 2.4 (left two) shows the logical diagram of such a switch. This switch can be implemented easily by a pair of multiplexers (MUXes) as shown in Figure 2.4 (right). The two multiplexers share the same selector signal which is one bit of the challenge set. If it is 0, then input 'a' goes to output 'y1' and input 'b' to 'y2'. If it is an 1, then 'a' goes to 'y2' and 'b' goes to 'y1', thus implementing this switch.

At the end of the signal propagation, an arbiter is used to determine which path is faster compared to the other. A D latch or a D flip-flop can be used as the arbiter. The two arbiter PUF paths are connected to the 'D' and 'clock' input. If the first path ('D') is faster, the PUF response is 1, otherwise it is 0. Thus the arbiter converts the analog delay difference into digital signatures.

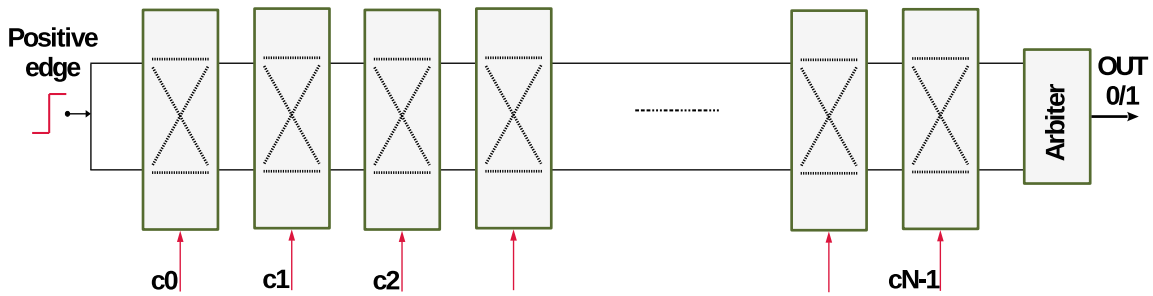


Figure 2.3: Arbiter PUF

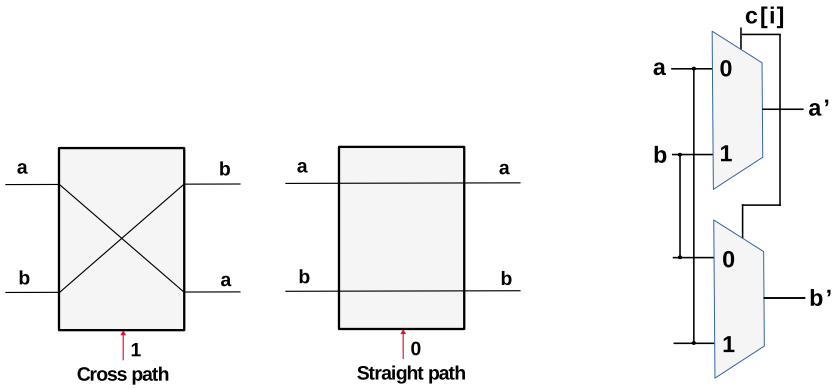


Figure 2.4: Two input/two output switch (left two) and its simple implementation with two MUXes (right)

Ring-oscillator PUF (RO-PUF) [65] is another very commonly used PUF, which can be easily implemented in both ASIC (Application Specific Integrated Circuit) and FPGA (Field Programmable Gate Array). A ring oscillator is a long chain of odd number of inverters connected in a ring such that the output oscillates between high and low voltage level, as shown in Figure 2.5. Thus a ring oscillator configuration can create a particular frequency where the clock period is sum of all individual delays of the inverters in the chain. Due to inherent process variation, the inverters would have slightly different delays and thus two identically laid out ring oscillators would have slightly different frequencies. A counter and comparator can be used to measure the difference between these two frequencies and convert into digital signatures. An RO-PUF is shown in Figure 2.6.

Both the arbiter PUF and RO-PUF are delay based PUFs since they use electrical delay as the property to implement PUF functionality. Any other electrical properties that are subject to uncontrollable process variation can be used to build a PUF. In an SRAM (static random access memory) PUF [22], random initial state of each SRAM cell during power-up is used to generate PUF signatures. In each SRAM cell, there is a cross-coupled inverter pair which are identical by design. During power-up when there are no externally exerted signal to an SRAM cell, due to slight voltage difference arising from parametric variation of these transistors would be amplified by cross-coupled inverter action and thus would show a tendency towards logic '1' (high) or '0' (low). Thus all the SRAM cells in a memory would give such random initial values during start-up which can be used as a signature of that chip.

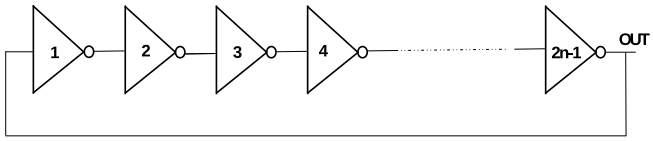


Figure 2.5: Ring oscillator: a chain of an odd number of inverters

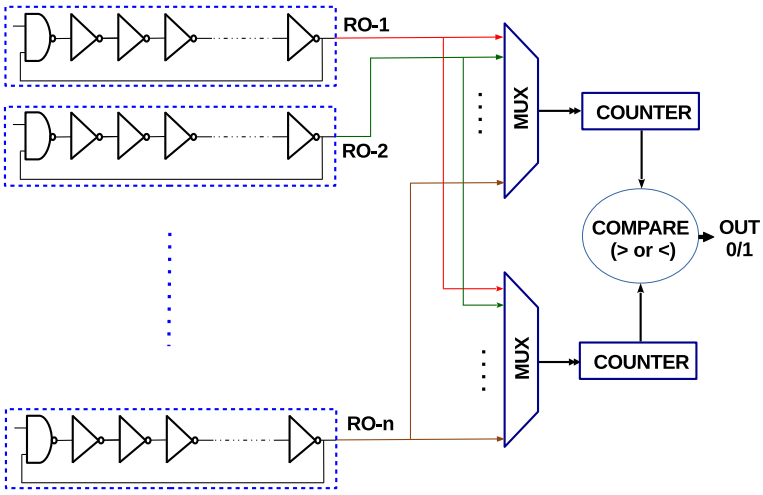


Figure 2.6: A ring oscillator PUF

Since CMOS scaling has become very slow in recent years, researchers have been looking for alternative technologies to provide application specific unique solutions. These emerging devices display switching variations and can be exploited to implement different memory based PUFs like memristor PUF [32], STT-MRAM (Spin transfer torque magnetic random access memory) PUF [83], PCM (Phase change memory) based re-configurable PUF [84] etc.

Memristor based PUF

A simple PUF based on the stochastic switching time of a memristor is first proposed in Rose *et.al.* in [60]. Memristors display stochastic switching behavior i.e. even if the same voltage, greater than its threshold voltage is applied to a memristor, it may or may not switch. First a mean switching time of a set of memristors are evaluated and then that a voltage is applied equal to the duration of that mean switching time to have a 50% switching probability. This is named as write-time based memristive PUF (WTMPUF) [59] which was later fabricated

and demonstrated experimentally in [45]. Kavehei *et.al.* demonstrated a PUF which use both memristors and ring oscillators, called the mrPUF [27].

Since WTMPUF uses absolute write-time of memristors as the source of entropy to implement a PUF, it could be subject to errors due to environmental changes. Since relative nature of measurement of APUF and RO-PUF make them robust against such environmental variations, a memristor PUF circuit built from a crossbar array of memristors where relative write-time of a pair of memristors are compared to generate a PUF response [57]. This memristor crossbar PUF or the XbarPUF [57] use a read-monitored-write approach developed in [42] to implement a PUF challenge and thus gradually nudge the memristors from one memristive state to another before one memristor column reaches to the other state faster than its adjacent column of memristors. Just like an arbiter PUF, this XbarPUF also uses an arbiter to determine which crossbar column of memristors were faster in its pair and as soon as read-monitored-write approach completes a complete write (transition from one state to another), a response is generated. Thus this PUF generates a complete response when all the crossbar column pairs are resolved. This XbarPUF shows a huge improvement in area in terms of transistor count compared to CMOS PUF (APUF) and its uniqueness and uniformity are very close to the ideal value of 50%. However, the biggest concern about PUF, the reliability metric wasn't reported in this work and data to generate these other results do not directly comply with any particular memristor. Therefore, my first work was to gather data from a HfO_2 memristor to build more a more realistic model and evaluate the XbarPUF performance using this real data.

PUF Performance Metrics

In order to gauge the degree of security provided by a PUF and to be able to fairly compare it with other PUF implementations, standardized performance metrics have been devised. Maiti *et al.* [39] and Hori *et al.* [23] have discussed several metrics to quantify a PUF's performance. The major six metrics are: uniqueness, reliability, bit-aliasing, uniformity, steadiness, and diffuseness. As shown in Figure 2.7, these metrics are used to quantify a PUF's performance across multiple device dimensions: inter-chip space, intra-chip space, and

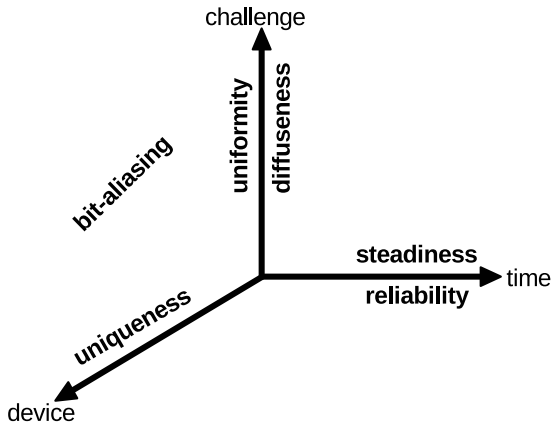


Figure 2.7: Standard metrics used for the evaluation of a PUF

time. They have become standards in hardware security research everywhere to quantify a PUF's performance.

1) Uniqueness: It is a measure of a PUF's ability to produce a unique ID in terms of its challenge-response pairs which are a specific function of its implementation on a given chip. To be able to efficiently distinguish every IC chip (from other equivalent ones) with a PUF circuit, the uniqueness must be tending to 50. That means, for a given challenge set, almost half the responses produced by two PUFs should be different from each other. Uniqueness or also known as inter-chip hamming distance is defined as:

$$Uniqueness(\%) = 100 * \frac{2}{N_{chips}.(N_{chips}-1)} \sum_{i=1}^{N_{chips}-1} \sum_{j=i+1}^{N_{chips}} r_i \oplus r_j, \tag{2.3}$$

for each response bit and each challenge set, where N_{chips} is the number of PUFs or chips to be measured, $resp_i$ and $resp_j$ are responses from the i-th and j-th chip respectively.

2) Uniformity: Another PUF performance metric which measures a PUF's ability to produce distinct responses across a set of challenges is the uniformity. With the flipping of even a single bit of the challenge, nearly half of the response bits are expected to flip. Effectively, uniformity is a measure of the ratio of 0's and 1's across the whole of the response set of the PUF. A PUF with poor uniformity would allow the attacker to reduce the possible response space and get a better prediction. This metric is defined as:

$$Uniformity(\%) = 100 * \frac{1}{N_{challenges}} \sum_{c=1}^{N_{challenges}} resp_c, \tag{2.4}$$

for each response bit and each chip, where $N_{challenges}$ is the number of challenges applied to a single PUF and r_c is the response for c-th challenge.

3) Reliability: Also represented as intra-chip hamming distance, this metric quantifies a PUF's consistency over time. If a PUF is unable to produce the same response every time for a given challenge, then the PUF is considered unreliable and may require error-correction. Reliability, as a metric is defined as:

$$Reliability(\%) = 100 - 100 * \frac{2}{N_{cycles}.(N_{cycles} - 1)} \sum_{t=1}^{N_{cycles}-1} \sum_{it=t+1}^{N_{cycles}} resp_t \oplus resp_{it}, \tag{2.5}$$

per response bit and chip, where N_{cycles} is the number of times of applying a challenge and measuring a response.

4) Bit-Aliasing: Different PUFs might produce similar responses for certain challenges. This would decrease the unpredictability of the PUF. Bit aliasing measures the average hamming distance for the k-th response bit of different PUFs. This metric is defined as:

$$BitAliasing_k(\%) = 100 * \frac{1}{N_{chips}} \sum_{m=1}^{N_{chips}} resp_{k,m}, \tag{2.6}$$

for each bit k of a response.

5) Diffuseness: The same PUF should generate different uncorrelated responses if different challenges are applied, especially for PUFs with large CRPs. Diffuseness measures this degree of different among different responses generated from a single PUF for different challenges. Diffuseness is evaluated by measuring the hamming distance among all different response vectors generated by a single chip and is defined below:

$$Diffuseness(\%) = \frac{1}{N_{bits}} \frac{4}{N_{chips}^2} \sum_{bit=0}^{N_{bits}} \sum_{i=1}^{N_{chips}-1} \sum_{j=i+1}^{N_{chips}} resp_{i,bit} \oplus resp_{j,bit} \tag{2.7}$$

6) Steadiness: When the same challenge is applied to a PUF, the response should be the same ideally. However, this is not the case for a practical PUF. Similar to the reliability metric, steadiness measures if there is any bias of a PUF response bit towards a particular binary value (0 or 1) over a certain number of evaluations. For N_{chal} different N_{bits}-bit PUF response evaluated over N_{cycles} times each, the overall steadiness is defined as [23]:

$$Steadiness(\%) = 100 * (1 + \frac{1}{N_{chal}N_{bits}} \sum_{i=1}^{N_{chal}} \sum_{bit=1}^{N_{bits}} log_2 max(p_{i,bit}, 1 - p_{i,bit})) \tag{2.8}$$

where the term $p_{i,j}$ represents the percentage of bias of a particular bit towards 1 over of a response vector evaluated many times using the same challenge. This is defined as:

$$p_{i,bit} = \frac{1}{N_{chal}} \sum_{t=1}^{N_{chal}} resp_{i,t,bit} \tag{2.9}$$

Another important metric when PUF is being used for authentication purposes is the probability of misidentification as defined in [64, 39]. Since we'll be using our PUF mainly as a key generator which requires more reliable PUF, this metric is omitted here and bit-error rate is measured instead.

Modeling Attack Resistance

Robustness against machine learning based modeling attack PUF provides an unique way of generating signature from the hardware itself utilizing manufacturing process variation. Although complex, PUF, especially delay based PUFs can be expressed as a sum of individual delays of many smaller circuit blocks along the path of the response generation unit of the PUF. Thus researchers have argued that any PUF is potentially vulnerable to machine learning based modeling attacks [62, 36, 78]. Specifically, an adversary can collect a subset of all the challenge-response pairs of a PUF and can build a numerical model using a machine learning algorithm to predict the binary outcome of the response bit(s). Rührmair *et al.* showed in [62] that an arbiter PUF (APUF) [66] and its variants could be broken with more than 99% accuracy using algorithms like logistic regression and evolution strategies. While different variants of APUF could take more time to train, they are still breakable with an increased number of CRPs. In this work, we have collected a small subset of all

possible CRPs of an XbarPUF and developed algorithms to perform machine learning based modeling attacks. Our implemented attack model is also found to be roughly 99% successful in predicting an XbarPUF outcome, although the modified XbarPUF architecture shows resistance against the same attack model with accuracy being around 50-60% only.

2.3.4 True Random Number Generator (TRNG)

Introduction to TRNG

TRNG is a very important hardware security primitive which is the only way a device can get truly random numbers as the name suggests. In most cryptographic applications, there is a need for secret key. Usually these keys are generated from the software using from some complex algorithms. However, software generated keys can be at best pseudo-random in nature and in some applications, there is needs for truly random numbers as well. TRNG can produce a stream of completely random bits, using some naturally occurring random phenomena. In our IoT security architecture, during each encryption operation, we need to provide the XbarPUF with a truly random key. TRNG usually harness randomness from a random and stochastic physical phenomena from the hardware or the circuit itself.

PRNG vs. TRNG

PRNG or pseudo random number generator produces a random sequence of numbers from the software itself. It usually requires an initial seed which dictates the output. If the seed is the same, PRNG produces the same sequence of numbers. Cryptographically secure TRNGs are useful for many cryptographic applications where a pseudo random number is good enough and also there is a need to regenerate the same sequence of random numbers. A TRNG, on the other hand, are not suitable for such applications because it is not possible to regenerate the same sequence of output. Thus PRNG and TRNG serve two different purposes and their application space is, therefore, different as well.

TRNG Sources

TRNG output must be free from the influence of environmental variation and process variation. The source of randomness or entropy for a TRNG should be a truly random phenomenon i.e. the probability or producing a '0' or '1' should be equal. Any temperature or supply voltage variation shouldn't affect the output. Process variation should also have a minimal effect as to have a better yield from the design as well. Many naturally stochastic phenomena like clock jitter, random telegraph noise (RTN), thermal noise, metastable state, quantum state etc. can be harnessed to build a TRNG. Figure 2.8 shows some of these sources that are available in an electrical circuit.

Memristor based TRNG

Memristor's switching is intrinsically a stochastic process. Researchers have thus explored different memristive devices too to build good TRNGs as found in literature [79, 26]. Due to probabilistic switching behavior, memrsitors display variation in its memristive states from from one clock cycle to another [52]. This variation is useful to build other hardware security primitives [79, 40, 71] as well. In [79], random distribution of memristors's high and low resistive states based on the switching time is used to design the MTRNG (memristor-based TRNG). Researchers have also utilized a differential readout scheme to harvest the random telegraph noise or RTN in a memristor along with LFSR (linear feedback shift register) to implement a TRNG in [28]. The differential nature of this circuit design improves resilience against temperature and supply voltage variation. Switching variability in memristor's set process is extracted to generate true random numbers in [4]. In [26], authors have designed an improved TRNG utilizing the random switching time of a diffusive memristor as the source of entropy and is shown to be able to produce true random numbers without any post-processing. This diffusive memristor is different than regular metal-oxide non-volatile memristors, does not require forming and reverts to its high resistance state from low resistance state spontaneously. However, this self-OFF switching ($\approx 1ms$) makes the switching processing slow and thus these memristors are volatile.

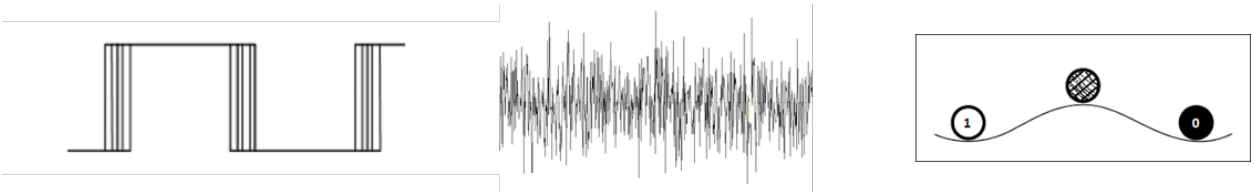


Figure 2.8: TRNG sources: clock jitter, noise, metastability

Chapter 3

Design and Analysis of Fundamental Hardware Security Components

3.1 Memristor Crossbar PUF

3.1.1 Introduction to HfO2 Memristor

The HfO_2 memristor technology considered for this research was designed and fabricated at the SUNY Polytechnic Institue, Center for Semiconductor Research (CSR) [8]. A cross-section of this device is shown in Figure 3.1. Memristors were integrated with IBM 65nm 10Lpe CMOS process on a 300mm wafer platform with CMOS at the front-end and memristive metal layers at the back-end. Memristors are places between metal-1 (M1) and metal-2 (M2) layers. Here, standard copper M1 layer was replaced by tungsten (W) layer because of its high boiling point to withstand high temperature during back-end CMOS process and an additional via (W-V1) was used between the memristive oxide layer and the M1 tungsten layer. M2 is still copper (Cu). A thin film of hafnium oxide (HfO_2) layer was deposited between these two metal layers using a precise atomic layer deposition (ALD) technique by using front-end-of-the-line (FEOL) tools before altering the composition of M2 and V1 layers. This metal-oxide is the active switching layer for the memristive device. A Ti (titanium) is just on top of the oxide layer, works as a oxygen-getter while an TiN (titanium

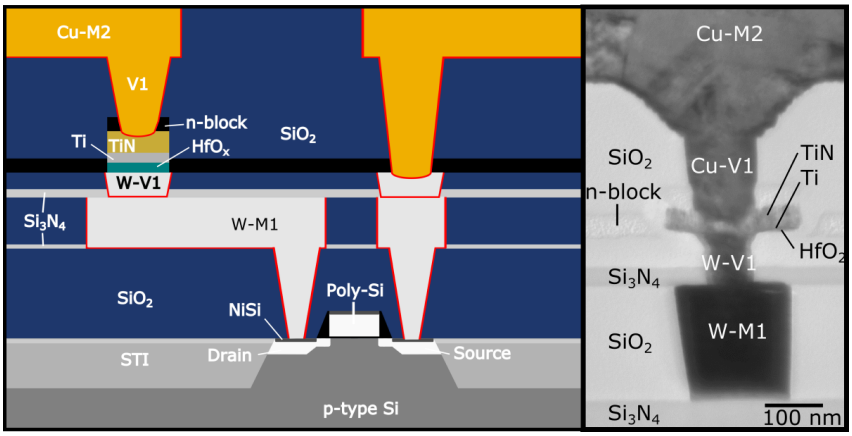


Figure 3.1: Illustration (left) and TEM (transmission electron microscopy) image (right) of a hafnium-oxide memristor embedded between the M1 and M2 layers. In addition, the Illustration depicts a seamless integrated 1 memristor 1 transistor (1M1T) structure where the transistor acts as the current limiting device [8, 69].

nitride) layer is used as the inert top electrode with via V2 and M2 layers (both Cu) work as the top contact.

3.1.2 Environmental Modeling of a HfO2 Memristor

Existing Model with Variability Taken into Account

The memristor model used for all the simulation works in this research was adapted from the model proposed by McDonald *et. al.* in [47]. Although this original model is more generic and could represent a wide variety of memristors, in our work, we are specifically concerned with bipolar memristor. Mostly binary metal oxide memristors are bipolar and although there are many different types of memristors, in throughout this work, we only mean bipolar binary or transition metal-oxide (TMO) when we use the word 'memristor'.

As discussed in last chapter, the binary memristor has two stable resistive states: LRS (low resistance state) and HRS (high resistance state). These two states can be reached by applying voltage greater than the corresponding positive or negative threshold voltages with duration greater than the corresponding positive or negative switching times. This model increments or decrements the memristance gradually when the applied voltage is greater than the threshold voltage. According to the model, when the voltage applied across a

memristor is greater than the positive threshold, the memristor starts to transition to LRS from HRS in each step using this equation:

$$M(t_{i+1}) = M(t_i) - \frac{\Delta r \Delta t \left| V(t_{i+1}) \right|}{t_{swp} V_{tp}}, \tag{3.1}$$

which is the set (HRS to LRS) operation. During the reset (LRS to HRS) operation, the memristance is changed using this equation:

$$M(t_{i+1}) = M(t_i) + \frac{\Delta r \Delta t \left| V(t_{i+1}) \right|}{t_{swn} V_{tn}}, \tag{3.2}$$

where $M_t i$ is the memristance at time instant t_i, $V(t)$ is the applied voltage across the memristor at time instant t_i, Δr is the absolute resistance difference between HRS and LRS, Δt is the simulation step-size, V_{tp} (V_{tn}) is the minimum threshold voltages needed to switch the memristor's state, where t_{swp} and t_{swn} are the minimum time for a full set or reset, respectively, when bias voltage is equal to the respective threshold voltage. Figure 3.2 shows IV plot of a HfO_2 memristor fabricated on the SUNY-Poly and a simulated IV plot using this model. It is worth noting that the bend at the top right side of the actual data plot in Figure 3.2 (left figure) is due to a current limiting transistor used during SET operation to prevent the device from destroying itself which is not used in simulation plot (right).

As can be seen in Figure 3.2, the IV plot doesn't follow a single path, but instead show a lot of variation from one cycle to another cycle because of the inherent stochastic switching mechanism of a memristor. Therefore, to have a realistic memristor model, this stochasticity must be incorporated in that model. This is done by considering the memristor's parameters as Gaussian random variable with mean and standard deviation coming from device statistics. We have used different sets of cycle-to-cycle variation (2%, 5%, 10% etc.) to incorporate these randomness into our model.

As an emerging technology, memristors also display a large amount of process variation from one device to another. The OFF resistance or HRS and the on resistance or the LRS are the parameters that show large variation [55] and we have considered larger standard deviation for them. Other parameters like threshold voltage, V_{tp} and V_{tn}, and switching times, t_{swp} and t_{swn} show smaller variation compared to resistive states. The mean and

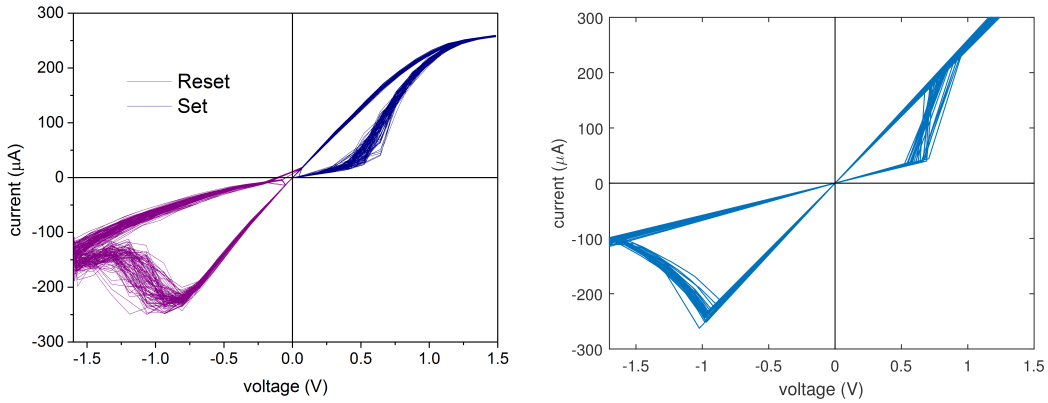


Figure 3.2: IV measurements of a memristor (left) and simulated IV characteristics using our memristor model (right) [69].

standard deviation considered for this work are listed in Table 3.1, taken from data generated at SUNY-Poly and also from literature [8, 55]. These values are not exhaustive and they depend heavily on a particular process for a particular memristor material and there are hundred sets of different values reported in literature.

Inclusion of Temperature Dependence

To realize any device in a electrical circuit, its dependence on temperature and other environmental variation must be taken into account to ensure reliable operations. For a transition metal-oxide, the LRS acts much like metal and shows positive temperature coefficient i.e. its resistance increases with increasing temperature [8]. The HRS, on the other hand behaves much like a semiconductor and thus its value decreases with increasing temperature [11]. Moreover, HRS decreases faster than LRS increases with temperature. Since HRS decreases and LRS increases with increasing temperature, the margin of separation between them decreases which may result in a reduced read/write margin in circuit level. Threshold voltage either change slowly with temperature [75] or do not show any noticeable change [11]. These changes are either linear or can be approximated as linear changes with temperature in the range (0^0C - 100^0C) of temperature that we considered for this work. Equation 3.3 shows this relationship [69].

Table 3.1: Mean and standard deviation for device level parameters of a HfO_2 memristor considered in this work [69]

Parameters	**HRS**	**LRS**	$\mathbf{V_{tp}}$	$\mathbf{V_{tn}}$	$\mathbf{t_{swp}}$	$\mathbf{t_{swn}}$
Mean	300KΩ	30KΩ	0.7V	-1.0V	1μs	1μs
Standard deviation	20%	10%	5%	5%	5%	5%

Table 3.2: Temperature coefficients for different memristor parameters [69]

α_{LRS}	α_{HRS}	α_{Vtp}	α_{Vtn}
0.004	-0.008	-0.001	0.0008

$$X_\theta = X_{ref}[1 + \alpha_X(T_\theta - T_{ref})], \tag{3.3}$$

where 'X' represents any of the four parameters: HRS, LRS, V_{tp} or V_{tn}, α_X is the temperature coefficient (per ^0C) of that parameter and 'T' represents temperature. Room temperature (300K) has been considered as the reference temperature. The values of temperature coefficients that we calculated using data from literature are listed in Table 3.2. The negative value of temperature coefficient indicates that parameter decreases with increasing temperature and higher absolute value of that parameter indicates stronger temperature dependence. Since we have found any data connecting the switching times with the temperature change directly and thus haven't included any dependence of switching time with temperature in our memristor switching model.

Inclusion of Aging

Another important device non-ideal characteristics is aging or limited endurance. Many different types of aging is observed in literature [55, 7, 9]. In case of worst case aging for memristors, both HRS and LRS display aging where distribution of HRS increases with time and LRS distribution increases with time. Thus the effect result of aging is reduced HRS/LRS or OFF/ON ratio. We approximated linear aging rate, with HRS being aged faster than LRS. This is demonstrated in Figure 3.3. It should be noted that this aging rate can also vary a lot across different processes and materials.

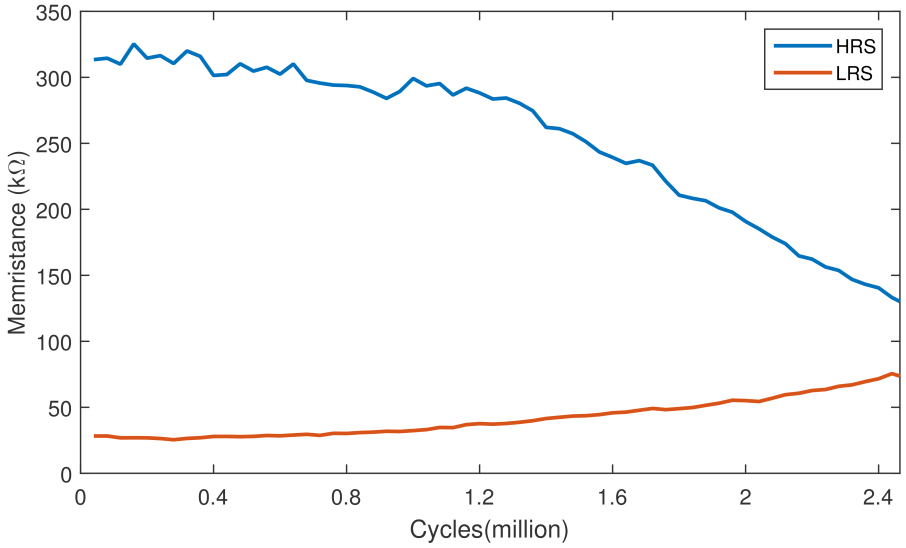


Figure 3.3: Simulated effect of (worst case) aging on both HRS and LRS of a memristor over time [69].

3.1.3 Working Principle of XbarPUF

The memristor crossbar PUF or the XbarPUF proposed in [57] uses a dense 2-D array of memristors to implement a strong PUF with potentially arbitrary number of challenges and responses. A circuit diagram of such an XbarPUF is shown in Figure 3.4 [72]. This is an updated version of XbarPUF from [57] with XORing included. In an XbarPUF, challenges are applied on the rows while responses are taken from the columns. For an N×M XbarPUF i.e. an XbarPUF with N challenges and M responses, its memristor crossbar size would be 2N×2M. For each challenge bit, there are two crossbar rows while for each response bit, there are two columns.

Before apply any challenges to the XbarPUF, a negative write voltage, V_{reset}, greater than the memristor's negative threshold voltage, V_{tn} is applied for a sufficient amount of time (at least more than t_{swn}) to reset all the memristors in the crossbar to HRS. Then depending on each challenge bit, a true challenge voltage is applied on each row while an inverted challenge voltage is applied on the corresponding adjacent row. If the challenge bit is high, then it drives all the memristors of that row towards LRS from HRS while the memristors with the inverted challenge remain at HRS. The opposite scenario happens if the challenge bit is low.

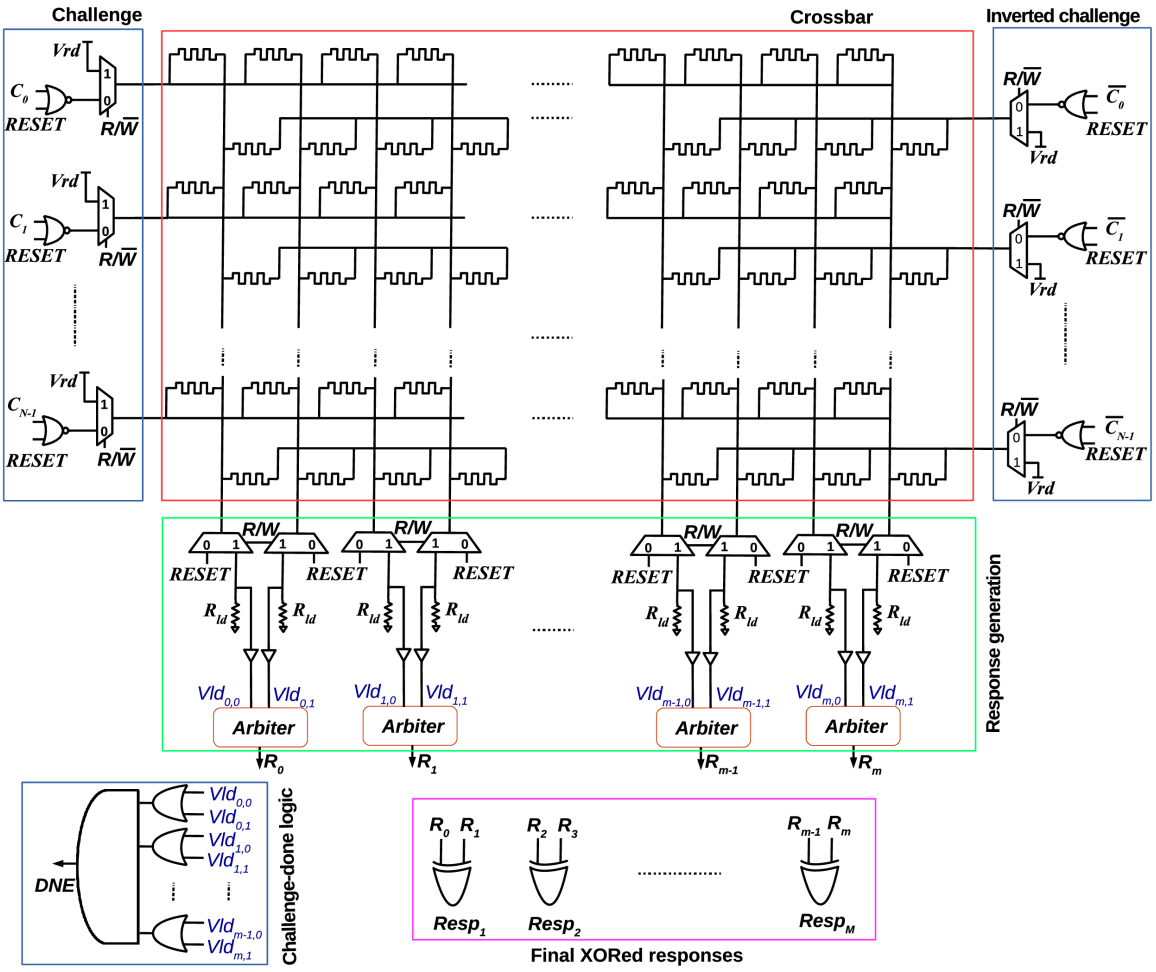


Figure 3.4: Schematic of a $N \times M$ XbarPUF circuit (with XORing) [72, 71].

The challenge or set voltage, V_{set} is at least as large as the positive threshold voltage, V_{tp} of the memristors in that crossbar. In a read-monitored-write approach, the challenge voltage is applied for a very short amount of time ($<<t_{swp}$) to nudge the memristors towards LRS. Then a small read voltage, V_{read}, less than the memristor's threshold voltages is applied to check which memristor reaches to LRS first between a pair of memristors in the same row and adjacent columns. As soon as one of these two memristors reach LRS, the load voltage of that particular memristor column goes to high, creating a low to high transition at one of the inputs of a D flip-flop attaching at the end of each pairs of crossbar columns. The arbiter or D flip-flop thus decides the winner and generates a response of '1' or '0' depending on which crossbar column in a particular pair was faster. When all the column pairs of the XbarPUF are resolved to generate valid responses, a 'Dne' (done) signal is generated to indicate the end of one PUF cycle. A load resistor of appropriate magnitude is connected at the end of each crossbar column in the read path while the load voltages of a pair of crossbar columns are connected to the arbiter in the read path as well. For an XORed XbarPUF, like in Figure 3.4, outputs from two arbiters are XORed to generate one final response thus effectively taking four crossbar columns to generate one bit response. As we will see in later chapters, XORing adds a non-linearity in the XbarPUF circuit which improves its robustness and also reliability as seen in [72].

We have evaluated and analyzed our XbarPUF by tweaking different circuit level and device level parameters to find the optimal optimal parameters for this circuit and to also discuss noise margin, scalability and security and overhead of the XbarPUF.

3.1.4 Clock Frequency and Other Parameter Selection

In this read-monitored-write approach [42] , we apply a 'read' voltage to evaluate the states of the memristor and a 'write' (challenge) voltage to push the memristors towards LRS from HRS. Thus the equivalent memristance of a column is decreased with each write pulse until it generates a sufficiently high voltage at the load resistance to determine a 'winner' by the arbiter. A write voltage with small pulse width i.e. of high frequency would help to change the memristance slowly and reliably from HRS to LRS. If the clock frequency is not high enough, then a single or a few write pulses might set both the memristors effectively at

the same time, making the response invalid. Therefore, we should choose a clock frequency which wouldn't create this situation while not being impractically high and that the circuit becomes too slow. For a memristor with 1μs switching time, a write pulse width of 100ns or lower or clock frequency of 10MHz should work and we choose this clock frequency for the remainder of this chapter.

The threshold voltages of memristors strongly dominate the choice of supply voltage as this voltage must be higher than the magnitude of both positive and negative threshold voltages of a memristor. Higher supply voltage contribute to power consumption quadratically and, therefore, it is chosen to be as low as possible. For example, for an HfO_2 memrisor with 0.7V and -1V threshold voltages, the supply voltage chosen is 1.2V.

3.1.5 Choice of Load Resistance

By performing a noise margin analysis, we have evaluated the optimum load resistance for our XbarPUF circuit. The load resistance should be chosen such that it maximizes the difference between two memristive states of a pair of crossbar columns. After RESET, all the memristors of the crossbar are in HRS while at the end of challenge phase, one half of the memristors are in HRS while other half change to LRS. The difference in load voltage during these two situation can be referred to as the noise margin (NM) for this circuit. Since the load resistance is in series with the memristive column, to reliably differentiate between logic '1' and logic '0', the load resistance has to be such that the voltage drop across the load is maximum just after reset (all HRS) and minimum just after challenge (half LRS, half HRS). For a crossbar with 'N' challenges (and thus 2N rows), the equivalent memristance in these two time (R_1 and R_2, respectively) are shown in equation 3.4 below.

$$R_1 = \frac{HRS}{2N}, R_2 = \frac{HRS}{N} \parallel \frac{LRS}{N}. \tag{3.4}$$

The load resistance is in series with either this R_1 or R_2 and the situation is illustrated in Figure 3.5. The corresponding voltage drop with both cases is shown in equations 3.5 and 3.6, respectively.

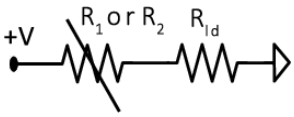


Figure 3.5: Two resistance in series, one is the equivalent memristance of a crossbar column and another is the load resistance [69].

$$V_{eq,R_1} = \frac{R_{ld}}{R_1 + R_{ld}} * V_{read} \tag{3.5}$$

$$V_{eq,R_2} = \frac{R_{ld}}{R_2 + R_{ld}} * V_{read} \tag{3.6}$$

, where R_{ld} is the series load resistance.

The difference between these two voltage levels, V_{eq,R_1} and V_{eq,R_2} is the **noise margin**, ΔV_R as shown in equation 3.7. In order to maximize this difference and thus to get optimal load resistance, we need to differentiate equation 3.7 with respect to load resistance, R_{ld} and set to zero. Then we find a closed-form expression (equation 3.8) for the best R_{ld} for this crossbar circuit.

$$\Delta V_R = V_{eq,R_2} - V_{eq,R_1} \tag{3.7}$$

$$R_{ld,best} = \sqrt{R_1.R_2} = \sqrt{\frac{HRS}{2N} * (\frac{HRS}{N} \parallel \frac{LRS}{N})} \tag{3.8}$$

3.1.6 Shift Back to Fixed Challenge Scheme

As can be seen in [57, 72, 69], challenges are applied for a very short period of time, much smaller than the actual switching time of a memristor, to nudge the memristance towards HRS or LRS depending on the magnitude and direction of applied challenge voltage. Then a small read voltage pulse is applied to read the memristance of both crossbar column memristors. This whole process is repeated until one of the columns reach to either HRS or LRS first and thus declared the winner. This is the so-called read-monitored-write approach [42] and is shown in Figure 3.6. This is very useful when there is little knowledge of actual memristor switching time and/or switching time has a very wide distributions.

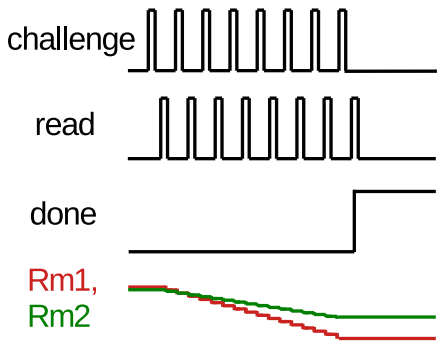


Figure 3.6: Read-monitored write approach to gradually check memristance of a pair of memristors until one finishes switching.

There are, however, practical problems associated with this approach when we try to implement. First, if the switching time of a memristor is very fast, then to perform read-monitored-write, it'd need a lot faster clock. For example for an TaO_x memristor, switching time can be around 100-150ps and thus, to perform read-monitored-write, the clock has to be as fast as 1-5ps which is impractical for a low cost system. Second, there is no prior knowledge how many cycles it takes for an XbarPUF to perform read-monitored-write once i.e. we can not determine the exact time when to sample the response. This asynchronous nature would make it very difficult to implement the PUF as an IP (intellectual property) within a bigger SoC (system-on-chip). Therefore, it is also impractical to use this approach in any processor based system.

To overcome these issues, we have returned to be the fixed-challenge approach used by Rose *et.al* in [60, 59]. Although, it requires prior knowledge of exact switching time of memristors, it makes the input-to-output sample time fixed and predictable and thus useful to incorporate in any system. The memristor switching time can be determined beforehand from fab or we can change the clock period to try with different frequencies and select whichever gives the most stable output. We choose the clock period to be around half the switching time of the memristor. Although, theoretically the clock period should work for any value from less than switching time to ultra-high frequencies, we choose this value which would give the most separation between the states of two memristive columns. Choosing a fixed low frequency could make both the column memristors reach to either LRS or HRS and thus eventually having little difference between them. Choosing very high frequency could

make the change in memristance very small and thus could be impractical to differentiate in a circuit. For our HfO_x memristor, where the switching time is on the range of 50-100ns, we have chosen the clock period to be 25ns.

3.1.7 Reverse Read Scheme

To improve the reliability of any circuit component, any opportunity of minor modification which doesn't add any overhead, should be considered. As measured in the lab in SUNY Poly Institute, HfO_2 memristor has positive (HRS to LRS) and negative (LRS to HRS) threshold of around 0.7V and -1.0V respectively. In our previous designs, we have been using a READ voltage of around 0.5V in positive threshold direction to read the state of a memristor. However, it is also found from experiment that memristors can actually switch or lose its state over 0.2V voltage applied across it.

Thus even if a voltage less than the threshold is applied across a memristor, the memristor can lose its state over time. To improve this situation by increasing the retention time of the memristor is to apply a READ voltage in a direction where the separation between READ voltage and memristor's threshold voltage in that direction is higher. Since for our HfO_2 memristor, negative threshold is larger in magnitude than positive threshold, it's better to apply read voltage in that direction. Thus we setup the memristors in the crossbar in such a way that the direction of RESET and READ voltage are the same. This is called reverse read scheme since read voltage is actually in reverse compared to the positive voltage direction.

3.1.8 Sense Amplifier

For read-monitored-write approach a series of inverters or a buffer is used to sample the load voltages of each memristive crossbar columns. For that approach, whenever one of the column reaches a load voltage of at least VDD/2, the buffer starts to operate and rises to VDD first. Both column buffers are connected with an arbiter (i.e. a D latch). The arbiter outputs a '1' or a '0' depending on whichever crossbar column rises to VDD first.

Our new PUF won't work this way now that the read voltage is only 0.4V and, therefore, the buffers would never get VDD/2 from load resistances and thus won't be able to produce

a rising edge to arbiter input. A READ voltage of 0.4V also means the absolute difference between two column load resistances is also small and more precise measurement is needed. For that purpose, we have replaced the system of buffer chain and an arbiter with a sense amplifier. A voltage latched sense amplifier (VLSA) is used instead which takes the two column load voltages as its two bit-line inputs. It then produces a full-swing output depending on whichever column voltage is higher. Additional buffers may be added to these outputs for separation. Our implemented sense amplifier is able to differentiate voltage differences of around 1mV and thus increase stability.

Since the magnitude of reset voltage is usually greater than the magnitude of set voltage, with a fixed read voltage, the error probability, due to unintended set or reset during read is lower if we the direction of read voltage is in the same direction as reset.

3.2 Specialized Peripherals for Memristor based Circuits

3.2.1 Current Compliance

The SET operation of a memristor is destructive. As described earlier, during SET, the memristance of a memristor starts to decrease towards LRS if the the magnitude and duration of applied voltage are greater than memristor's threshold voltage and switching time, respectively. However, the problem is that if the applied voltage pulse is not removed, the memristance continues to reduce even after reaching LRS and large current flows through the device and thus it could destroy itself completely. To prevent this from happening, an upper limit of current must be imposed in the SET path of the memristor. Usually an NMOS transistor could be used with a reference gate voltage to implement current-limiting condition. The RESET circuity, however, does not need such current limiting condition, because memristance increases and thus current decreases during RESET and it stops at HRS naturally. Thus there is an asymmetry between SET and RESET paths of a memristor. We, therefore, separate out these two paths in our implemented PUF along with tunable

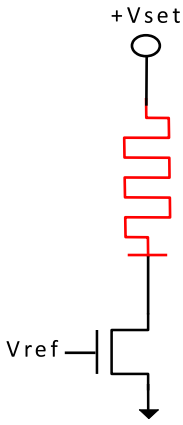


Figure 3.7: An NMOS transistor with appropriate gate voltage to control the current through a memristor during set.

current compliance in the SET direction. The SET current compliance circuitry is shown in Figure 3.7.

3.2.2 Forming Circuitry

Many memristors require an forming step before they can be used as a multi-level resistor i.e. as a memristor. This forming step requires applying a voltage of much higher magnitude to perform the first SET of the memristor. Without forming, the memristor acts like a resistor with resistance on the range of MΩ. For our HfO_x memristor, forming voltage is ≈3.3V whereas SET and RESET voltages are ≈1V. The regular MOSFET of IBM 65nm can only sustain voltages up to 1.4V. Therefore, a different and high voltage transistors are needed for the forming circuitry. We have used DGXFET for this purpose. These DGXFETs have high resistance and, therefore, can not be used in circuits where the VDD is near 1V. Since forming is another special SET operation of the memristor, it also requires current compliance control. Thus forming necessitates another separate path for a memristor circuit. The current compliance transistor should be a DGXFET in this case to handle high voltage (≈3.3V). The forming circuitry is shown in Figure 3.8. A current compliance transistor is needed as well during the forming to limit the current and set the memristor to a good high LRS value. This forming path is only used once and then never used again. Therefore, multiple memristors can be formed together by sharing this forming circuitry

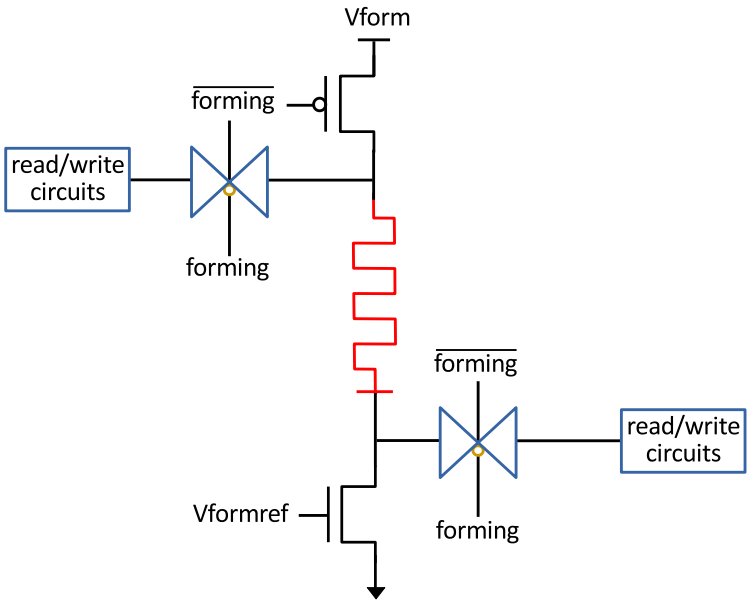


Figure 3.8: Forming circuitry of a HfO_2 memristor with 65nm CMOS technology

which would reduce the area overhead. Moreover, if the forming voltage can be reduced further so that similar transistors can be used for both regular read-write and form, then they all can share the same control circuitry and significantly reduce the design effort and resource overhead. Researchers are working with forming-free memristors to totally eliminate this design overhead.

3.2.3 Memristor Read-Write-Form Circuitry

Circuit Design

We have implemented a memristor read-write-form circuitry to ensure that the memristor can be formed correctly and it can be written and read with confidence. The schematic and layout for this read-write-form circuitry are shown in Figure 3.9 and 3.10, respectively. The transistor widths are chosen to reduce voltage drop across them and to facilitate enough current to flow through the memristor. Forming path requires higher voltage (3.3V in this technology), uses high voltage DGX transistors and thus forming path is separate from regular 1.2V transistors to prevent any damage to these transistors. A current compliance transistor is used in both forming and set path to prevent overflow of current through this

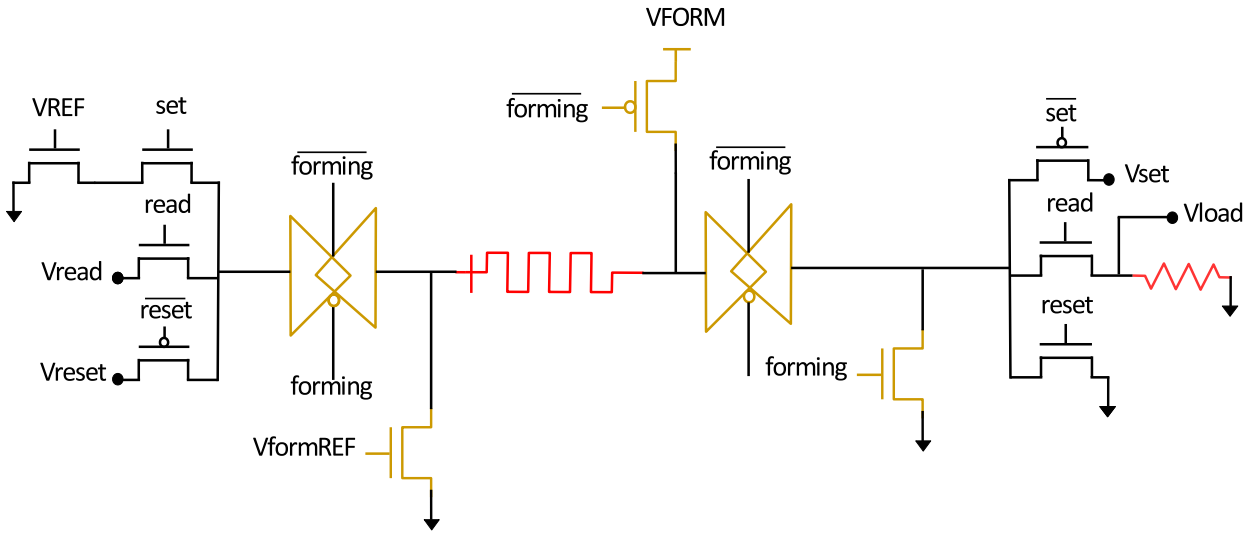


Figure 3.9: Schematic of the the read, write and forming circuitry for the HfO_2 memristor used in this work

memristor. In this circuit, the gate voltages of these compliance transistors are chosen in such a way that a maximum of 100μA current can flow the memristor, which is the safe limit of operation for these memristors. This read-write-form is an essential part of the memristor PUF circuit.

On-Chip Testing

We have designed and done layout of this memristor read, write, and form circuitry as a stand-alone test structure with 65nm CMOS technology. We have initially have performed forming test which is vital for proper operation of a memristor based circuit.

Details about the test setup can be found in Appendix D. The chip lies inside a probe station and circuit connections can be established by connecting probe pads to the test structures in the chip. Using a PSOC microcontroller, we have applied different control input voltages to this test structure while we have used a source meter to apply precise forming voltage and current compliance control voltage to this circuit. We are able to successfully form several memristors and the I-V plots are shown in Figure 3.11. Sudden spikes in current reflect the situation when the forming occurs. Figure 3.12 presents this more clearly as it shows the current measured through the memristor before and after forming is

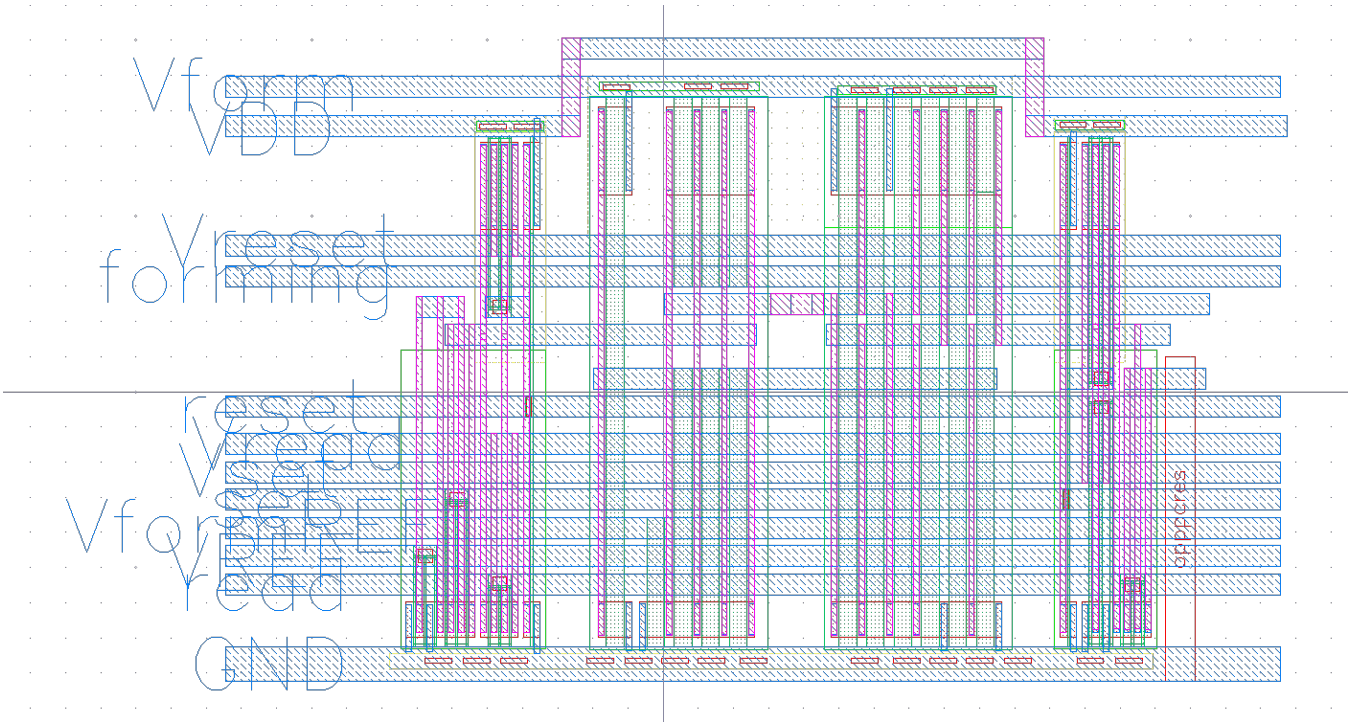


Figure 3.10: Layout of the designed read, write and forming circuitry for the HfO_2 memristor (18.31μm $\times$ 22.60μm)

performed using the read control available in the circuit. Before forming, memristor acts as an insulator which is confirmed by the very small current and very high resistance of the left plot in Figure 3.12. After forming, the memristor sets to its LRS for the first time and displays a much larger current as the plot on the right confirms. For this particular memristor, the resistance measured before and after forming is found to be on the range of 12MΩ and 12KΩ, respectively.

I have also performed some additional small experiments to test the functionality and characteristics of these memristors. Figure 3.13 shows a successful reset of a memristor which verifies the ability to reset using our designed circuit. From the figure, the reset voltage can be estimated as -0.5V. Figure 3.14 shows forming current vs. reference voltage for the current compliance transistor during the forming step. By changing this forming voltage, the forming current can be controlled to set the memristor at different resistance levels. The linear relationship indicates the ability to control this forming step very precisely by controlling this reference voltage.

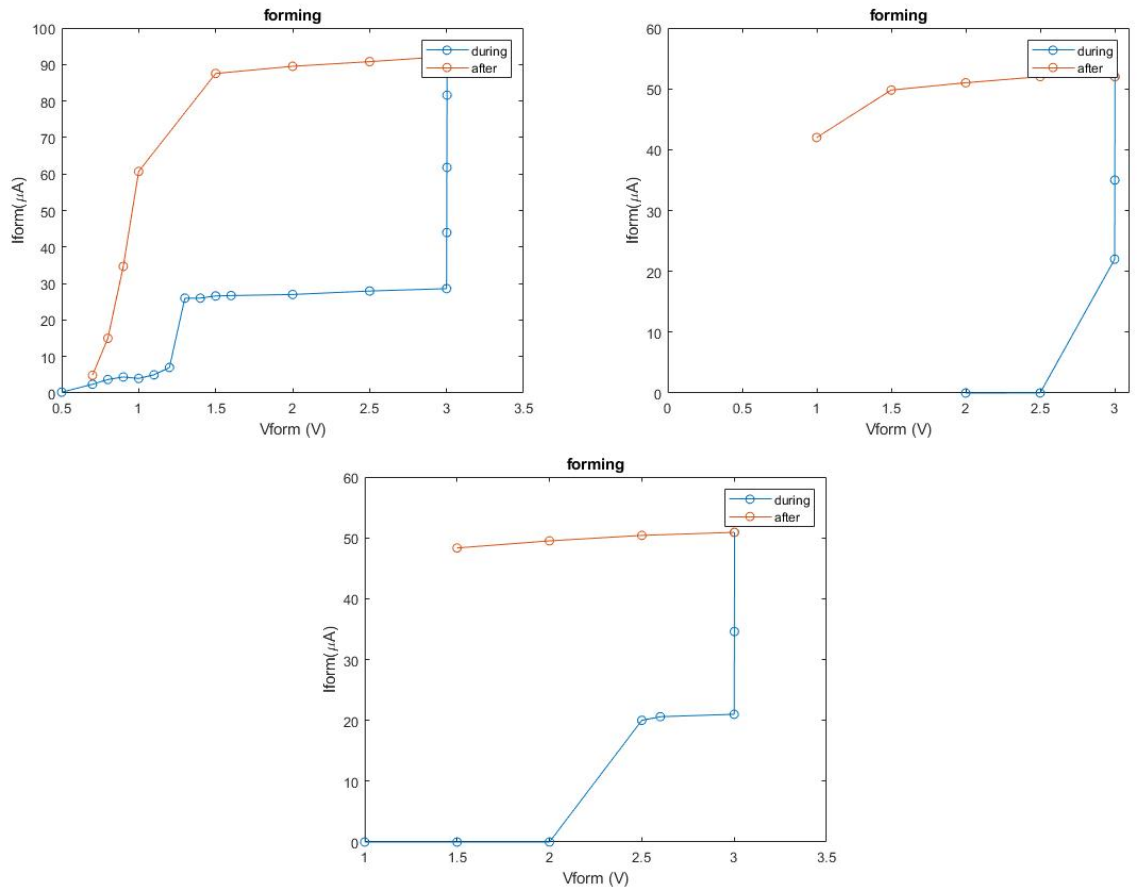


Figure 3.11: Successful forming of three different memristors with our designed memristor read-write-form circuitry. The forming can be easily identified by the presence of a sudden increase in current in the IV plots. The gate voltage to control the current compliance transistor is also varied to demonstrate forming at different levels which is why there are multiple forming-like spikes in these plots

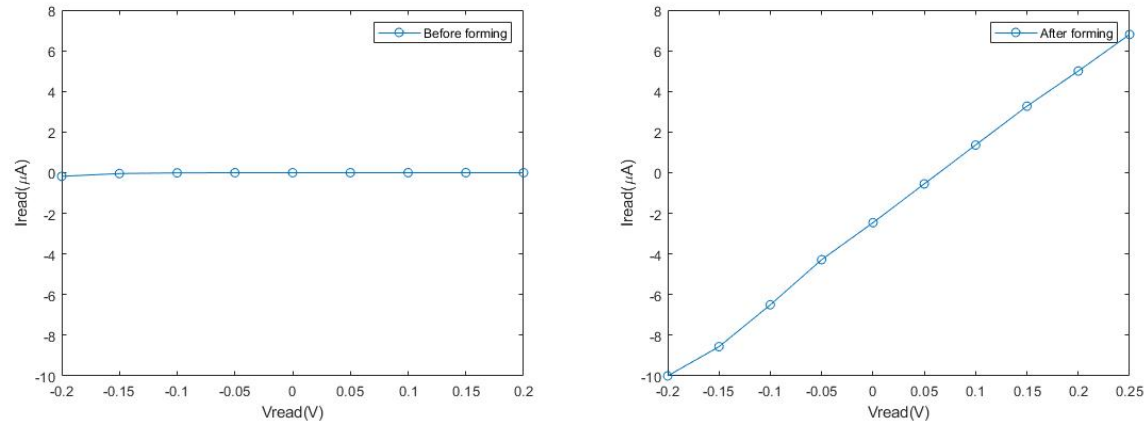


Figure 3.12: The current before (left figure) and after forming (right figure) is performed. The current is very low (∼pA) before forming but was high (∼50-70μA) after forming.

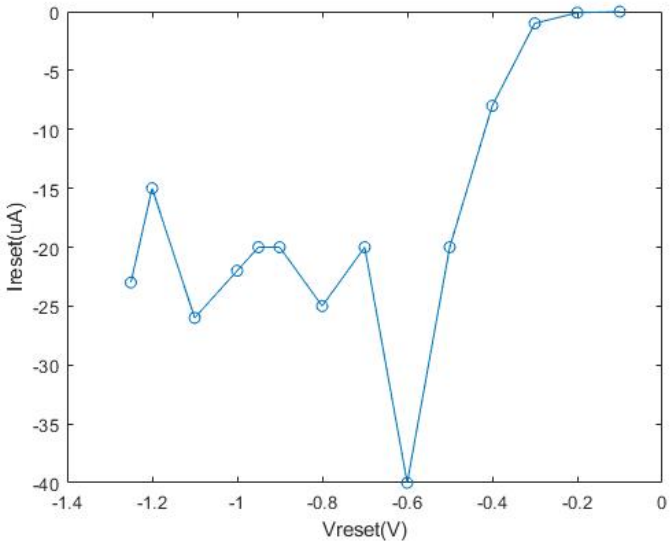


Figure 3.13: This plot shows a successful reset of a memristor with our read-write-form circuitry. The plot clearly shows how current increases linearly with increasing reset voltage voltage i.e. there is no change in memristance or resistance. After when reset voltage (-0.5V) is reached, memristors finally resets to a high resistance where there is a sudden and noticeable drop in current.

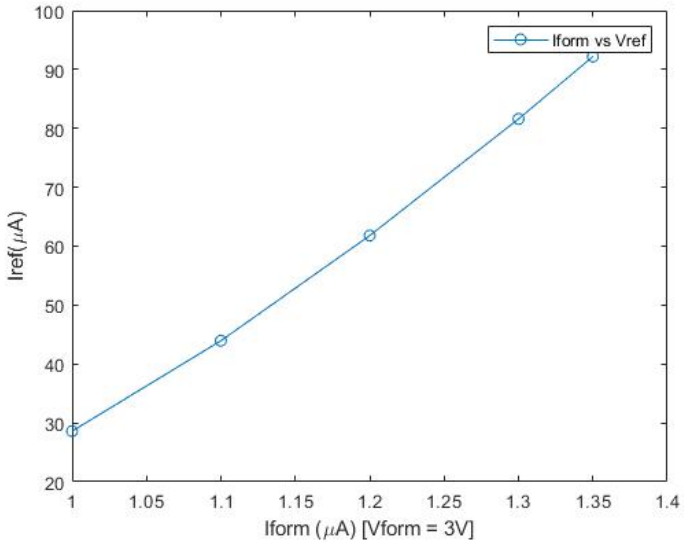


Figure 3.14: Plot shows the stable current through the memristor when formed using several different current compliance control voltage (Vref). These several different current levels prove the ability to form these memristors at different LRS values with confidence.

3.2.4 Sense Amplifier for Memristor Crossbar Circuits

Sense amplifier (SA) is one of the essential components of fast and dense memory architectures. A sense amplifier can sense or detect small voltage difference between its input voltages and produce a rail-to-rail output. The input voltage with higher value goes up to VDD (supply voltage) and the other input voltage goes down to GND (ground). In case of memory circuits, SA detects small difference between two bit-line voltages and produces a full digital output during a read operation. As we have mentioned in previous sections that we have shifted away from read-monitored-write approach and now the XbarPUF response is determined by comparing the voltage difference at the load due to the memristor's HRS/LRS variation, not the switching speed. Arbiter is thus no longer useful at making this decision and we need some kind of a voltage comparator instead. Although our XbarPUF isn't exactly a memory circuit, but it has a crossbar of memory elements i.e. memristors and can help measure the load voltage difference between each pair of crossbar columns during read operation. Due to unavoidable manufacturing process variation, memristors display variations in their memristive states and other electrical properties. Therefore, two memristor crossbar columns would have slightly different equivalent memristancs and thus the voltage drop at the load resistor, attached at the end of each crossbar column, would be slightly different as well. The SA would be able to measure these tiniest differences to provide a digital response for the XbarPUF.

Initially we have looked in the literature for established SA circuit designs. First, we have narrowed our design topology choices based on the strict design requirements for our designed XbarPUF. Then we have analyzed representative circuits of those selected topology in terms of performance, reliability of operation, and overhead. We have picked a topology which is best compatible with our design requirements. Then we have worked on that topology to suit to our design needs. We have also performed Monte Carlo analysis to choose optimal or near optimal set of circuit parameters e.g. lengths and widths of different transistors, supply voltage in terms of speed, energy, reliability, and yield. This also helps us to gain insight about the quality of our design in a real hardware.

Previous Sense Amplifier Design

We have considered several SA design topology and built and simulated their representative circuits to make a fair comparison among them. We have analyzed each of these circuits using our design parameters to understand their strengths and weaknesses with respect to our XbarPUF application. The two primary broad classes of SA topology are current-mode and voltage-mode [49] while hundreds of their variations exist in literature for different applications. By doing some design and parameter tuning, we expect some of these designs to meet our design requirements.

Current latched SA or CLSA have been widely used in many memory designs for their reliability and speed of operation [29]. A CLSA circuit can detect voltage difference between two pre-charged bit-lines of a memory. The input voltages are connected to the gate inputs of a pair of differential NMOS transistors and depending on which of the input voltage is lower, that branch discharges to GND while the other remains at VDD. CLAS requires the bit-lines to be pre-charged (close to VDD) to ensure their transistors operate in saturation region. Thus their speed of operation of a CLSA decreases significantly with decreasing supply voltage and when the bit-lines are not pre-charged, thus making the transistors operate out of saturation region.

Voltage latched SA or VLSA are another category of popular SA topology [49]. In most typical memory applications (e.g. SRAM), CLSA circuits usually outperform or on par with VLSA circuits. However, VLSA design is more resilient against threshold voltage or V_t mismatch and transistor size or β mismatch compared to most other SA designs. Similar to CLSA, the performance of a VLSA circuit decreases when the input bit-lines are not pre-charged and close to GND.

Researchers have proposed offset cancellation techniques to improve the resolution of SA circuits [33, 13]. However, some of these techniques involve adding capacitors which leads to a significant increase in area and not suitable for very lightweight application. Plus capacitors have mismatch problems themselves.

Researchers have also designed SA circuits suitable for other emerging devices like MTJ or magnetic tunnel junction based memory [85, 82]. They have worked on making their SA

circuit, named as pre-charge SA or PCSA [85] suitable for MTJ based circuits specifically. Their design is fast despite being simple which is what we need for our application here too. However, unlike MTJ, memristors have a hard limit on the allowable read voltage being applied across them as higher voltage increase the chance of unintentional switching. PCSA also does not provide full swing output unless additional buffer stages are employed. The design also looks prone to be affected heavily by mismatch variation.

We have considered a few other popular and promising SA designs as well. However, as we will see, due to the strict requirements imposed by memristor based circuits, only a handful of SA designs could be considered promising for our application. We have built those circuits alongside ours in Cadence Virtuoso with 65nm technology and compared against each other to decide on the final design.

Design Requirements

Memristors' read voltage should be low enough so that the applied voltage doesn't cause a switch. The suggested voltage for a reliable read of HfO_2 memristors considered for our work (fabricated at SUNY-Poly [8] is only 0.2V. With sufficient time, the memristor device might switch to either LRS or HRS if the read voltage is greater than 0.2V depending on its polarity. In our XbarPUF circuit, the read voltage is chosen to be 0.4V with careful choice of load resistance so that the voltage drop across a memristor never exceeds 0.2V. Another thing to consider is the variation of the load resistance themselves, especially since they are on-chip and thus have high variation. For example, n-well resistors of the 65nm technology that we have used can have 20% mismatch. Thus if their mismatch dominates over the mismatch of the memristor crossbar itself, then it would result in a very unreliable read of the memristive states. Thus we have to ensure a very low mismatch between two load resistors in a pair of crossbar columns and should choose their value so that they can translate the variation in memristive states into a noticeable load voltage difference between each pairs of crossbar columns. These two load voltages would be the input bit-line voltages to each SA. Lowering the supply voltage than read voltage further makes the circuit slower, leaky and thus not useful. For our particular XbarPUF circuits [69], the load voltage inputs to the SA would be on the range of 100-300mV. Memristor's switching speed also determine

the clock frequency of the XbarPUF as well as the SA while also setting the upper limit on acceptable delay. In this particular work, clock frequency of 2μs is used. While trying with different supply voltage levels, only available on-chip voltages are considered i.e. which would be already available for an XbarPUF. The area and energy overhead of the SA should be kept as low as possible since for each response bit of an XbarPUF we need an SA and the overall design should be as lightweight as possible. All of these are hard requirements and they limit and shape our choice for a SA design. Thus we need a SA that can work reasonably fast with high resolution and low overhead despite having low or near-to-GND input voltages.

There are several things to consider to improve an SA circuit. Any mismatch among the transistors in an SA should be negligible compared to the expected difference between the two differential input voltages. For example, the impact of V_t and β mismatch between drive transistors should be minimized. These two mismatches affect CLSA designs more than VLSA designs as found in literature [49]. Another thing to consider is that the sensing speed would be slower if the bit-line voltages are applied directly to the gates of transistors since near-to-GND gate voltages would drive those transistors out of saturation. On another note, because each memristor has a very low capacitance while having large resistance, the whole memristor crossbar can be considered a big resistive network with load resistors at the end of each column. Therefore, the inputs to the SA can be considered to come from a resistor division circuit with only wire capacitance. We have considered all of these while choosing and designing our own SA circuit.

Our Sense Amplifier Design

As we just mentioned earlier in this section, the factors behind selecting and designing our particular SA are a low overhead design with fast operating speed, with input bit-line voltages near to GND, capable of differentiating input voltage pair with small differences. The available choice of clock frequencies and supply voltages are also limited. As we discovered, VLSA topology based designs work better with small (sub-threshold) VDD and when bit-line voltages are close to GND, rather than VDD. Thus we choose our own SA design to be based on VLSA. Researchers have also shown that the effect of V_t and β mismatch is also

low in VLSA compared to CLSA [49, 81] while this effect can be reduced further easily by increasing the transistor size. Lower mismatch is very important to have a higher resolution sensing capability. Therefore, we have designed our SA based on VLSA while also modifying it further to meet our design requirements. The schematic of our designed SA is shown in Figure 3.15.

Four transistors, M5-M8, in Figure 3.15 are access transistors which allow the input voltages to be latched into the internal nodes of the circuit. Since the inputs are applied at the diffusion terminals of these four transistors, this SA circuit offers low input resistance. These two inputs are voltage drops across load resistors of a pair of XbarPUF columns. The equivalent memristance of each column and the load resistor act as a voltage divider and this voltage is one input to the SA. Therefore, it is important to remember that the inputs do not have very high drive strength. The four transistors, M1-M4, form a cross-coupled inverters which decide the final state of the SA with their positive feedback action. Their high voltage is connected to the sense enable signal 'SE', instead of VDD. A constant positive voltage, VDD can affect and change the internal nodes 'O' and 'Ob' by leaking voltage through the PMOS transistors, M1 and M2 when the input voltages are sampled through the access transistors. Therefore, this is a very modification that a digital signal is used as the power source for this circuit. When the SA is in sampling phase, the 'SE' signal is low, meaning the power supply is disconnected to allow proper latching or sampling of inputs into the internal nodes. No current path is formed through M1 and M2 and thus they do not interfere with the sensing nor change the latched voltages at nodes 'O' and 'Ob'. During sensing, 'SE' rises to VDD or logic '1' and act as the power source for resolving these two nodes. This SA is used at the end of each pair of XbarPUF columns to help determine which column has higher voltage drop across its load resistor to generate a one-bit response as shown in Figure 3.16. The equivalent input-output circuit for each SA in the XbarPUF is shown in Figure 3.17 when the XbarPUF is in read or response generation phase. As mentioned earlier, the voltage division formed by the load resistor and equivalent memristors in two adjacent columns in the crossbar provide the two inputs to each SA. SA then compares these voltages and generate a full-swing output of either logic '1' or '0' depending on which of the

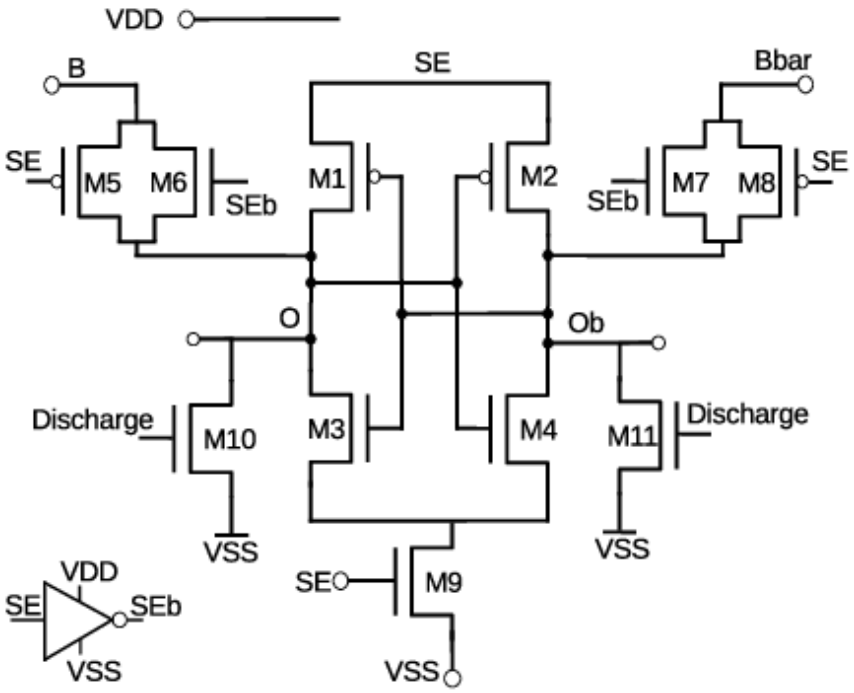


Figure 3.15: Schematic diagram of the our designed sense amplifier circuit suitable for memristor based crossbar architectures [73]

two column load voltages is higher. Buffers can be added to isolate these internal output nodes from other parts of the circuit.

If the magnitude of the supply voltage used is low, latching delay of inputs to internal nodes of our SA becomes large and thus can dominate over the sensing delay. We have used both NMOS and PMOS as transmission gates to work as the access transistors for this SA. This increases the range of analog input voltage that can be reliably sampled or latched into the SA nodes. However, since we know our input voltages would be close to GND, the NMOS transistors are made stronger than PMOS ones.

In Figure 3.15, there are two additional (and optional) transistors, M10 and M11 which we call discharge transistors. After a sensing operation, these two transistors can help to discharge nodes 'O' and 'Ob' so what the result from one cycle doesn't interfere the latching operation of the next cycle, thereby potentially affecting the outcome on that cycle. However, a PUF circuit is usually a part of a security block that we only need to use once or twice during start-up of a chip. Also we can also discharge the internal nodes 'O' and 'Ob' by latching zero-valued inputs to reset them (zero-ing the nodes). Therefore, for our XbarPUF

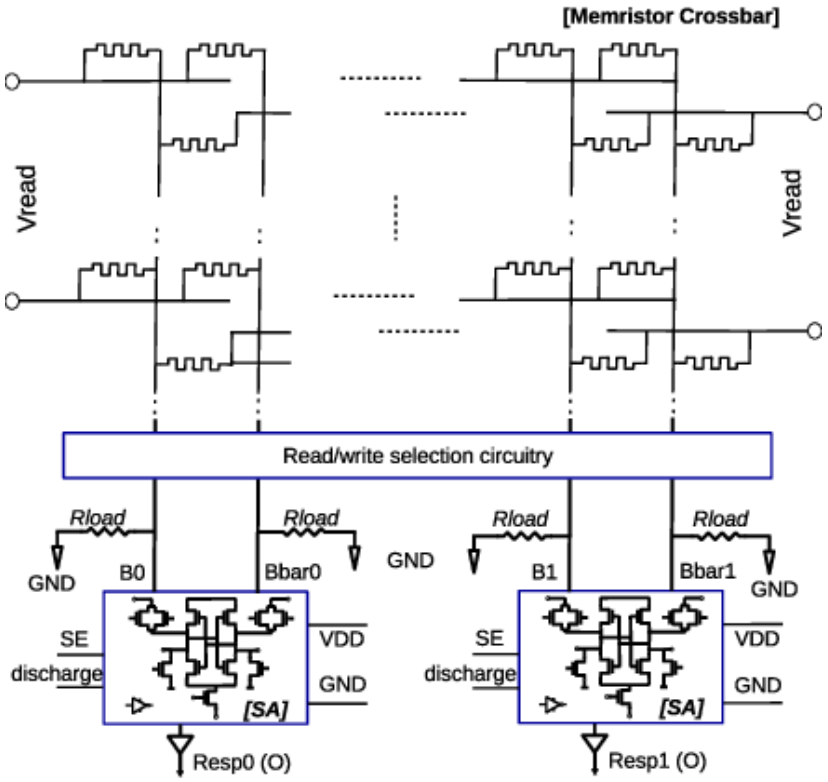


Figure 3.16: Application of the our designed sense amplifier for the XbarPUF [73]

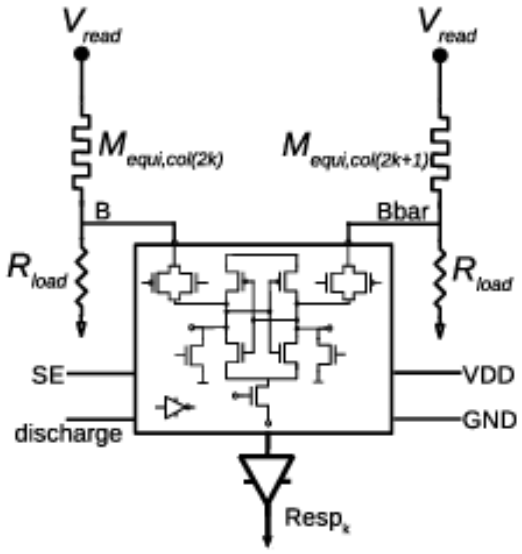


Figure 3.17: An equivalent input-output circuit for our sense amplifier circuit when used in an XbarPUF [73]

circuit, discharge transistors can be omitted to save power, area, and delay. We still keep these transistors in our generic SA design as they would be useful for memristor crossbar based memory applications.

As we have mentioned before, we are designing everything with 65nm CMOS technology and the minimum transistor length, L_{unit} is 60nm. Transistors with more larger than minimum size are used to ensure low percentage of mismatch in analog circuits which can be found from the datasheet of this particular technology. Thus we choose the base minimum width, W_{unit} for this SA to be 1.2μm. Each of the four access transistors, M5-M8 in Figure 3.15 are sized to be of 1.2μm width. With equal size, NMOS transistors would be stronger than their PMOS counterpart, thereby facilitating the latching of near-to-GND input voltages. They also have very small voltage drop across them. M1-M4 are the four sensing transistors in this design. Two PMOS transistors, M1-M2 are sized to have 1.2μm width each while the two NMOS sensing transistors are sized 2.4μm each. The NMOS are thus four times stronger than their PMOS counterpart, similar to the cross-coupled inverters in an SRAM cell. Two discharge transistors, M10 and M11 are sized of 2.4μm width each to have a balance between quick discharge time vs. quick recharge of internal nodes. The bottom NMOS, M9 is sized 3.6μm, larger than the sensing transistors, to create a fast discharging path when the SA is in sensing or evaluation phase. The single inverter in Figure 3.15 and other buffers which might be needed as the output stage of this SA to comply with the load demand are also sized properly to ensure fast reliable operation, reduce mismatch, and improve yield. It is important to note that all these transistor sizes are the minimum size that we have worked with and later we have analyzed to find the best transistor size for our design.

This SA circuit works in three states: latching phase, sensing phase, and discharge phase. As we have discussed before, the discharge phase and the discharge transistors are optional for our SA when used as part of the XbarPUF. During read or latching phase, the sense enable signal, 'SE' is kept low and thus the access transistors are turned on. Thus the input voltages, 'B' and 'Bbar', which are basically load voltages from a pair of crossbar columns, are latched or sampled into internal nodes 'O' and 'Ob', respectively. These initial voltage act as seed values during the next phase. After inputs are sampled properly, 'SE' is turned high

in the sensing or evaluation phase, thereby shutting off the access transistors and turning on transistor M9 to create a path to ground. Depending on which of the two nodes, 'O' or 'Ob' is higher, the other node discharges through M9 to ground while the node with the higher voltage recharges to VDD through the PMOS transistors by the positive feedback action of cross-coupled inverters formed by M1-M4. The signal 'SE' should be kept high during this whole time as it also provides power for this recharge besides controlling most of the transistors. In the next phase, these internal nodes are discharged through M10 and M11 to ground to be ready for another sensing operation. This phase is only needed when these discharge transistors are used in the SA design. A small short pulse of around $50ns$ can be applied to discharge both the nodes in short time. We also add buffers to both output 'O' and 'Ob' when we integrate this SA circuit into our XbarPUF to be able to drive larger load. Buffers add negligible or even zero delay to our circuit due to relatively fast rising/falling delay of inverters compared to the SA internal output nodes.

The layout of the SA should be symmetric to both inputs as well as for the cross-coupled inverters to avoid creating any bias to any of the two inputs. Wire lengths, widths, and other internal connections are routed in such a way that output nodes see equal amount of load. We have learned that finger widths or the number of fingers do not affect the design for VLSA based designs [81]. Thus our choice of the number of fingers is only determined by the motivation of maintaining a low aspect ratio, low resistance path, low area and also not to increase the pitch of the memristor crossbar circuit.

Results and Analyses of the Proposed Design

As we have argued already, capability of driving near-to-GND bitline voltages with high speed, able to detect small voltage difference, and compatibility with memristor based crossbar architectures are some of the factors behind choosing this SA design. We have built and experimented with the comparable SA circuit topology ourselves. Table 3.3 lists the results of comparison among all these different circuits. The supply voltage used for this simulation is 0.4V and differential input voltages are 150mV and 155mV. For a fair comparison, same unit length ($60n$m) and width (1.2μm) are used to each of these circuits. Our designed SA is easily found to be the fastest among these designs. SA circuits based

Table 3.3: Comparison of our designed SA with different circuits from different topology [73]

Topology	CLSA [29]	VLSA [56]	PCSA [85]	Ours [73]
Precharge needed	yes/no	yes	no	no
Worst delay (μs)	2.75	3.51	0.62	0.27
Avg. power (nW)	0.48	0.77	0.42	0.63 / 0.19

on CLSA [29] and VLSA [56] are slow when the input voltages are close to GND and no pre-charging available. The PCSA in [85] is found to consume small power with relatively high speed. Our design, on the other hand, has comparatively higher power consumption when discharge transistors are used. However, these are not used when we use this SA to build PUF circuits, thereby making our design to also have the lowest power consumption. The power consumption of Table 3.3 includes power consumption by an inverter and also power consumption during all phases of the SA. Another thing to note is that because of the high delay of some of these designs, the results of Table 3.3 are generated using a lower frequency input with clock period of 12μs instead of the required maximum of 2μs to make sure they all converge to a valid output within the time frame. Therefore, actual power consumption would be higher than this if higher clock frequency is used.

Figure 3.18 displays the waveform for our implemented SA design at 0.4V VDD with bit-line voltages of 150mV and 155mV ('B' and 'Bbar' in Figure 3.15). The sense enable signal, 'SE' has a clock period of 2μs. The analog voltages of two input signals 'B' and 'Bbar' are flipped on each clock cycle to make the output flip on every cycle as well to see if our design can resolve to opposite output voltages quickly. A pulse signal of 50ns duty cycle with the same 2μs period is used to discharge the output nodes 'O' and 'Ob' at the end of each sensing/decision phase. During this sensing phase, 'O' and 'Ob' resolves either high (VDD) or low (GND), based on which of the input signals is higher than the other. The output is valid before discharge signal is high and a sufficient delay after 'SE' goes high. Outputs, 'O' and 'Ob' should be alternating on each cycle with because of the input pattern which is evident from Figure 3.18.

We have run several Monte Carlo simulations to determine the yield of our designed SA circuit and the results are displayed in Table 3.4, 3.5, and 3.6. For each Monte Carlo

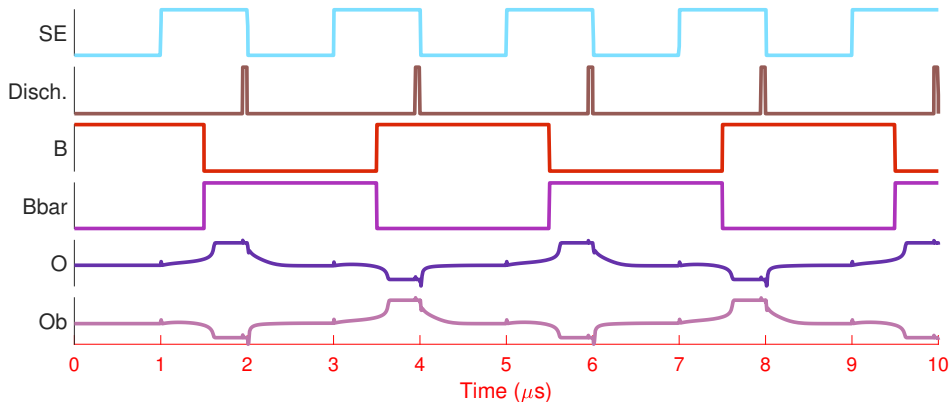


Figure 3.18: Waveform for transient simulation of proposed SA for input bit-line voltages of 150mV and 155mV with VDD = 0.4V [73]

simulation, we have applied two sets of input patterns, one where output should be a '1' and one where output is a '0'. The clock period (of signal 'SE') is chosen to be 2μs with 50% duty cycle, compatible with the XbarPUF. Thus 1μs is being allotted for for the SA to make a decision plus an additional 50ns discharge phase. The input voltages are again chosen to be 150mV and 155mV initially, flipped on each second cycle to get both a '1' and '0' output on consecutive cycles. The supply voltage (VDD) and transistor widths (W) and lengths (L) are also varied in different Monte Carlo simulations. To determine the best/near optimal W, H and VDD for the SA, we have run Monte Carlo analysis for 500 chips for our designed SA with different (W/L) ratios and different VDD. Here in these tables, W = W_{unit} = 1.2μm and L = L_{unit} = 60nm (min. transistor length of this 65nm technology). As expected, increasing the L as well as the (W/L) ratio increase transistor area and thus reduce mismatch, thereby considerably improving the yield of the design. It can also be noticed from Table 3.4 that with increasing VDD, the percent of yield decreases. It can be explained by the fact that input voltages are close to GND rather than VDD and, therefore, with higher VDD, there is a higher amount of leakage from it which corrupts internal nodes of SA just after SE goes high to start the sensing phase, thereby reducing yield. However, with a VDD lower than the threshold voltage (at 0.4V VDD where for this technology, threshold voltage, V_{th} is around 500-550mV), the delay is longer but the frequency is still the same and, therefore, it may not resolve for all cases due to additional delay arising from mismatch making the input voltage

Table 3.4: Percent yield of our designed SA for different transistor sizes and supply voltage [73]

Transistor sizes	$(\frac{W}{L})$	$(\frac{W}{3L})$	$(\frac{3W}{3L})$	$(\frac{2W}{5L})$	$(\frac{5W}{5L})$
VDD = 0.4V	67.35	72.45	78.55	83.25	87.20
VDD = 0.6V	64.20	77.20	89.70	90.70	98.50
VDD = 0.9V	61.10	73.10	84.70	87.00	95.50
VDD = 1.2V	60.50	72.70	84.30	87.20	95.40

Table 3.5: Sensing delay (nS) of our designed SA for different transistor sizes and supply voltage [73]

Transistor sizes	$(\frac{3W}{3L})$	$(\frac{2W}{5L})$	$(\frac{5W}{5L})$
VDD = 0.4V	712	625	627
VDD = 0.6V	19.63	16.7	16.7
VDD = 0.9V	1.02	1.524	1.523

difference almost indistinguishable. Thus, at 0.4V VDD, this design has the lowest yield compared to higher, over the threshold VDD as in many cases the SA doesn't resolve fast enough to a full-swing output. Although for the leftmost column of transistors in Table 3.4 with unit length (60nm), yield is actually the highest for 0.4V VDD. The reason behind this is that the SA operates very fast with the transistor length is smaller and thus the decreasing effect of extra delay due to mismatch is less than the yield improvement achieved by a lower VDD. Thus it can be concluded that to achieve an overall better yield ($\geq$80-85%), larger transistors (rightmost three columns of Table 3.4) should be considered.

Table 3.6: Energy(Joule) per cycle of our designed SA for different transistor sizes and supply voltage [73]

Transistor sizes	$(\frac{3W}{3L})$	$(\frac{2W}{5L})$	$(\frac{5W}{5L})$
VDD = 0.4V	20.66f	16.79f	41.79f
VDD = 0.6V	0.80p	0.50p	1.26p
VDD = 0.9V	11.08p	5.54p	13.65p

Larger transistor sizes reduce variation due to mismatch and thus improve yield, but they also contribute to larger area, delay and energy overhead. Therefore, there is a trade-off between yield vs. delay and energy overhead of the circuit. To find system delay and energy consumption, we have experimented with the transistor sizes listed in Table 3.4, except for the (W/L) and (W/3L) as their yield are too low and thus are left out from further analysis. 1.2V VDD is also not considered anymore as it increases the power consumption considerably, without improving the yield. Table 3.5 lists the sensing delay for the selected three transistor sizes with comparatively better yield. Table 3.6 lists the total energy consumption per cycle of the designed SA for the same transistor sizes. From these two tables, it can be found that transistor size of (2W/5L) i.e. 240μm width and 300nm length is a good trade-off offering much lower sensing delay as well as low conversion energy at 0.6V VDD, compared to other. Area and leakage power should also be lower compared to the other two sizes as W and (W/L) ratio are smaller. However, depending on the design requirements and application, a trade-off among yield, delay, total area and energy overhead can be easily made from our detailed analysis.

We have also measured the reliability of our design for different differential voltage between the two input voltage and for different temperature. The reliability is calculated as the percent of time our design produces a correct output out of 500 evaluations while temperature is being varied. The detailed result is presented in Figure 3.19. The results are obtained from SPICE simulation as well using Monte Carlo analysis by varying temperature from 0^0C to 100^0C for different transistor sizes and input voltage differences. As expected, the reliability increases quickly as the difference between two input voltage increases. An increase in transistor area also displays a linear increase in reliability of operation.

On-Chip Design and Testing

We have designed a unit size sense amplifier as a stand-alone test structure using a 65nm CMOS technology. The schematic and layout of this designed sense amplifier are shown below in Figure 3.15 and 3.20, respectively. The layout dimension is 8.039μm $\times$ 13.035 μm. Metal layers are made larger than minimum to help reduce any drops in analog circuit path. Every gaps between any layers are kept at minimum distance.

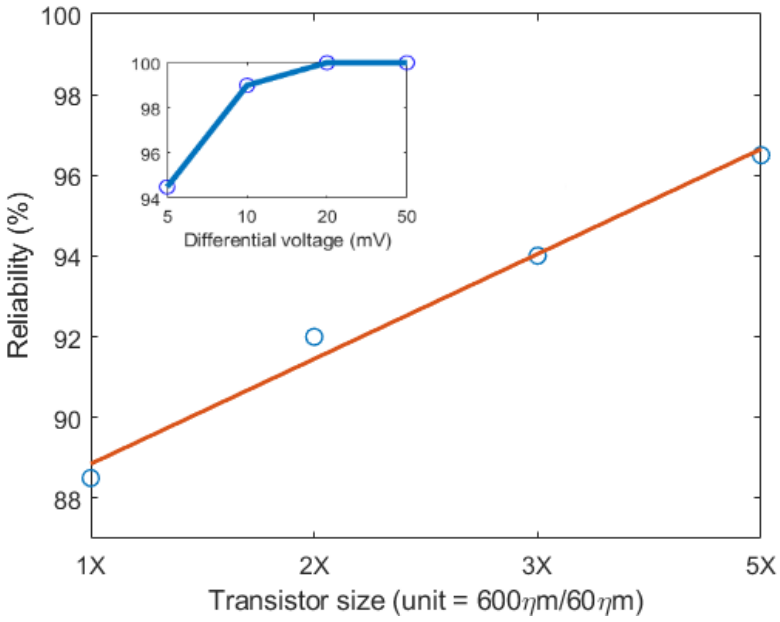


Figure 3.19: Reliability across multiple evaluations of our sense amplifier for different transistor size and differential voltage

After fabrication, I have also performed an initial functionality test of this sense amplifier. Using PSOC (programming system on chip) microcontroller, different input voltages are applied to get corresponding outputs, as shown in Figure 3.21. One input is at a fixed DC voltage while the other input is varied between two DC voltage levels such that it is higher than the first input in half of the times and lower in other times. This means the output would toggle between high and low voltage alternately when 'sense enable' signal is asserted. Even with the presence of noise, it is clearly visible that the output is correct i.e. when 'sense enable' signal is high, the output shows a logic high when the first input (B0) is higher than the second input (B1) and output is low when B1 is larger than B0. I have tested several such sense amplifier test structures and all display correct functionality. The data are collected from an oscilloscope after getting the output from our chip located inside of a probe station. More details about this testing can be found in Appendix D.

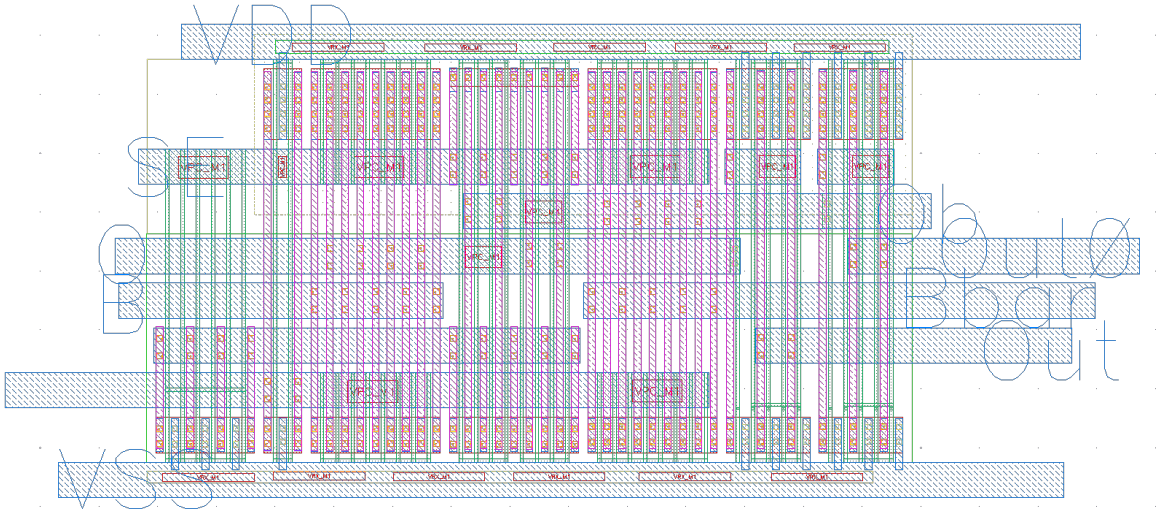


Figure 3.20: Layout of the designed sense amplifier ($8.049\mu\text{m}\times13.03\mu\text{m}$)

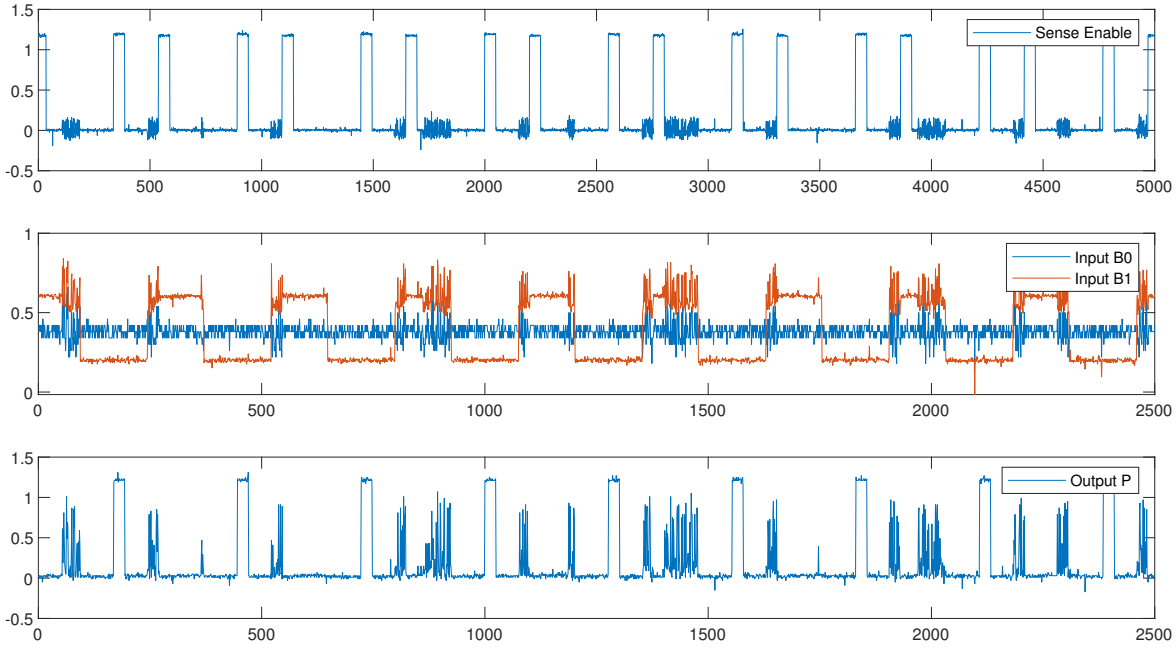


Figure 3.21: On-chip test results from our designed sense amplifier

3.3 Design of Twin Memristor TRNG

3.3.1 Motivation

The main purpose of this part of our work is to analyze existing memristor based TRNGs from their underlying theory and mathematical basis. We have analyzed a very recent and popular memristor based TRNG circuit and formulated how a change in supply voltage and temperature or process variation can affect the performance of a TRNG. To improve resiliency against environmental effects and process variation, we have suggested a TRNG design that leverages the variability in HRS (high resistance state) of a memristor. This core idea is to utilize relative nature of measurements, i.e. the HRS of one memristor is compared with the HRS of another to generate a single random bit. Since HRS usually has shown higher variability compared to other parameters of a memristor [69], we have chosen HRS as the source of randomness for our design. The differential nature of our TRNG design should make it robust against process variation and environmental changes.

3.3.2 Our Design

Process variation, temperature and supply voltage variation can shift the switching time distribution of a memristor. Therefore, any TRNG design based on an initial estimated distribution of C2C (cycle to cycle) switching rate would be biased towards either '1' or '0' and thus could degrade the TRNG performance. In this particular work, we are instead designing the TRNG based on relative distribution of two HRS of two memristors. In this design, the main source of entropy is the C2C HRS variation of a pair of memristors. After applying the same voltage for a fixed duration of time across a pair of memristors, their states are compared to generate a random stream of bits. Figure 3.22 shows the circuit diagram of the existing switching time based single memristor TRNG [26] and our designed twin memristor TRNG based on relative HRS difference. Temperature, supply voltage change and process variation all have smaller effect on this TRNG performance as we analyze below.

We consider two memristors in a chip which are physically close to each other. Because they're physically close, we can assume that their mean HRS are much closer to each other

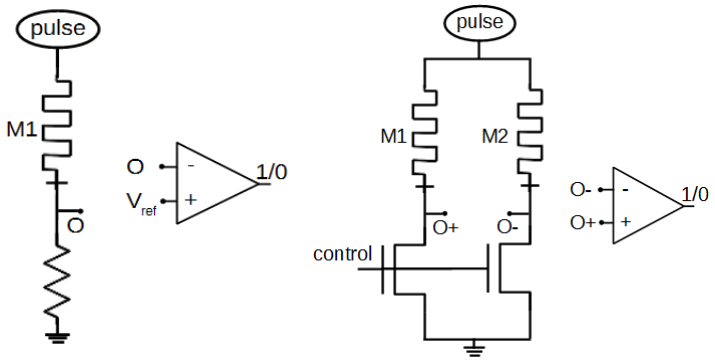


Figure 3.22: Conceptual circuit diagram of (left) existing switching time based single memristor TRNG [26], and (right) our designed twin memristor TRNG based on HRS difference [67].

than the usual chip-to-chip process variation. Suppose the mismatch variation between the HRS of these two memristors is $1\sigma = 1\%$. Each memristor's HRS has a C2C normal (gaussian) distribution of mean μ_{HRS} and standard deviation $\sigma_{HRS} = 10\%$ of μ_{HRS}.

$$\begin{aligned} \mathbf{HRS_1} &= N(\mu_{HRS1}, \sigma_{HRS1}) \\ \mathbf{HRS_2} &= N(\mu_{HRS2}, \sigma_{HRS2}) \end{aligned} \tag{3.9}$$

Since in our design, we compare these two HRS states, the result of comparison can be expressed as another Normal random variable as:

$$\begin{aligned} \mathbf{Z} &= \mathbf{HRS_1} - \mathbf{HRS_2} \\ &= N((\mu_{HRS1} \sim \mu_{HRS2}), \sqrt{\sigma_{HRS1}^2 + \sigma_{HRS2}^2}) \end{aligned} \tag{3.10}$$

If μ_{HRS2} is greater than μ_{HRS1} and thus $\mu_{HRS2} = 1.01^*\mu_{HRS1}$ (1% larger) and $\sigma_{HRS1} = 0.1^*\mu_{HRS1}$, $\sigma_{HRS2} = 0.1^*\mu_{HRS2}$ (10% standard deviation), then equation 3.10 becomes:

$$\begin{aligned} \mathbf{Z} &= N(.01 * \mu_{HRS1}, \sqrt{(0.1 * \mu_{HRS1})^2 + (0.1 * 1.01\mu_{HRS1})^2}) \\ &\approx N(.01 * \mu_{HRS1}, 0.1 * \sqrt{2} * \mu_{HRS1}) \end{aligned} \tag{3.11}$$

The distribution, **Z** doesn't have a zero mean, thus the ratio of ones and zeros in the output wouldn't be the same if we build a TRNG using this distribution. However, the mean should be close to zero as which can be easily found from a Z-table. The z-score of this normal distribution which is equal to finding the probability of a randomly chosen value to be less than 0 is:

$$\begin{aligned} z = \frac{x-\mu}{\sigma} &= \frac{0 - 0.01 * \mu_{HRS1}}{0.1 * \sqrt{2} * \mu_{HRS1}} \\ &= -0.0707 \end{aligned} \tag{3.12}$$

where, z represents a Normal random distribution of mean 0 and standard deviation 1, transformed from the distribution, **Z** to calculate the probability using a z-table.

From z-table, the probability of getting a '0' value i.e. the area on the left side of the distribution from the center for a z score of above-calculated -0.0707 is:

$$\mathbf{P}(x \in \mathbf{Z} \leq 0) = 0.4718 \tag{3.13}$$

It should be noted that if the difference in means of two HRS values of two memristors are less than 1σ, then the the probability number in equation 3.13 would be closer to the ideal of 0.5, i.e. the center of distribution would more close to 0. Our design thus inherently has some degree of bias. But this provides this design with a better resilience against environmental changes compared to existing designs as we analyze next.

3.3.3 Effect of Environmental and Process variation on Existing Single Memristor TRNG

As mentioned earlier in this section, most of the existing single memristor TRNG designs are based on the stochastic switching time (or the RTN). We already know memristors display C2C variation in its switching characteristics. The switching time (towards either LRS or HRS) of a single memristor can be approximated to also have a normal distribution over

many cycles. This can be formulated as:

$$\mathbf{t_{swn}} = N(\mu_{tswn}, \sigma_{tswn}) \tag{3.14}$$

and,

$$\mathbf{P}(t \leq \mu_{tswn}) = \mathbf{P}(t > \mu_{tswn}) = 0.5 \tag{3.15}$$

To reiterate, equation 3.14 describes that the switching time of a memristor over different cycles is a Normal random distribution with mean of μ_{tswn} and standard deviation of σ_{tswn}. Therefore, 3.15 expresses the probability that the switching time at any random cycle, t ϵ t_{tswn} to be less than or equal to the mean switching time is 0.5 or 50%. Thus if we determine the mean, (μ_{tswn}) of that switching time over many cycles and apply a voltage greater than the threshold voltage of that memristor repeatedly for many cycles, then half of the time, that memristor would switch and the other half of the time, it would retain its state. This is the basis for most memristor based TRNG designs where memristors's stochastic switching time is the source of randomness.

This method requires an application of a voltage pulse of pulse width approximately equal to the mean switching time of that memristor. Clock jitters, skews or other noise could slightly change this pulse width which may change the switching probability drastically. The switching time also depends heavily on the magnitude of applied voltage pulse. Thus, any power supply noise (change in magnitude) would also strongly affect this switching time. Since a memristor's resistive states and threshold voltages are strong functions of temperature, a change in temperature would also change the switching rate and thus the switching probability overall. Thus any change in any of these factors would change the switching probability and thus equation 3.15 would not hold for the new changed distribution. Then the bit stream produced from that memristor based TRNG could be heavily biased towards either '1' or '0'. Next we are going to formulate how these environmental changes cause a change in the switching time distribution.

Supply Voltage Variation

We have used the improved McDonald model from [46, 69] as a memristor's switching model. Then we have used the switching equation to analyze the impact of voltage variation. The switching from LRS to HRS (or reset) of a memristor is governed by this equation [72]:

$$M(t_{i+1}) = M(t_i) + \frac{R_{diff}\Delta t V(t_{i+1})}{t_{swn}V_{tn}}, \tag{3.16}$$

where, V_{tn} and t_{swn} are negative threshold voltage and negative switching time (during reset) of a memristor, respectively. R_{diff} is the resistance difference between HRS and LRS. $M(t)$ presents the memristance and $V(t)$ is the voltage applied across the memristor at a particular time instant, t. Equation 3.16 can be rearranged as:

$$\Delta M = \frac{R_{diff}}{t_{swn}V_{tn}} * \Delta t, \tag{3.17}$$

If we integrate equation 3.17 over the total switching time, we can establish a relationship between applied voltage and switching time. We assume that the magnitude of applied voltage remains fixed for one switching cycle. Now from equation 3.17, we can estimate the actual switching time for any applied voltage:

$$\int_{LRS}^{HRS} dM = \frac{R_{diff}\left|V_{applied}\right|}{t_{swn}V_{tn}} * \int_{0}^{t_{sw,actual}} dt \tag{3.18}$$

$$\implies t_{sw,actual} = t_{swn} * \frac{V_{tn}}{\left|V_{applied}\right|} \tag{3.19}$$

The resulting relationship between voltage applied across a memristor and actual time to switch is found to be proportional which is what we expect from a linear switching model [47]. If the change in applied voltage is constrained ($\pm$10%), then a linear equation should provide reasonable accuracy in this window. If $V_{applied}$ is equal to the threshold voltage, V_{tn}, then the actual switching time is equal to the minimum switching time. Otherwise, from equation 3.19, we can find out that a $\pm$10% change in $V_{applied}$ from V_{tn} would result in a ($\approx$ -9%,+11%) change in actual switching time.

Temperature Variation

Temperature directly affects states and threshold voltages of a memristor and thus in turn affect the actual switching time. From [11, 18], we can find that the magnitude of threshold voltage decreases as temperature increases within the range $\approx [0, 100]^0$C and we can approximate this change using a linear equation as:

$$V_{tn}(\theta) = V_{tn0}[1 + \alpha_{Vtn}(\theta - \theta_0)], \tag{3.20}$$

where, V_{tn0} is the threshold voltage at room temperature, θ_0=25^0C. We have calculated the temperature coefficient for negative threshold voltage from [18] as ≈-0.007/^{0}C. Therefore, for a temperature change of $\Delta\theta$, the new V_{tn} would be:

$$V_{tn}(\theta) = [1 - 0.007 * \Delta\theta] * V_{tn0}, \tag{3.21}$$

The modified switching time is thus found to be:

$$t_{sw,actual} = t_{swn} * \frac{[1 - 0.007 * \Delta\theta] * V_{tn0}}{V_{applied}} \tag{3.22}$$

Process Variation

Emerging devices like memristor usually have a large process variation as the technology, fabrication technique etc. are not mature. All major parameters of a memristor show a relatively larger variation from chip to chip. For switching time based existing TRNG circuits, we are only interested in the variation of switching time. For our purpose, we have considered a 5% standard deviation (σ_{tswn}) on negative switching time, t_{swn}.

$$\mathbf{t_{swn}} = N(\mu_{tswn}, \sigma_{tswn}), die - to - die \tag{3.23}$$

where $\sigma_{tswn} = 0.05^{*}\mu_{tswn}$. Thus for a 1σ variation of this switching time, the shifted mean of the new distribution would be:

$$t_{sw,actual} = \begin{cases} 1.05 * \mu_{tswn}, & 1\sigma \ above \ mean \\ 0.95 * \mu_{tswn}, & 1\sigma \ below \ mean \end{cases}. \tag{3.24}$$

3.3.4 Variation Modeling for Our Designed TRNG

Supply Voltage Variation

Changing the magnitude and duration of a voltage applied across a memristor would change its switching rate. However, there is no one-to-one relationship between this voltage magnitude ($\pm$10% variation) with final HRS. In our design, we make the switching time large enough so that a memristor would always switch as we are only interested in the final randomized HRS values, not the probabilistic switching itself. Therefore, due to supply voltage variation, the time to generate a valid bit from our designed TRNG changes slightly, but the value of that bit itself is considered to be unaffected here.

Temperature Variation

Temperature can change the HRS of both memristors in our TRNG. For a relatively narrow temperature window of $[0-100]^{0}$C, this change is approximated by a linear equation, similar to threshold voltage change [69]. Suppose, we increase the temperature by an amount of $\Delta\theta$. From equation 3.20, we can find expressions for the two modified HRS as:

$$\begin{aligned} HRS1'(\theta) &= HRS1[1 + \alpha_{HRS} * \Delta\theta] \\ HRS2'(\theta) &= HRS2[1 + \alpha_{HRS} * \Delta\theta] \end{aligned} \tag{3.25}$$

where, HRS_0 represent HRS at room temperature and α_{HRS} is the temperature coefficient (linear) of HRS and approximated as -0.008/^{0}C. The chip-to-chip or more specifically die-to-die standard deviation of HRS considered here is $\sigma_{HRS} = 0.1^{*}\mu_{HRS}$ (10% of HRS).

Now, we define the term $C_{\Delta\theta} = 1 + \alpha_{HRS} * \Delta\theta$. Thus we can rewrite equations 3.9. We assume that the standard deviation (and variance) of both HRS stays the same.

$$\begin{aligned} \mathbf{HRS'_1} &= N(\mu_{HRS1'}, \sigma_{HRS1'}) \\ &= N(\mu_{HRS1} * C_{\Delta\theta}, \sigma_{HRS1}) \\ \mathbf{HRS'_2} &= N(\mu_{HRS2'}, \sigma_{HRS2'}) \\ &= N(\mu_{HRS2} * C_{\Delta\theta}, \sigma_{HRS2}) \end{aligned} \tag{3.26}$$

and

$$\begin{aligned} \mathbf{Z'} &= \mathbf{HRS'_1} - \mathbf{HRS'_2} \\ &= N(C_{\Delta\theta} * (\mu_{HRS1} \sim \mu_{HRS2}), \sqrt{\sigma^2_{HRS1} + \sigma^2_{HRS2}}) \end{aligned} \tag{3.27}$$

Thus equation 3.26 and 3.27 represents scaled versions of the normal distribution for HRS of equation 3.9 and 3.10, respectively. Now we can derive the new distribution of our designed TRNG by using equation 3.26 with equations 3.10 and 3.11. The z-score of this scaled normal random distribution is:

$$\begin{aligned} z = \frac{x - \mu}{\sigma} &= \frac{0 - C_{\Delta\theta} * 0.01 * \mu_{HRS1}}{0.1 * \sqrt{2} * \mu_{HRS1}} \\ &= -C_{\Delta\theta} * 0.0707 \end{aligned} \tag{3.28}$$

For a 50^0C temperature increase, $C_{\Delta\theta}$ = (1-.008*50) = 0.6 and thus the z-score would be z = 0.0424. The probability of getting 0 is:

$$\mathbf{P}(x \in \mathbf{Z} \leq 0) = 0.4831 \tag{3.29}$$

This z-value is very close to the value derived in equation 3.13. Thus a change in temperature does not cause a significant change on the ratio of 1's to 0's and the TRNG performance does not degrade too much with changing temperature.

Process Variation

We have already observed that a change in temperature creates a HRS distribution which is a scaled version of the original distribution. Process variation would make the HRS values to be different from their die-to-die mean HRS, causing a shifted distribution profile. However, since we are using the difference in states between two memristors, the result would just be a random distribution with scaled mean and standard deviation of the original distribution.

$$\begin{aligned}\mathbf{Z} &= \mathbf{HRS_1''} - \mathbf{HRS_2''} \\ &= N((\mu_{HRS1''} \sim \mu_{HRS2''}), \sqrt{\sigma^2_{HRS1''} + \sigma^2_{HRS2''}}) \\ &\approx N(.01 * \mu_{HRS1''}, 0.1 * \sqrt{2} * \mu_{HRS1''})\end{aligned} \tag{3.30}$$

Suppose, $\mu_{HRS1''}$ is 1σ larger than the mean μ_{HRS1}. Thus $\mu_{HRS1''} = 1.1^*\mu_{HRS1}$ and the z-score of this would again be:

$$\begin{aligned}z = \frac{x-\mu}{\sigma} &= \frac{0 - 0.01 * \mu_{HRS1''}}{0.1 * \sqrt{2} * \mu_{HRS1}} \\ &= \frac{0 - 0.01 * 1.1 * \mu_{HRS1}}{0.1 * \sqrt{2} * \mu_{HRS1}} \\ &= -0.0778\end{aligned} \tag{3.31}$$

The probability of getting 0 for 1σ variation of HRS is thus:

$$\mathbf{P}(x \in \mathbf{Z} \leq 0) = 0.469 \tag{3.32}$$

Thus the relative probability of getting 1's and 0's change by a small amount with process variation for our design. This is an excellent property since it means that it does not require changing any design parameters even if memristors have large variations across different die as the TRNG performance should remain close to its ideal operating condition.

3.3.5 Results and Analyses of Designed Twin-Memristor TRNG

First, we have analyzed both the existing single memristor TRNG and our new twin memristor TRNG design in terms of the probability of producing 1's and 0's with respect to

varying operating conditions. We have used MATLAB platform to simulate both designs. We calculate the percentage of 0's with the sum of 0's and 1's in normal operating condition, when temperature changes $\pm 50^0 C$ from room temperature, with $\pm 10\%$ variation in applied voltage and 1σ process variation. This 1σ process variation corresponds to switching time variation for single memristor TRNG and is assumed to be 5%. For our TRNG design, this variation is of HRS and assumed to be 10% as HRS shows higher variation than switching time or other memristor parameters. Table 3.7 lists the performance of both TRNGs for different operating conditions. As expected, since both process variation and environmental variation strongly affect switching time, single memristor TRNG shows a large bias towards either 1's or 0's except for ideal condition. Our TRNG is based on relative measurement of HRS and because of difference due to mismatch, inherently has a small bias i.e. the proportion of 0 is not 50% ideally, as can be seen from Table 3.7. However, due to the relative nature of measurement, our TRNG design only shows negligible change in performance with changing temperature or voltage. Thus our design is found to be robust against both environmental changes and process variation. As expected, performance of our designed TRNG doesn't show any big deviation from ideal condition where single memristor TRNG performance could be heavily affected.

To better understand the effect of process variation on both TRNG designs, we have run Monte Carlo simulation with 10000 different chips and generated 5000 random bits from each chip. Figure 3.23 shows the percent of 1's for different chips for both single and twin memristor TRNGs. A TRNG should ideally has a 50% probability of producing 1's i.e. the center of the distribution should be at 0.5. As expected from our detailed theoretical analysis, switching time based single memristor TRNG design displays a wider distribution with a standard deviation of 0.1805 when we consider this process variation. On the other hand, our twin memristor TRNG only has a narrow and much tighter distribution over process variation with a standard deviation of only 0.0322. This result shows our TRNG design is robust against process variation and eases circuit design work by eliminating the need to redesign for every different chip.

We have also run Monte Carlo simulation to show how the probability distribution of TRNG output changes from ideal for a $\pm 10^0 C$ change in temperature and for a $\pm 10\%$ change

Table 3.7: Percentage of 0's to the sum of 0's and 1's for different operating conditions [67]

Design with varying conditions	**Ideal/No change**	**Temperature**		**Voltage**		**Process**	
		+25^{0}C	**-25^{0}C**	**+10%**	**-10%**	**+1 σ**	**-1 σ**
Single-memristor	50	4.01	95.99	18.16	86.67	69.15	30.85
Our Twin-memristor	47.18	47.76	46.62	47.18	47.18	46.90	47.46

in supply voltage. This result is shown in Figure 3.24. Single memristor TRNG shows a large deviation from ideal distribution for both increasing and decreasing temperature and supply voltage, as the center of distribution shifts to far left or right. However, our designed TRNG almost show zero change.

3.3.6 Output Correction

Since our designed TRNG inherently has a small percentage of bias i.e. amount of 1's and 0's in the output are not equal even in ideal condition, some form of post-processing could be very useful to improve the quality of TRNG output. Therefore, we suggest to use simple Von-Neumann correction which can be easily implemented in circuit with minimum hardware overhead. Von-Neumann correction discards when two consecutive TRNG output bits are either both '1' or both '0'. But if the sequence of consecutive bits is '10' then the output becomes a '1', and if it is '01' then the output becomes a '0'. It helps remove any small bias and reduce correlation between consecutive stream of bits. XORing of two TRNGs is another technique. If outputs from two TRNGs are not correlated, then XORing them could help improve randomness in the output. Another technique is whitening, which XORs the output of a TRNG with a cryptographically secure PRNG to improve the statistical properties of output bit stream.

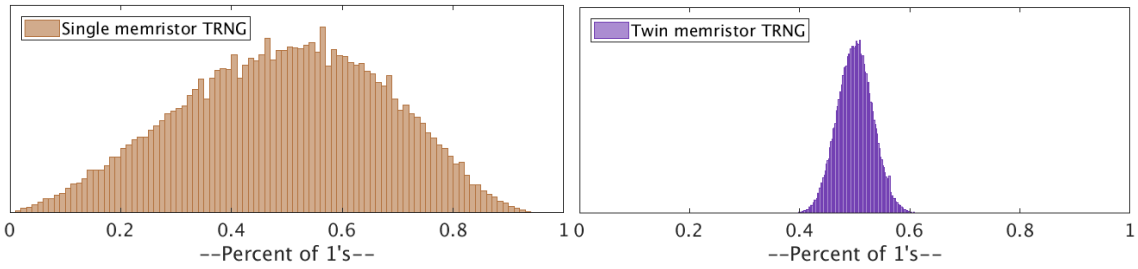


Figure 3.23: Histogram of bit bias with process variation for (top) existing single memristor design and (bottom) proposed twin memristor design [67]

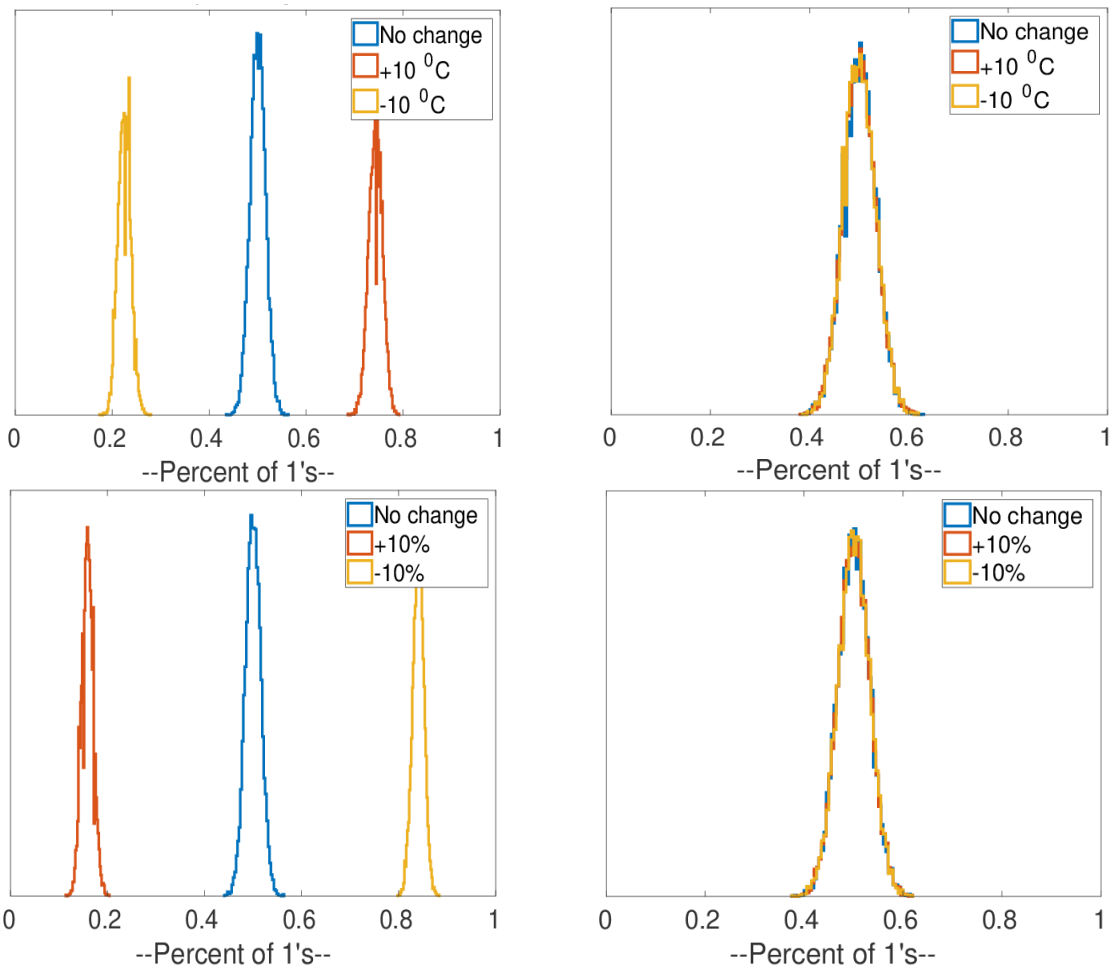


Figure 3.24: Histogram plot showing the trend for shift in distribution with temperature and voltage variation for both single memristor TRNG (left two figures) and our designed twin memristor TRNG (right two figures). Top two figures represent variation due to temperature for single and twin memristor TRNG, respectively while bottom two figures are for voltage variation. Existing single memristor TRNG displays a large variation while our design almost unchanged [67].

Chapter 4

Design and Robustness Analysis of XbarPUF

4.1 Design Analysis of XbarPUF

4.1.1 Noise Margin Analysis

The parameters that we have used in this work are listed in table 3.1. We have experimented with four different crossbar sizes, 4×2, 8×2, 16×2 and, 32×2, all XORed XbarPUF. We have not considered larger crossbar due to huge simulation time and memory requirement. For a 32×2 XORed XbarPUF, it takes more than a week just to generate a single security metric using a 40-core server. That's why it was impractical to use larger circuits than those used in this work. We have used the same set circuit and device level parameters to get a fair comparison across different crossbar sizes. The read and write voltages are chosen to be 0.6V and 1.3V, respectively for this simulation.

Noise Margin vs. Load Resistance

We have already evaluated the best load resistance for maximum noise margin for the XbarPUF in equation 3.8. However, we have also performed a sweep of the load resistance, R_{ld} for a 4×2 XORed XbarPUF using equations 3.7, 3.5 and 3.6 where the HRS and LRS values can be found in table 3.1. This is shown in Figure 4.1. The noise margin is plotted

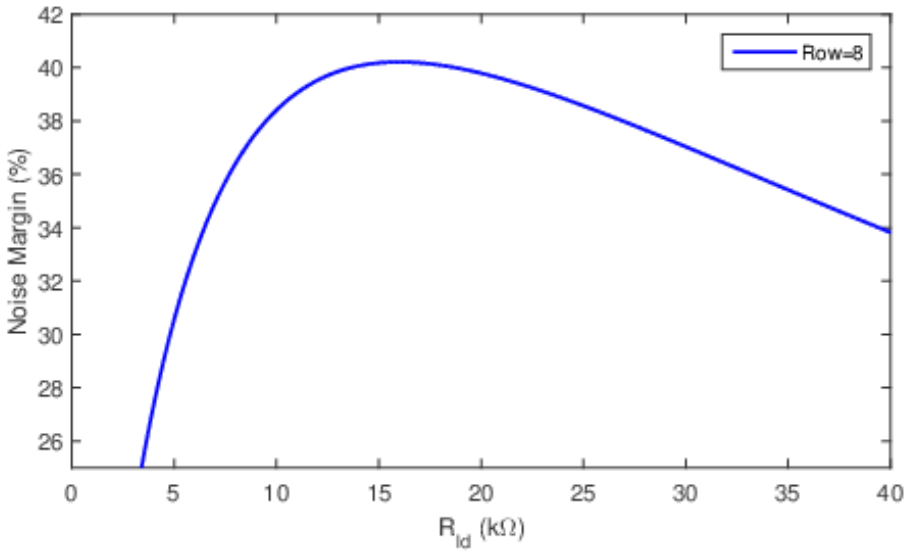


Figure 4.1: Relationship between noise margin and the load resistance [69].

as a percentage of read voltage and thus one can achieve a maximum of around 40% noise margin for a 4×2 XbarPUF with HRS = 300kΩ, LRS = 30kΩ. With 0.6V or 600mV read voltage, this translates to a 240mV noise margin voltage.

Noise Margin vs. Crossbar Size

We have also plotted the relationship between noise margin and no. of crossbar rows in Figure 4.2, again using parameters listed in table 3.1. As expected, the peaks of each of the curves for different crossbar sizes in Figure 4.2 are the same (at around 40%) which means the maximum achievable noise margin only depends on the memristor itself (HRS/LRS ratio) and does not change with a change in crossbar size. However, it is important to note that each curve in this figure gets narrower around the peak as the number of rows increases. This is because with increasing number of rows, more memristors are in parallel with each other which lowers the total effective memristance of a column and thus demands lower and more finer resolution for load resistance. Thus if the crossbar is too large, then a small deviation of load resistance from its optimum value due to process variation might reduce the noise margin considerably. This is an important result since this imposes an upper limit on the size of crossbar for a particular memristor device.

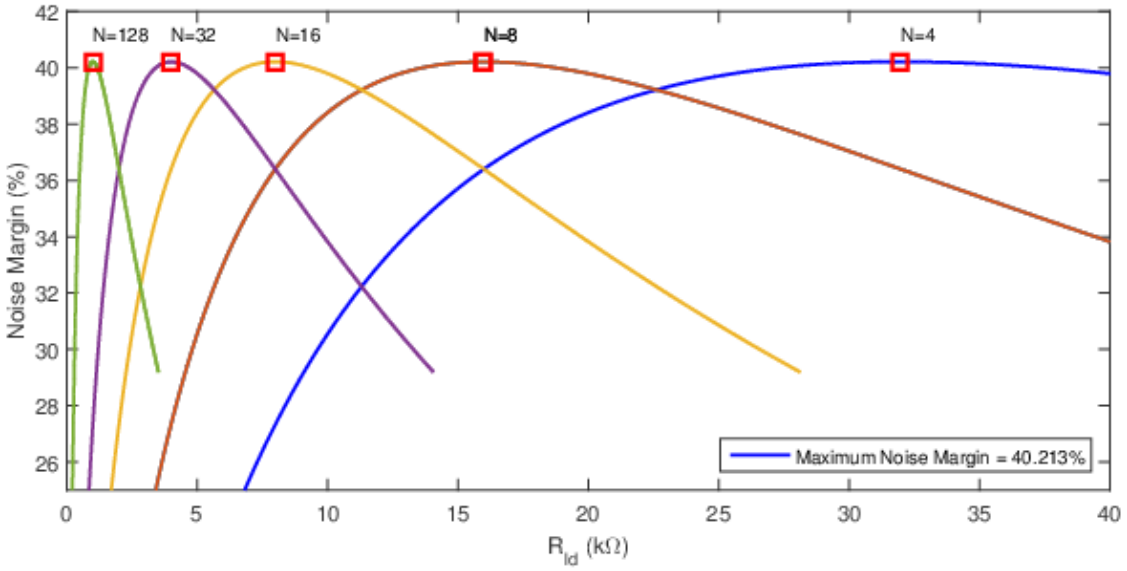


Figure 4.2: Relationship between maximum achievable noise margin and the number of rows (N) [69].

Noise Margin vs. Memristor Device Properties

We have analyzed to see if an improvement in device characteristics result in an improvement in the maximum noise margin achievable for the same crossbar size. Fig 4.3 shows the result for this where the read cross mark in the plot shows the current situation with the memristor parameter set that we have used (OFF/ON ratio of 10). In all the data points in this figure, LRS was kept at 30kΩ while HRS was increased. It is clear that using a memristor device with higher OFF/ON (or HRS/LRS) ratio would give increasingly better noise margin. Thus if we want to use large crossbars with reasonably good noise margin, we should look for memristors with high OFF/ON ratio.

4.1.2 Security Analysis of the XbarPUF

There are four primary security metrics that are often used to evaluate a strong PUF. These metrics are uniqueness, uniformity, bit-aliasing, and reliability. To evaluate the uniqueness and bit-aliasing, we have run 100 Monte Carlo simulations using Cadence Spectre for a set of several different PUF challenges. To evaluate uniformity, 100 unique challenges were applied over several different chips. To evaluate reliability, different challenges were applied

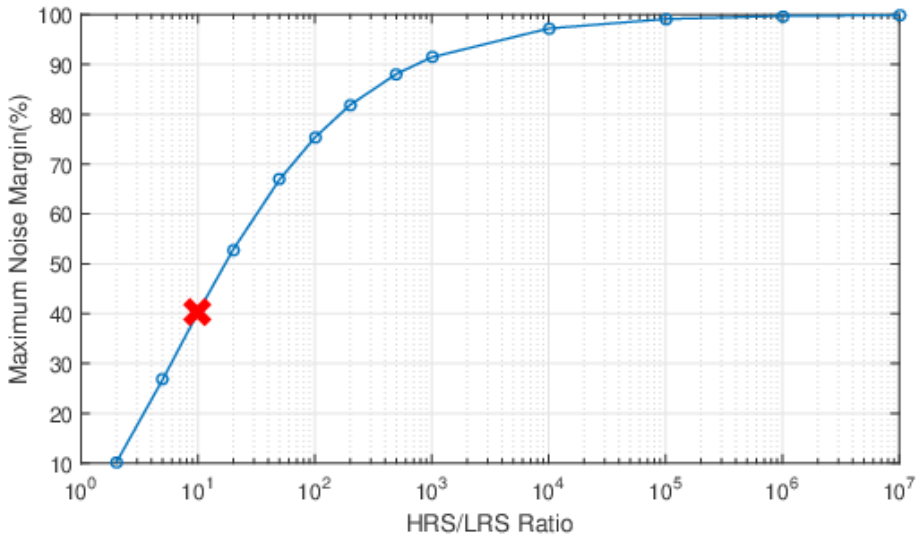


Figure 4.3: Relationship between the maximum achievable noise margin and HRS/LRS ratio [69].

for 10 different chips for 100 clock cycles each. Each challenge vector is randomly chosen and unique.

Security Metrics vs. Crossbar Size

First we have evaluated all the four security metrics for different crossbar sizes and represented in Figure 4.4. The uniqueness almost remains at its ideal value of 50%, uniformity and bit-aliasing are near 50% for all crossbar sizes. Reliability is near its ideal value of 100% for larger crossbars while being over 90% for smaller crossbars. This may seem counter-intuitive at first because with an increasing number of rows, the equivalent memristance drops further and thus should decrease noise margin. However, for a 32×2 XORed XbarPUF, the load resistance is found to be 2kΩ using equation 3.8. This is not too low so that wire resistance and other peripheral resistance would influence and worsen noise margin. However, it can be inferred that with an increasing number of rows the effect of cycle-to-cycle variation for each individual memristor to the overall equivalent memristance of the crossbar is reduced. Therefore, during different cycles, the probability of getting different responses for the same challenge is reduced and thus reliability is improved. However, at some point this would not be the case for very large size crossbar sizes since the equivalent memristance would be

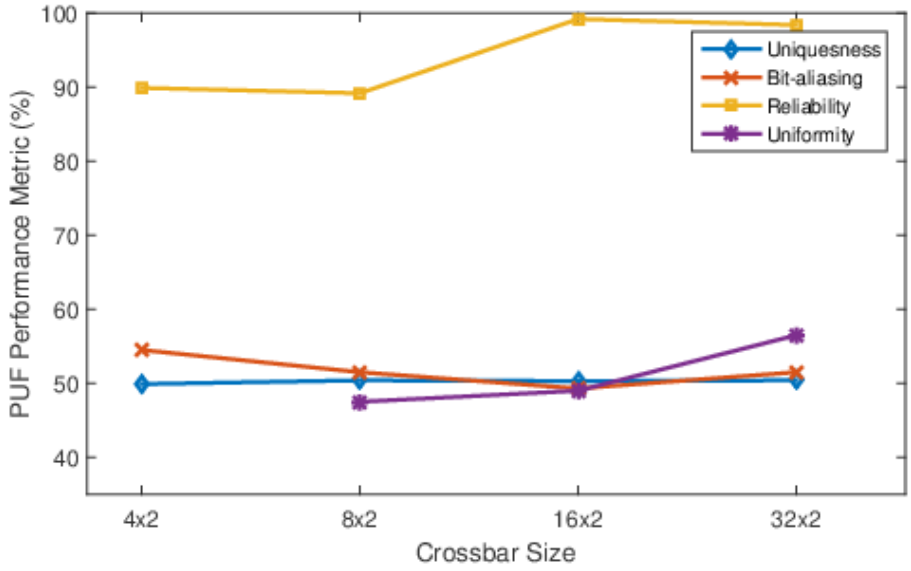


Figure 4.4: Security performance of XORed XbarPUF for different crossbar size [69].

comparable with any changes caused by environmental variation or noise. Having high HRS and LRS values would help reduce this effect and improve scalability, which is shown later in this section.

Security Metrics vs. Memristor Device Properties

Uniqueness and bit-aliasing with respect to different HRS/LRS (OFF/ON) ratios are shown in Figure 4.5. The LRS was kept at 30KΩ while HRS was varied to obtain all the data in these two plots. From Figure 4.5, we can see an improvement in both uniqueness and bit-aliasing (moves toward 50%) with increasing HRS/LRS ratio, which is expected since an increase in HRS/LRS ratio provides a better noise margin. It is also evident that even at a smaller HRS/LRS ratio (leftmost point on the plots in Figure 4.5) and with 10% cycle-to-cycle variation, good values for both uniqueness (≈47%) and bit-aliasing (≈48%) can be observed. This is expected from any PUF consisting of nano-materials such as memristors. Since memristors display much higher process variation compared to CMOS devices, it is statistically intuitive that memristor based PUFs would also exhibit near ideal uniqueness, uniformity and bit-aliasing in hardware implementation.

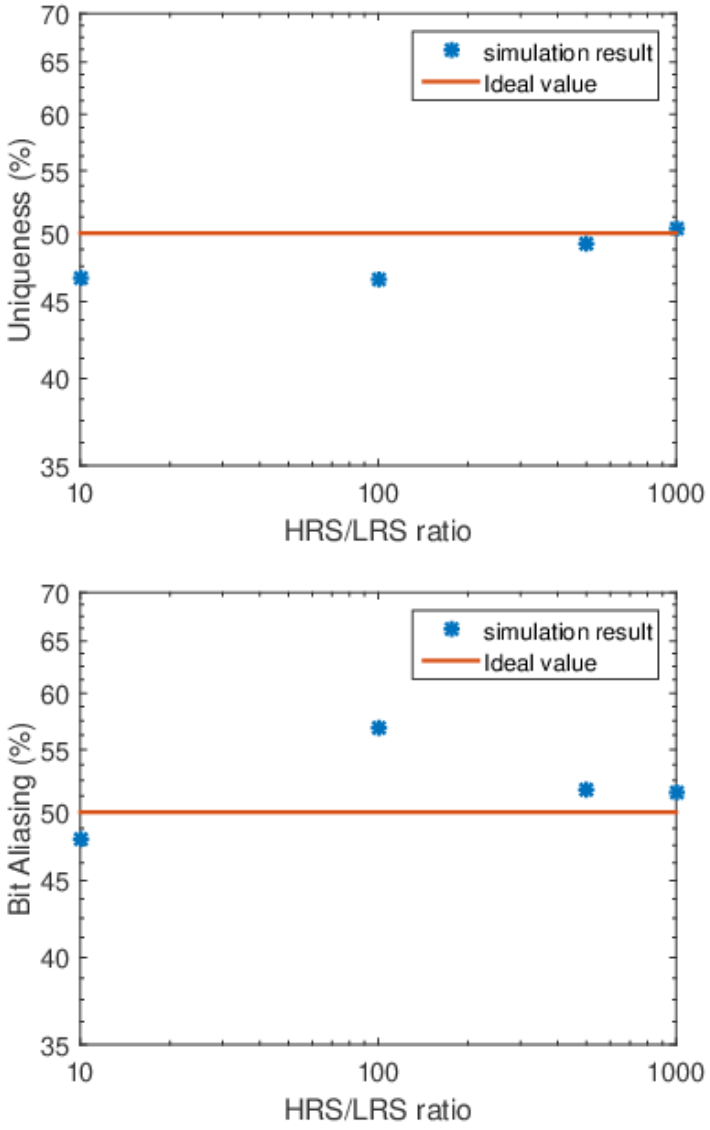


Figure 4.5: Dependence of uniqueness (left) and bit-aliasing (right) with HRS/LRS ratio and the absolute values of HRS and LRS. The base LRS was 30KΩ for both of these plots [69].

Security Metrics vs. Process Variation of Load Resistance

We have also performed a small corner analysis by including CMOS variation. Since the variation of peripheral circuitry would only affect slightly the read/write voltage and thus does not change the overall performance of the circuit we have used higher than minimum read/write voltages to obfuscate any such changes with 2% variation for load resistance. The security metrics either do not change or show only minuscule change and, therefore, the results is not explicitly provided here.

Uniformity is also relatively constant across different load resistances as it is a property of challenge. The other two metrics, uniqueness and bit-aliasing also remain effectively constant for up to 32×2 XbarPUF, which is shown in Table 4.1.

4.1.3 Detailed Reliability Analysis

Reliability is the most frowned upon security metrics of a PUF and in this work, we have evaluated reliability of our proposed XbarPUF against varying memristor device parameters i.e. HRS and LRS, against varying temperature and aging. Because of the high cycle-to-cycle variation present in memristors and other emerging nano-devices, they tend to show slightly different behavior across different cycles. Thus, the stability or reliability of the circuit is hampered with increasing cycle-to-cycle variation. Temperature also changes the memristor parameters and thus can potentially affect reliability. The reliability results for our XORed XbarPUF with and without temperature variation along with two sets of variation are presented in Table 4.2. As expected, reliability is improved, from 80% to 90% for a 4×2 XbarPUF, if the variation decreases from 10% to 2%. We have not found any change in reliability with changing temperature. However, it is expected to have some adverse effect of changing temperature in any circuit. But since in XbarPUF, instead of measuring absolute voltage/current, the relative differences between adjacent columns voltages of the XbarPUF are measured and, therefore, any environmental change should affect both the physically nearby columns almost equally, thereby preserving the original relative differences.

Reliability is also presented for different LRS (and HRS) values in Figure 4.6. There is little dependence on reliability with LRS values. However, if the LRS is too low, then the

Table 4.1: Impact of 2% variation of load resistance on the security properties of a 32×2 XORed XbarPUF [69]

Metric	**Min. R_{ld}**	**Nominal R_{ld}**	**Max. R_{ld}**
Uniqueness (%)	50.17	50.40	50.12
Bit-aliasing (%)	51.00	51.50	50.25
Uniformity (%)	53.00	56.50	47.50

Table 4.2: Performance of the XORed XbarPUF with variable temperature (10^{o}C- 100^{o}C) and with different cycle-to-cycle variation for the memristor parameters [69]

XORed XbarPUF	**Reliability** (%) (10% variation)	**Reliability** (%) (2% variation)
Fixed/room temperature	80	90
Variable temperature	80	90

equivalent memristance of the crossbar would be very low, thus decreasing noise margin and reliability. Therefore, on the left side of Figure 4.6, we see that reliability tends to decrease with decreasing LRS.

We have also measured reliability over 100 cycles for the XbarPUF consisting of fresh memristors and memristors aged over many set-reset cycles. With aging, the HRS/LRS ratio drops as shown in Figure 3.3. Therefore, the reliability of the XbarPUF also decreases as illustrated in Figure 4.7, as is expected. To tackle this situation, memristors with much higher HRS/LRS ratio should be used so that even after a few million cycles, a good separation between HRS and LRS values exists to differentiate. Besides, there are also memristors where HRS and LRS do not change much with time and thus their relative difference is still intact after many cycles.

In addition, there are no floating rows as the read voltage is applied across all rows simultaneously and each column is measured using a load resistance, unlike in a crossbar memory where a single memory element needs to be read. Therefore, there is no negative impact of sneak-path current [12] in our crossbar circuit implementation as there are no unselected cells. For these reasons, we expect the reliability of the fabricated XORed XbarPUF also to be very close to its desired value if the variation is improved.

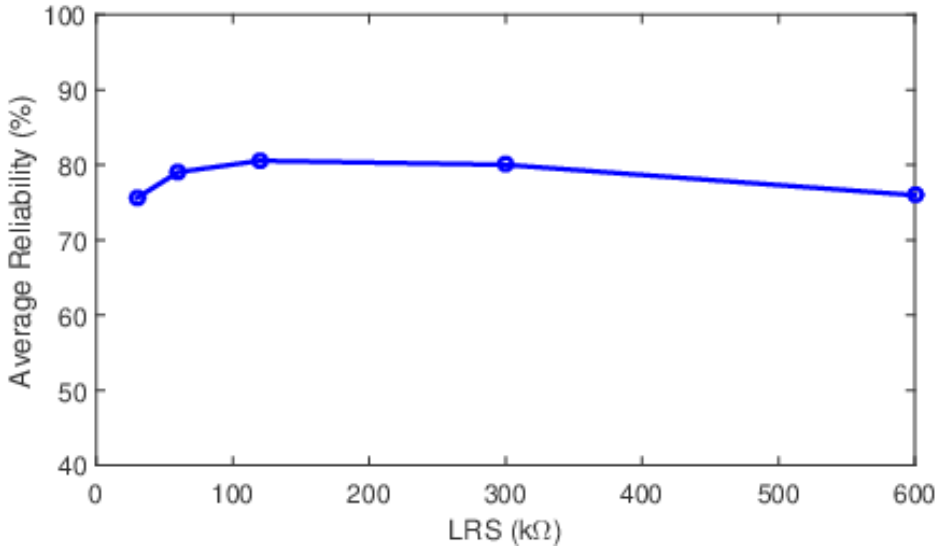


Figure 4.6: Reliability of the XbarPUF with respect to increasing LRS [69].

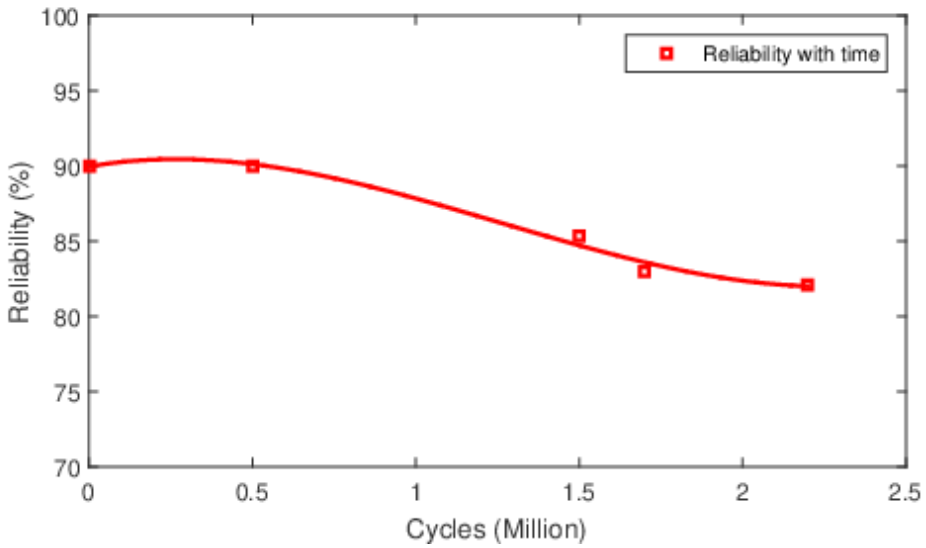


Figure 4.7: Reliability of the XbarPUF with age [69].

4.1.4 Performance Overhead Analysis

Change in power consumption for the circuit with respect to LRS and HRS values are shown in Figure 4.8. We have simulated and evaluated power for six different HRS/LRS ratios, each with five different sets of HRS and LRS values. The leftmost and topmost point in Figure 4.8 represents the power consumption with the current HRS and LRS values considered achievable for HfO_x memristors. Therefore, it is evident that there is room for further reductions in power consumption by improving device characteristics, something we expect to occur as the technology matures. Both an increase in HRS/LRS ratio and any increase in LRS (and HRS to keep the ratio same) result in a reduction in power. However, larger reductions are found at HRS/LRS ratios less than 100, beyond which the power curve tends to flatten out. With increasing LRS, the advantage in power reduction with increasing HRS/LRS ratios also reduces.

Memristors are very small compared to CMOS transistors and can be placed at each cross-points of a crossbar array. Therefore, the crossbar representation requires minimal area from the perspective of CMOS. Since the XORed circuit generates one response bit from four columns instead of two as was the case in XbarPUF without XORing, its area is double in size compared to an XbarPUF with no XORing. Only a few transistors are added to the whole circuit if column shuffling is used. Figure 4.9 shows the area comparison between XbarPUF and CMOS-based arbiter PUF (APUF) in terms of transistor count. The area of the memristive XbarPUF increases linearly with an increase in rows and/or columns while in the case of the APUF, the increase in area is exponential.

The delay of this circuit is the minimum clock cycle time that can be used to sample the response bits faithfully. This minimum clock cycle time is constrained by the longest time needed for a memristor to switch to either the high or low resistance state. In our model the switching time is 1μs. There is also a time needed to apply a challenge to the crossbar PUF and then get a valid response. From the simulation, we find this delay to be $25.2n$s which includes the sampling time using another clock. Thus, the delay of our XbarPUF circuit would be a function of the switching times of memristors and the time required to apply a challenge and get a valid response. It is worth noting that other memristor material stacks

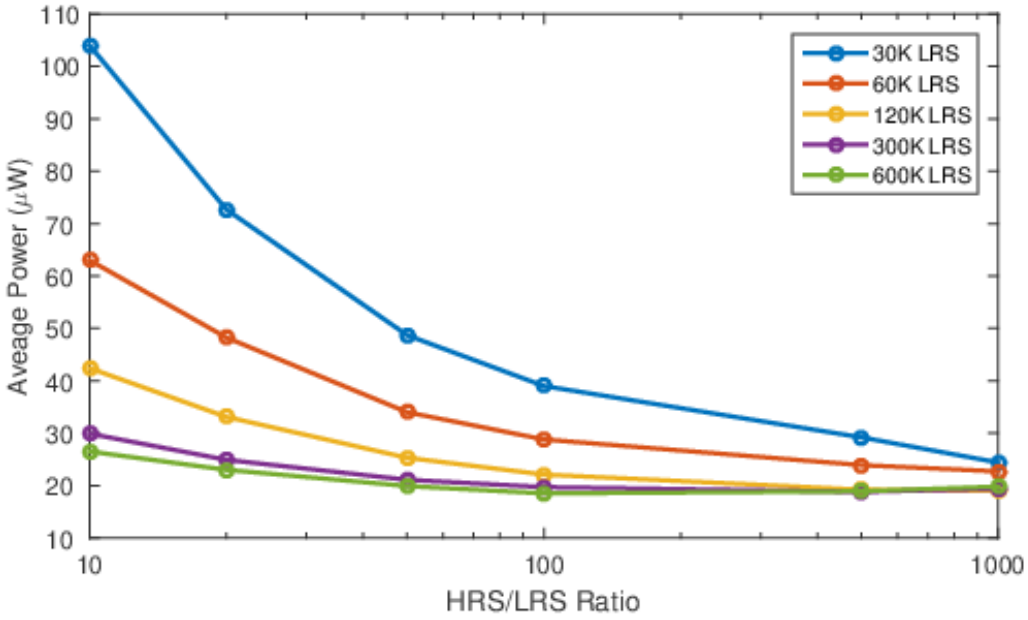


Figure 4.8: Dependence of average power on HRS/LRS ratio and the absolute values of HRS and LRS [69].

(such as [80]) have exhibited faster switching times so the delay of the XbarPUF is expected to improve as technologies mature.

4.1.5 Overhead analysis for Other Memristors

Since we have created an abstract high level model of the XbarPUF, we have also performed overhead analysis which otherwise would take a really long time to simulate and generate these different versions of XbarPUF for different memristor materials from a transistor level simulation. To measure power consumption, we have also modeled current and power consumption in our high-level XbarPUF model. First, we have computed the approximate power consumption of our proposed design and evaluate its applicability as a lightweight hardware security primitive. Since memristors are basically resistive devices, static current flows through the circuit. All the memristors and series transistor in these current paths consume static power. Moreover, the peripheral CMOS digital gates consume dynamic power. To verify our high-level power model of XbarPUF, first we have calculated power consumption of different sized XbarPUF from both transistor level simulation (from Cadence

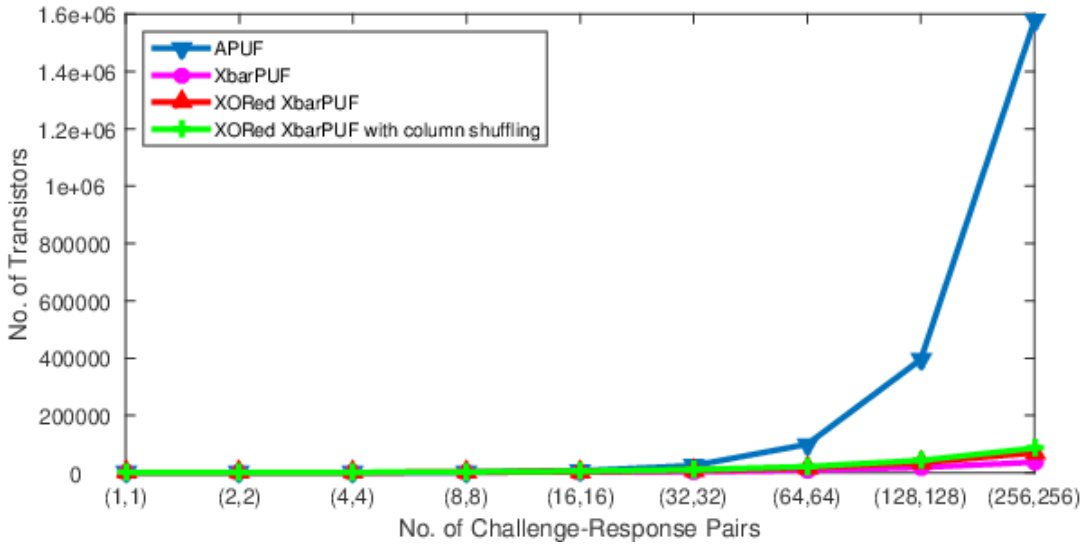


Figure 4.9: Area comparison of XbarPUF with Arbiter PUF [69].

Spectre) and from our high-level abstract model (from MATLAB). The results are shown in Table 4.3. Values obtained from MATLAB model are slightly lower than power consumption from transistor-level simulation as we have neglected dynamic power consumption for some control logic. However, as we can see from this table, the estimated power consumption values are consistently close to values from transistor simulation and follow the same trend. Thus we have used our approximate high level power model for the rest of our experiment to get a sufficient amount of data in a relatively short period of time.

So far in this work, HfO_x memristor has been used for all simulations. Now, we have also evaluated the XbarPUF performance for two other memristor materials too. Parameters

Table 4.3: Power estimation of a HfO_x memristor based XbarPUF from MATLAB and cadence [71]

Crossbar size (chall×resp)	**Average Power (mW) (MATLAB)**	**Average Power (mW) (Cadence Spectre)**
4×2	0.25	0.26
8×2	0.51	0.61
16×2	0.98	1.01
32×2	2.07	2.22

used for three memristors, HfO_x, TaO_x, and TiO_x (hafnium, tantalum, and titanium oxide) are listed in Table 4.4. Next, we have estimated the static power consumption of a 32×2 XbarPUF in its three different phases of operation for these three different memristors. The results are shown in Table 4.5. The RESET phase consumes the most amount of power as all the memristors are reset to HRS during this time. Although, the overall memristance might be lower during READ and CHALLENGE phases, since higher reset voltage is applied during RESET phase, the power consumption is also the highest. Our designed PUF stays most of its time in READ and RESET phase and thus these two phases dominate the overall average power consumption. As the crossbar size increases, the static power of the memristors dominates the overall power consumption.

Static power is directly dependent on the memristance and applied voltage magnitude. Because of its much higher HRS and LRS values, XbarPUF based on TiO_x memristor as listed in Table 4.4 consumes the least amount of power among these three memristors. Moreover, from Table 4.4, we can also see that TaO_x and TiO_x memristors have smaller threshold voltages which facilitates the use of lower read-write voltages. Thus the read-write voltages of HfO_x memristor are 0.6V and 1.3V, respectively while they are 0.3V and 1.0V, respectively both both TaO_x and TiO_x memristors. Because of its very small LRS and HRS values, TaO_x memristors consume much larger amount of power compared to the other two. When LRS/HRS values are small, with increasing crossbar size, the equivalent memristance or resistance of a crossbar column would become comparable to the on resistance of the peripheral CMOS transistors and thus CMOS would also contribute to a larger amount of static power consumption. Finally, since the values in Table 4.5 are heavily influenced by the choice of read-write voltage and memristor's parameters especially memristance and threshold voltages, these numbers are not the only representative of these types of memristors materials. Rather, each should be analyzed separately depending on the application.

We have also shown how the power consumption changes with a change in crossbar size. Figure 4.10 shows the relationship between static power consumption with increasing number of response bits (and hence number of crossbar rows) for all three memristors. XbarPUF with TaO_x shows a faster increase in static power compared to the other two memristors, HfO_x and TiO_x due to its small LRS and HRS values. It also limits the scalability of the

Table 4.4: Switching Parameters for Metal-Oxide Memristors [71]

Parameters	HfO_x [72]	TaO_x [80]	TiO_x [48]	Variation
HRS	300KΩ	$10k$	$2M$	±20%
LRS	30KΩ	$2k$	$500k$	±10%
V_{tp}	0.7V	0.5V	0.5V	±10%
V_{tn}	-1.0V	0.5V	0.5V	±10%
t_{swp}	10ns	105ps	10ns	±5%
t_{swn}	1μs	120ps	10ns	±5%

Table 4.5: Power Consumption during different phases of a 32×2 XbarPUF [71]

XbarPUF Phase	**HfO_x (mW)**	**TiO_x (mW)**	**TaO_x (mW)**
RESET	3.20	0.27	51.40
CHALLENGE	2.10	0.007	42.20
READ	0.60	0.014	1.10

crossbar for any memristor with smaller memristance values. Added mitigation technique would increase power consumption for the same number of challenge-response bits. Column shuffling block would increase the power consumption slightly where the XORing technique would double it because the effective number of response bits decrease by a factor of 2 with XORing. However, still XbarPUF displays a relatively much low power consumption compared to most lightweight hardware security primitives.

In previous chapter, we have presented area vs crossbar size results for an XbarPUF based on HfO_x memristor and it is shown again. All memristors considered here are very small and can fit in each cross-points of two metal layers, thus effectively requiring zero area. Therefore, changing memristors wouldn't ideally change the required area although the peripheral size is affected by the memristor's parameters. Thus, the only area overhead is the peripherals, where the area increases with increasing number of crossbar size. The XbarPUF with added mitigation technique requires larger area but still the overall area should be much smaller compared to a CMOS PUF like arbiter PUF or RO-PUF.

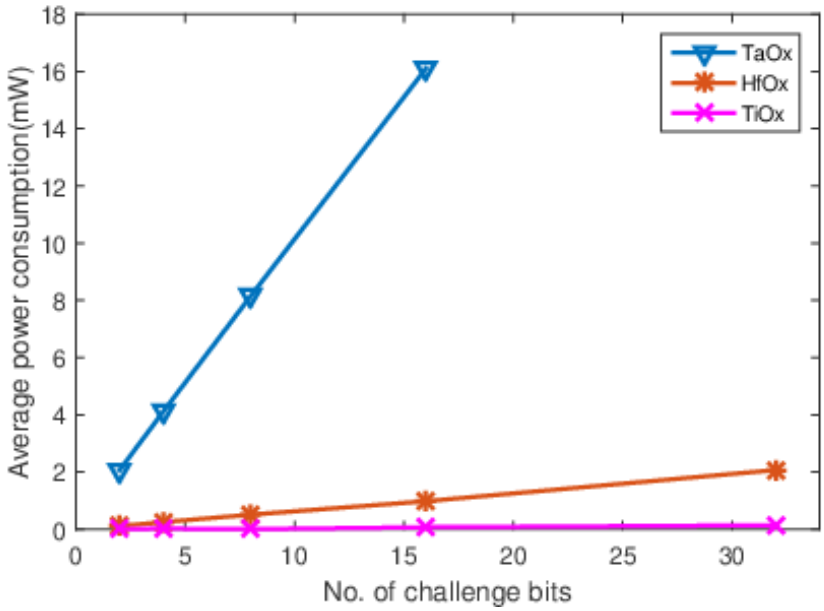


Figure 4.10: Relationship between average static power consumption and size of the XbarPUF [71]

4.1.6 On-Chip Design

To evaluate how the HfO_2 memristor based crossbar PUF behaves, we have designed small test structures to measure its performance. We have implemented a 1-bit memristor crossbar PUF in 65nm CMOS technology along with HfO_2 memristor. The schematic and layout for this circuit are shown in Figure 4.11 and 4.12, respectively. The schematic is not very clear from this image, but four memristors, their read-write-form circuitry, row and column logic of the PUF, and a sense amplifier at the bottom are clearly detectable. The layout dimension is 19.59μm $\times$ 33.63μm (excluding the metal layer extension for routing).

4.2 Robustness Analysis Against ML (Machine Learning) based Attacks

As stated, a PUF response results from a complex function dependent on internal process variations inherent to specific chip implementations. Therefore, it is very difficult and time consuming to predict a PUF response based only on the challenges provided. However, if

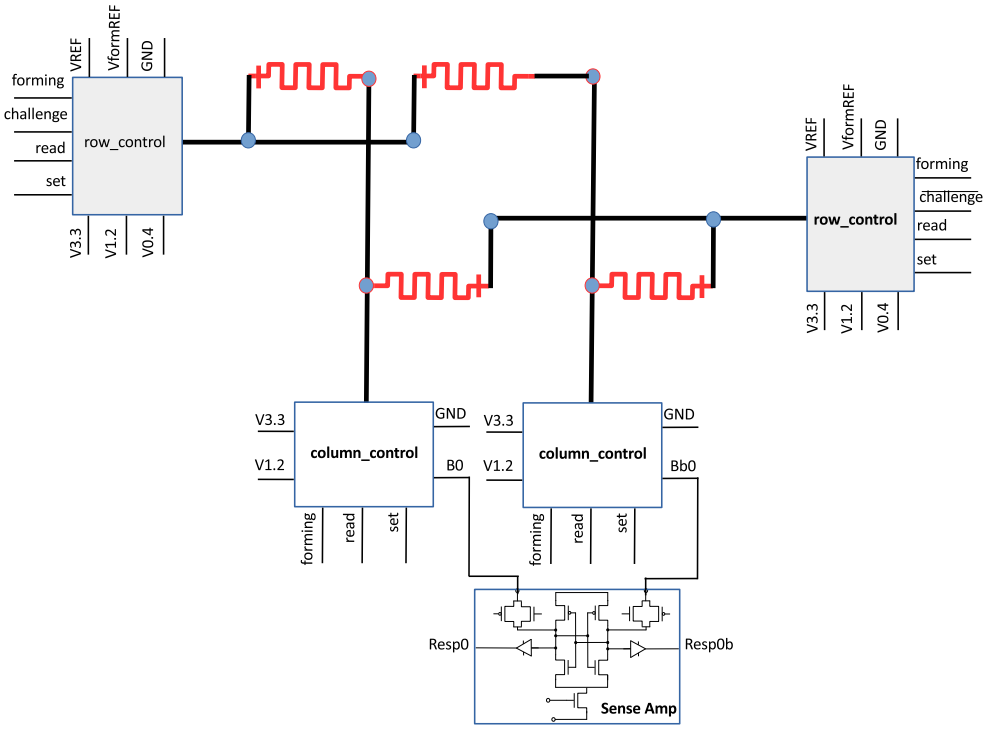


Figure 4.11: Schematic of the designed 1-bit memristor crossbar PUF

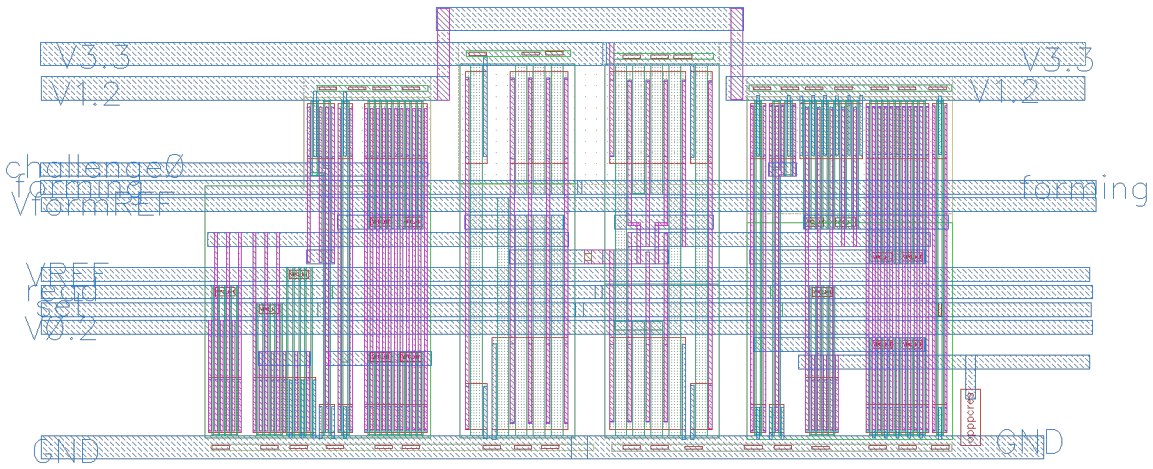


Figure 4.12: Layout of the designed 1-bit memristor crossbar PUF (19.59μm×33.63μm)

the internal functionality of a PUF is not complex enough, then using a machine learning approach such as any classification algorithm, it is possible to model a PUF and predict the response with high accuracy. In [62], a modeling attack on an arbiter PUF was shown to achieve 99% accuracy with sufficient CRPs. A modified APUF with XORing, although requiring a higher number of CRPs, was also shown to be modeled very accurately. High modeling accuracy was achieved for an APUF due in part to the fact that an APUF can be expressed with a linear additive delay model and a linear classifier can be used for modeling purposes. Since a PUF response is binary, individual PUF responses can be modeled with a binary classifier. XbarPUF is very similar in behavior to the APUF, because like APUF, XbarPUF also consists of a number of delay elements, implemented by leveraging the memristors' switching rates. Thus, it is also expected that XbarPUF would be vulnerable against these machine learning based modeling attacks. We have used most commonly used machine learning algorithms already implemented and optimized for data analysis problems by researchers and computer scientists.

We have created an equivalent behavioral model of a memristor and XbarPUF in MATLAB. It is much faster to simulate a high level behavioral model than simulating a large circuit at the transistor-level. Therefore, it's easy to construct large crossbar arrays and generate responses. However, it is essential that our MATLAB model imitates the actual transistor-level (Cadence Spectre) model accurately. Therefore, we have verified our MATLAB model by simulating several different circuits in Cadence to compare their performance and verifying that they behave similarly. In our previous work [72], the memristor crossbar size was only 4×2. It is impractical to use such a small circuit for any kind of machine learning based modeling attacks as the total number of possible inputs is only $2^4 = 16$. Therefore, we have used XbarPUFs with larger numbers of CRPs so that only a small subset of all possible combinations of CRPs are used in our experiments.

4.2.1 Development of High Level Behavioral Model of Memristor

In prior work demonstrating the XbarPUF [72], we have used a Verilog-A model for a hafnium-oxide memristor adapted from a model presented first in [47]. For this work, we have also created an equivalent high level model of a memristor to use in MATLAB based

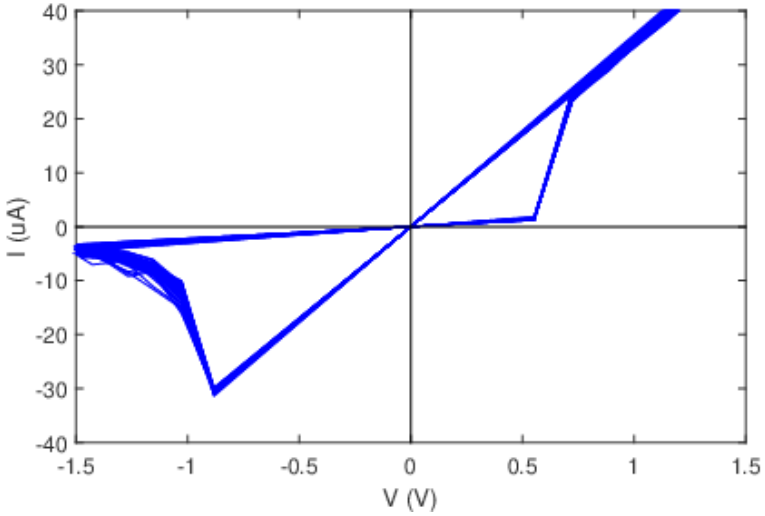


Figure 4.13: Simulated IV characteristics of a metal-oxide (HfO_x) memristor for 200 set/reset cycles for a 100kHz clock. [71]

simulations. At each simulation time step, memristance at the next step is calculated from the memristor's current state based on the time duration and magnitude of an applied voltage bias. Details of this switching operation can be found in [72]. As mentioned before, there are six primary parameters in our memristor model, specifically: HRS, LRS, V_{tp}, V_{tn}, t_{swp}, and t_{swn}. As necessary, parameter values are recalculated on each iteration using their mean and cycle-to-cycle variations for any particular memristor. Figure 4.13 shows the IV characteristics of a memristor for 200 set and reset cycles. All parameter values are allowed to vary from one cycle to another, even for a single memristor. The various hysteresis loops in Figure 4.13 demonstrate cycle-to-cycle variations for a single modeled device.

4.2.2 Development of High Level Behavioral Model of XbarPUF

The memristor crossbar PUF circuit consists of two main components, a memristor crossbar array and peripheral read-write circuit. A crossbar is simply a 2-D array with memristors at each crosspoint. A matrix of size $row \times col$ is created where each component is represented with a memristor 'class' in MATLAB. Using respective means and standard deviations for all the memristor parameters, and assuming a Gaussian distribution function, each memristor

is assigned to different random initial values for its parameters. This is the step where die-to-die variations of memristors are mimicked in the simulation.

As mentioned, three phases of operations are performed on the memristor crossbar. A vector with its length equal to the number of rows in the crossbar is created to represent the voltage applied to each row. During RESET, all these voltages are equal to -V_{wr}. Each memristor is reset to HRS and, therefore, the effective memristance across an entire column is at its maximum value. The load resistance is chosen such that it has minimum drop during this reset period and maximum drop during the challenge. During the CHALLENGE phase, either a positive V_{wr} or zero voltage is applied to each row depending on the state of the corresponding challenge bit. If the challenge bit is high, then memristors in that row start to set toward LRS and the voltage drop across the load starts to increase. Otherwise, any corresponding memristor is kept at HRS since the applied voltage is 0V. In the READ phase, all the memristors in a column are in parallel with the applied voltage V_{rd}. This effective column memristance is in series with the column's corresponding load and a simple voltage divider determines the load voltage. The load voltage from two side-by-side columns are compared to generate a response with '0' for one column and '1' for the other. Which column first hits the '1' state determines the column that "wins" the race, thus generating the corresponding response bit. For an XORed XbarPUF, two response bits (generated from 2 column pairs) are logically XORed to determine one final bit.

4.2.3 Introduction to Machine Learning Attack Models

We have used python's scikit-learn package [54] to implement several different machine learning models. Specifically, we have performed classification or modeling of PUF responses using logistic regression, support vector machine (SVM) with different kernel functions (linear, polynomial, RBF (radial basis function), and sigmoidal), k nearest neighbour (kNN), Naive Bayes (Gaussian, and Bernoulli), stochastic gradient descent, random forest, ensemble methods (bagged tree, and AdaBoost), and neural network (perceptron). All of these are very standard machine learning techniques and details about their theory, mathematics, and algorithms can be found in [54] and, thus, skipped here for brevity. Only the results from best few models are provided in this work.

4.2.4 Experimental Setup

We need a large enough crossbar with a sufficient number of challenge response pairs (CRPs) to build an attack model. For this particular work, we are only interested in the final CRPs, not the actual internal structure of the XbarPUF itself. We have experimented on an XbarPUF of size 32×2 in terms of CRPs for this work which contains 64×4 memristors. The idea is that an adversary would gain access into the system and can collect a small subset of all available CRPs. For the most part of the experiment, we have kept the number of CRPs to 5000 which is small compared to the total number of possible combinations of input (2^{32}). The read, write voltage and clock frequency are constrained by the particular memristor technology. In this work, we choose the read and write voltage to be 0.6V and 1.3V, respectively because the HfO_2 memristor that we have considered has threshold voltages of 0.7V and -1V in positive and negative switching direction, respectively. The clock frequency should be kept higher than the memristor's switching time so that a memristor require multiple clock cycles to switch from one state to another. This would ensure that little differences in switching speed between two memristors due to process variation could be detected with more confidence. For this work, the memristor's switching time is considered to be 1μs and we have used a clock with clock period of $100n$s (10 times faster).

Before the beginning of each CRP dataset collection, the memristors are instantiated randomly. First using the statistical mean and variance for each memristor parameter, a full crossbar of memristors are generated from a normal random distribution. Then a set of random challenges are applied and 2-bit responses are collected for all different versions of XbarPUF, without any mitigation technique and with different mitigation techniques applied as presented next. CRPs are saved in a database file with each row represents one data-point or CRP containing all challenge and response bits.

4.2.5 Modeling Attack on an XbarPUF

Table 4.6 shows the success of modeling attack against an XbarPUF without mitigation. As suspected, it is very much vulnerable against such attacks with a modeling accuracy of over

Table 4.6: Results from modeling attacks on basic XbarPUF (no mitigation) for different machine learning algorithms

M.L. models	**SVM (RBF)**		**L. R.**		**Gauss. N. B.**		**AdaB. Ensem.**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
Accuracy (%)	80.09	79.10	**99.96**	**99.33**	91.29	90.45	98.55	97.99

99%. Thus, next we are going to design some mitigation techniques to increase resiliency against these attacks and reduce the accuracy.

4.2.6 Mitigation of ML Attacks

We have proposed techniques to mitigate these attacks by introducing some form of non-linearity in the circuit. The first and well-known technique to XOR the responses with each other to produce different responses. However, this is not sufficient against stronger attack model with larger number of training data. That's why we proposed another mitigation technique called 'column shuffling' which shuffles the connection between top part of a crossbar to its bottom part based on some selector challenge input. In this particular work, we have proposed a four column shuffling or swapping block with three selector bits controlling the shuffling logic. An example implementation of such shuffling logic block is provided in Table 4.7.

There are some restrictions on the way this shuffling block to be implemented. Boolean logic minimization without keeping an eye on these requirements would change the XbarPUF behavior unexpectedly. Because these shuffling blocks are used directly in path of response generation, they can not introduce any uneven delay on these paths. Because that would introduce bias and could deteriorate the security properties. TO ensure, memristors, not these blocks are the only source of entropy for the XbarPUF, we have designed the logic in such a way that the logic path delay introduced by these blocks are equal on all four crossbar columns. Thus we have only used a subset of all possible combinations of $2^4 = 16$ for 4-columns. Moreover, current flows in both directions in the crossbar and thus these logic should be symmetric over both directions of current flow. Keeping all these in mind, we have implemented the shuffling operation as shown in Table 4.7. Figure 4.14 shows one

Table 4.7: Column swapping logic implemented in this demonstration [71]

S1	S0	y3	y2	y1	y0
0	0	x3	x2	x1	x0
0	1	x3	x2	x0	x1
1	0	x2	x3	x1	x0
1	1	x2	x3	x0	x1

S2	z3	z2	z1	z0
0	y3	y2	y1	y0
1	y1	y0	y3	y2

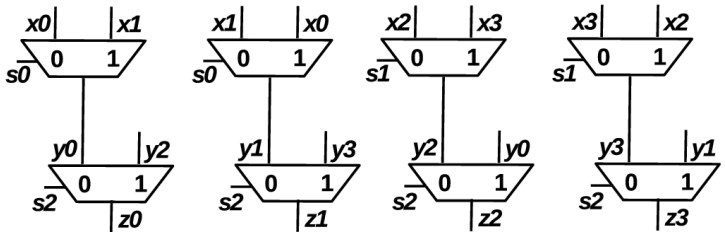


Figure 4.14: An implementation of column swapping circuit [71].

possible implementation. Three selector bits are needed as control input for this block and can be considered as extra challenge bits for the PUF. For an $M \times N$ XbarPUF in terms of no. of rows and columns, it would require $M/4$ number of column shuffling blocks. All of these blocks can have different selector inputs or they can all share the same selector bits as well. For simplicity and to reduce pin count, we have only considered the case when all these 4-column groups share the same selector bits for their column shuffling block as shown in Figure 4.15. Adding this column shuffling or mixing block with additional challenge bits would increase the complexity of any learning algorithms considerably and in next sections, we'll analyze the impact of this modification.

4.2.7 Results and Discussions

Table 4.8 presents the modeling attack results for all different XbarPUF versions i.e. XbarPUF with no mitigation, XOR mitigation, column shuffling, and XOR+column shuffling mitigation. This table lists the four best modeling attack results from several different machine learning/classification algorithms found in scikit-learn [54]. As mentioned earlier, basic XbarPUF with no mitigation technique is very vulnerable and can be modeled with more than 99% accuracy with logistic regression. As we add non-linearity in the design,

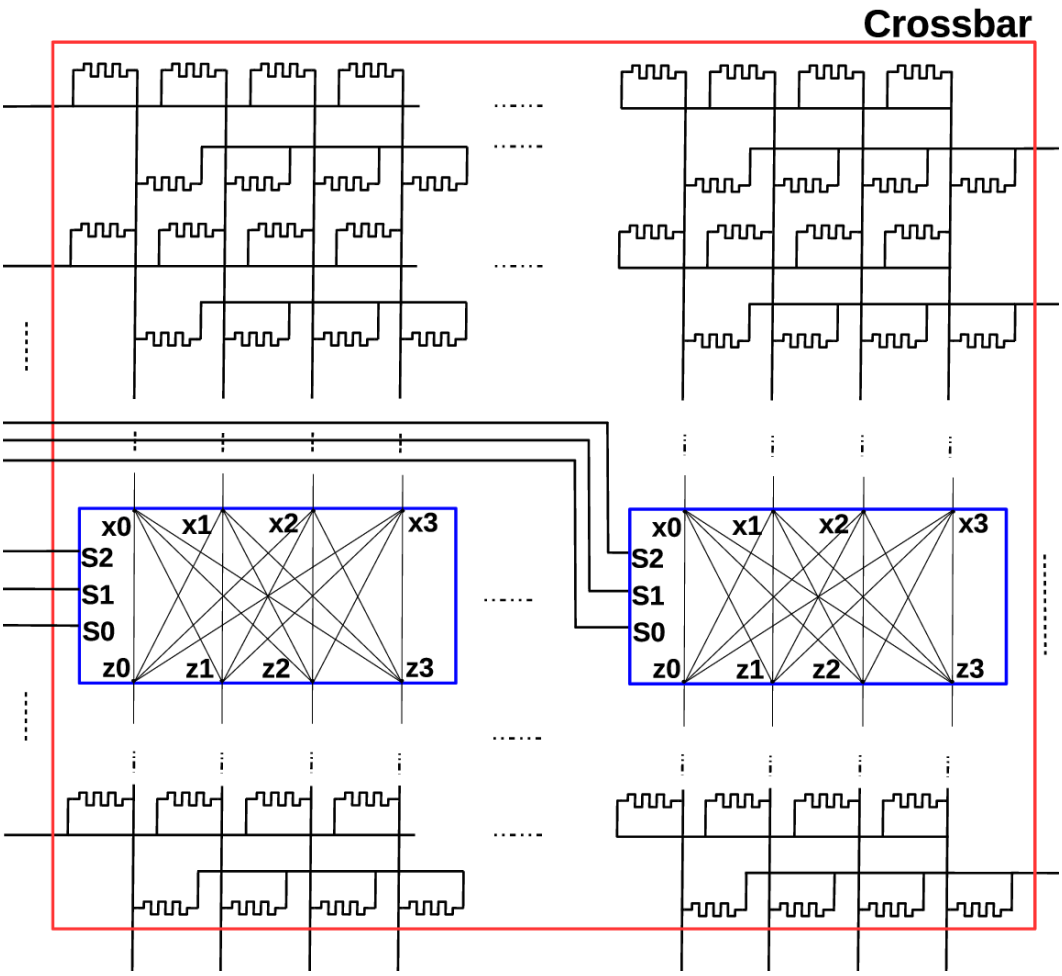


Figure 4.15: XbarPUF with the inclusion of column swapping logic [71].

Table 4.8: Results from modeling attacks after mitigation techniques applied for different machine learning algorithms

Mitigation Technique	**SVM (RBF)**		**L. R.**		**Gauss. N. B.**		**AdaB. Ensem.**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
None	80.09	79.10	**99.96**	**99.33**	91.29	90.45	98.55	97.99
XOR	66.29	63.82	**78.30**	**77.01**	76.87	75.68	76.84	75.70
Swap	**77.97**	**71.81**	66.93	64.38	66.63	64.64	66.71	64.37
XOR+Swap	**52.50**	**49.64**	54.62	49.19	54.59	49.20	54.56	49.18

this accuracy should drop. Thus both XORing and column shuffling technique reduce the modeling accuracy, with column shuffling presenting a higher resiliency against modeling compared to XOR. However, as we can see from the last row of this table, when we incorporate both of these mitigation techniques together in the XbarPUF design, the modeling accuracy drops to almost the probability of a random coin flip (50%).

So far we have evaluated the robustness of the regular XbarPUF and its 3 variants with different mitigation techniques against four different modeling attack for 5000 data-points. Now, we have performed experiments to observe how much the modeling accuracy changes with increasing number of data-points. Figure 4.16 and 4.17 show the modeling accuracy vs no. of data-points for LR and Gaussian NV (Naive Bayes) model, respectively for all four versions of the XbarPUF. We picked these two algorithms as representatives of linear and non-linear classification algorithms. Both of these models achieve very good accuracy with a small number of data-points (and features), its accuracy doesn't change or change slowly with increasing number of data-points. In [62], modeling accuracy of LR model vs. dataset size was analyzed against arbiter PUF and was shown to vary very slowly which further justifies our findings here. Since GNV uses a non-linear kernel with large number of features, its accuracy is expected to increase with increasing no. of data-points. It is evident from the first three lines of Figure 4.17. However, from Figure 4.17, for the combined mitigation technique of XOR+column shuffling, the accuracy of bot LR and GNV model do not improve at all with increasing no. of data-points and stay near random probability of near 50%. This proves that the XbarPUF with our designed mitigation technique is robust against both of these modeling algorithms.

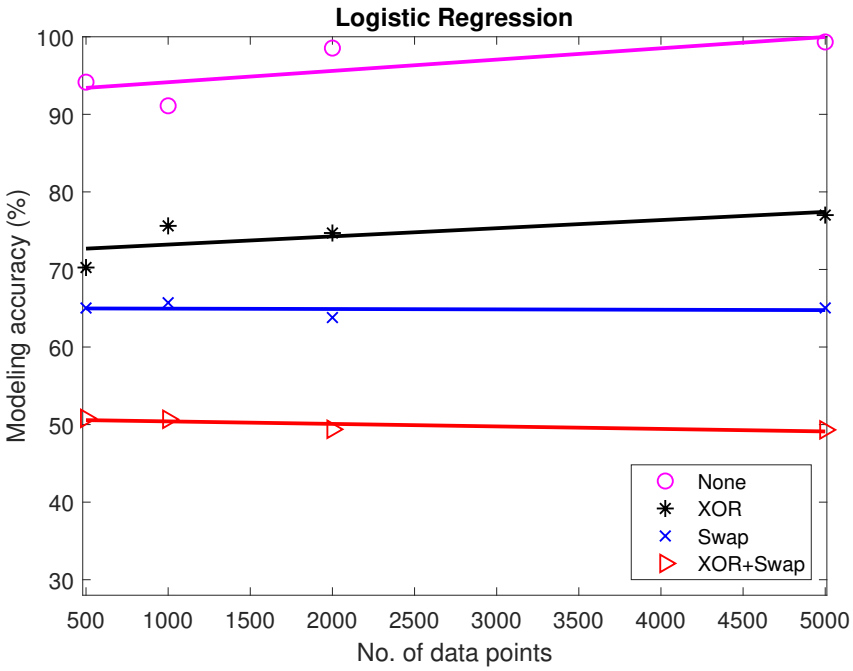


Figure 4.16: Modeling accuracy of LR vs. the size of dataset [71].

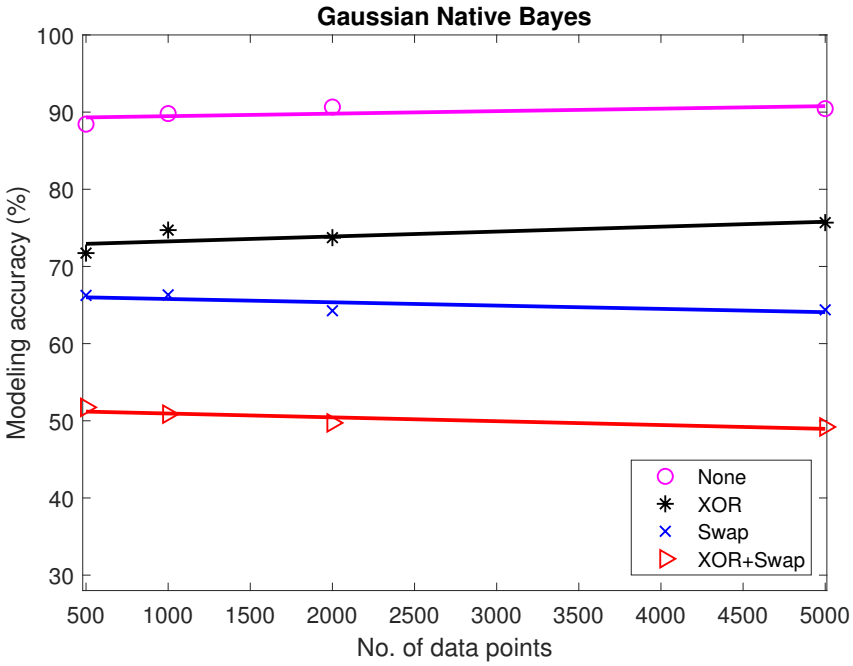


Figure 4.17: Modeling accuracy of Gaussian Native Bayes vs. the size of dataset [71].

Chapter 5

Lightweight Security for IoT/Embedded System

5.1 Motivation

In the era of connectivity and smart technology, there is a plethora of small devices that connect to the Internet and other networking sources with very limited security features. Many of these devices may contain time-sensitive data. Most of these devices run on battery with very limited power available to them and some are even batteryless, dependent on harvested energy alone [38]. Therefore, to save as much power as possible and also due to their intermittent nature of data sensing operation, they often go to sleep or ultra low power mode before backing up their sensitive states and other information for data forwarding [37]. Now, these backup data are usually saved in NVM and are vulnerable to probing and malicious read. In this work, we are mainly focused on providing a security for the backup data in these resource constraint devices. Now, since the data in an IoT device are mostly temporary, the information is no longer useful after a certain period of time. Therefore, their security requirements are relaxed in terms of complexity of code-breaking algorithms but has to be very lightweight in terms of area occupied, power consumed and delay introduced. In this work, we have focused our attention on developing an ultra lightweight security for the backup data of these devices.

5.2 Security vulnerability of IoT processor

We have considered two types of resource-limited embedded processor, typically common in an IoT system, as shown in Figure 5.1. First, a regular battery-operated or very low power processor may employ very aggressive power power saving technique and often go into ultra low power mode, before backing up necessary data in an unprotected NVM. Second, for energy-harvesting devices, since they lose all information when there is a power failure, they back-up their necessary information for computation in an unprotected NVM as well. Our idea to secure this unprotected NVM so that an adversary can not gain any information by reading this data or any illegal modification is detected by the processor. Therefore, a secure restore and back-up protocol has been developed.

First, a random vector is generated from a TRNG and used as a challenge to an on-chip PUF. The PUF can generate a unique random response which is used as a cryptographic to encrypt the data before back-up into an NVM. The challenge is also saved in the NVM and read again during wake-up to generated the same PUF response or cryptographic key again. This key is used to decrypt and restore the data. This is illustrated in Fig, 5.2. Since the same PUF can generate different keys in different devices, our design also provides device-specific security.

5.3 Our Security Solution

Since we are concerned about providing security for a non-volatile memory which may contain time-sensitive data, we need to provide some form of encryption which would provide good security as well as be lightweight in nature. For that purpose, a lightweight PUF based encryption is designed. We have considered the memristive crossbar PUF or the XbarPUF for this purpose which would provide the cryptographic key. At the beginning of each backup operation, a random challenge is applied to the XbarPUF. Then it generates a random response depending on the challenge which is used as the encryption key. To reduce heavy overhead of complex encryption engine, we are only proposing to use an XOR block which would XOR the data to backed up with the PUF key. Then this XORed or encrypted data is

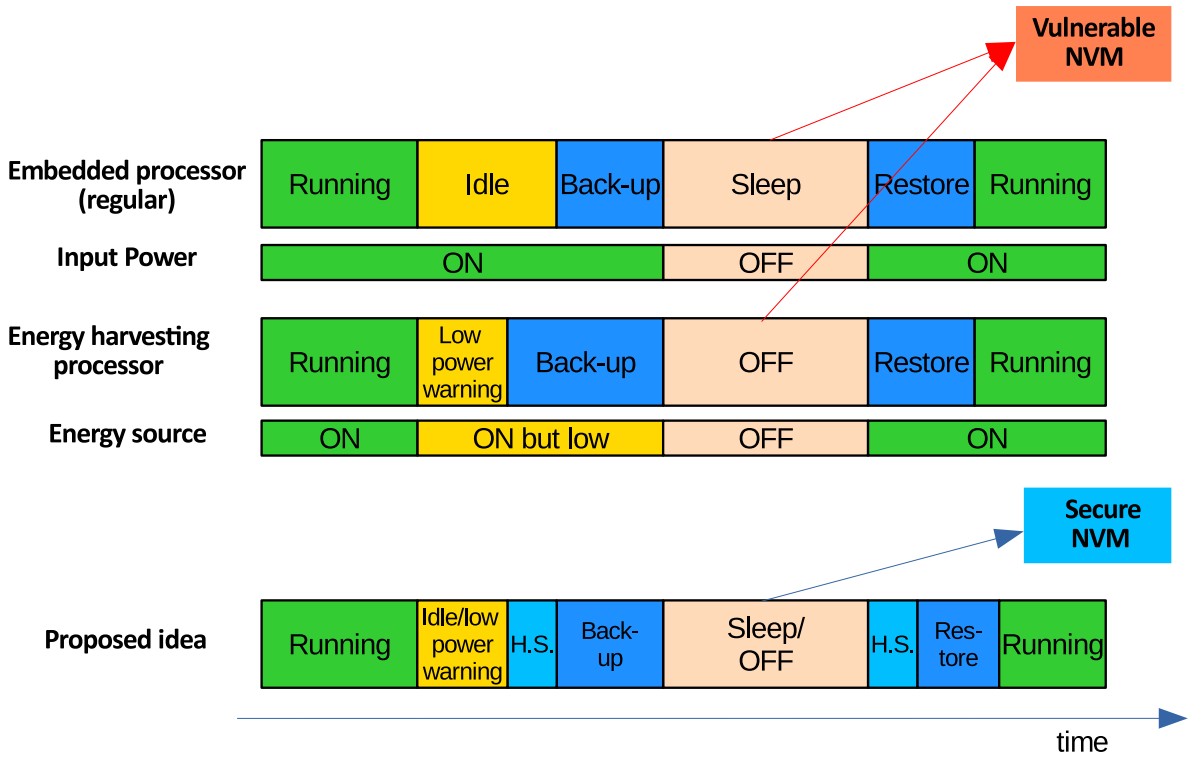


Figure 5.1: An embedded processor (either regular or energy harvesting) backs-up data for data-forwarding but the data left on NVM is unprotected.

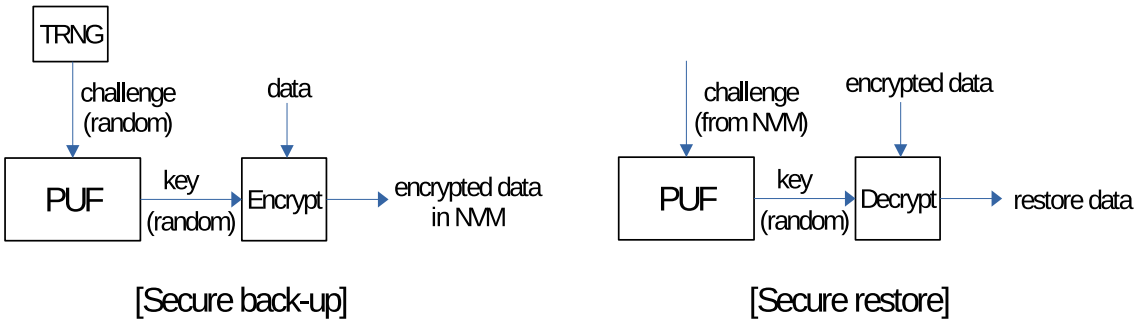


Figure 5.2: A unique random challenge is applied to a PUF to generate a cryptographic key for secure back-up and restoration of data in an embedded processor.

saved in an RRAM or memristor based memory. On each back-up, a new PUF key would be generated by applying a new challenge and thus essentially implementing an one time pad (OTP) system. Since OTP is theoretically the most secure, our system is also very secure. During a wake-up, the same challenge is applied to the XbarPUF again to generate the same key which is then XORed with the encrypted data to regenerate the original data.

Since reliability is issue for all PUFs and an IoT employed in the field might subject to larger temperature change, rail voltage and other noise, the reliability is even a bigger concern where the PUF response is used as the key. Thus the PUF must be improved to have a close to ideal reliability and also some kind of error correction needs to be implemented. Since traditional error correction code (ECC) introduces a large overhead, employing those would defeat the purpose of using PUF for its low overhead compared to traditional cryptography. Since XbarPUF is robust against environmental changes due to its relative nature of computation, it is very useful in this type of IoT system. However, because of memristor's cycle-to-cycle variation, there would be some unreliable bits in the response depending on the challenge applied. To overcome this, we have used a simple majority check to discard these erroneous bits for a particular challenge. Our idea is to use two small SRAM blocks for this purpose as seen in [cite]. First, a random challenge is applied to the XbarPUF and the response is saved in an SRAM. Then the same challenge is applied again (and maybe in a different environmental condition) and the response is saved in another SRAM with same location for a particular bit position. Then the contents from these two SRAMs are compared by a simple XOR/XNOR function. If two bits from the two SRAMs are the same, then it is a valid response bit and used as one bit of the encryption key. If the bits are different, then it is discarded. This system is thus not exactly an error correction method, rather a simple error minimization system.

During wake-up, when power is available for a resource-constraint system, the same challenge as before is applied to the XbarPUF twice to generate a fresh response key again. Then encrypted data from the RRAM is read bit by bit while contents of the SRAMs are read and checked for valid key bit and XORed with the encrypted data for decryption and then saved back to appropriate registers again to resume normal processor operation.

This whole PUF key generation, data encryption and decryption can be incorporated in a interrupt routine with a processor. Specifically, we can tie these operations with the highest level interrupt of a processor, like the low power/voltage warning. Thus for a energy harvesting system, when the power is unavailable or for a battery powered device, when the system wants to go to sleep mode, the low power warning interrupt might be enabled to perform all these operations before power failure or sleep. During wake-up, the decryption process is done first before resuming regular processor operation again. The different key components of this system are discussed in the following sections.

5.4 Our Security Protocol

The purpose of this design is to provide security for the unprotected persistent or non-volatile memory of any low power embedded systems or edge devices. During a sleep or low power mode or during a power failure for energy harvesting processors, this memory is vulnerable and our designed security architecture is active during this time. This security mechanism depends heavily on the XbarPUF based cryptographic key generator and a reliability enhancement block. Our idea is to generate a reliable and unique cryptographic key each time a backup operation is needed, thereby our system, effectively, is an implementation of one-time pad. We have some basic assumptions or criteria for our design. First, the key should be random. Since the response of a PUF is random by definition, the key extracted from it is expected to be random as well. Second, even with the same PUF circuit, different devices would produce unique keys compared to each other. This is the beauty of a PUF based system is that they are unclonable and unique by nature and an adversary won't be able to break the security of another chip even if he gains access to all the keys from one chip. This is very helpful for IoT domain since there are billions of devices out there and with traditional cryptography, breaking one device would compromise the security of these devices. Third, during each back-up, a new key is needed. A strong PUF with sufficiently large CRPs can provide unique keys throughout the lifetime of a device. Besides, since we only want to use one key once per encryption and generate a new key during another time, the reliability requirement of the PUF can be relaxed in this application. PUF response only

needs to be stable during one cycle of encryption and one cycle of decryption. Moreover, if a response remains stable during one encryption-decryption and changes over long time due to aging or other factors, it would actually increase the possible number of unique responses even for the same challenge over time, thereby increasing available number of keys over the lifetime of such devices. To fulfill the requirement of OTP, the key should be as large as the data itself. This is usually difficult to meet for larger systems, however, could be practical for small IoT systems needing to only back-up states and other information. If a single PUF response is not large enough to generate a big enough key for the data, then we can generate multiple much responses to create a key as large as the data itself.

We are also assuming that data is only valid and important for a short period of time for the type of system that we are targeting. When the processor wakes up from a sleep mode or when an energy harvesting processor resumes working on its stored data after a power failure, previous data like program states, temporary data, and other information are no longer useful. Thus we only need to provide security for the time being when data is left unprotected on the NVM during a sleep or power failure. Therefore, any rigorous cryptographic solution which can theoretically be unbreakable for hundreds or thousands of years is not essential in this case. Thus our goal to provide security for a practical time limit for such devices. Hardware security is the preferred choice because of its lightweight nature which is a must in this domain.

We have shown in this work how we meet all of these aforementioned requirements for an IoT system. Since the RESET phase of an XbarPUF consumes relatively much large amount of power compared to other states in the security architecture, multiple smaller XbarPUFs are used and activated one by one in a time-multiplexing fashion and their responses are augmented together to create a large enough response. Since some bits of this response vector might be unreliable and prone to flipping, a reliability enhancement block is used to get only the clean bits which forms the final cryptographic key. As mentioned before, the challenge to the XbarPUF circuit is generated from a TRNG before each back-up operation.

For reliability enhancement purposes, same challenge is applied to the PUF multiple times to detect which bits are prone to flipping and can cause bit-errors. For an XbarPUF, the main source of producing bit-flips is the case when memristors' cycle-to-cycle variation

overtakes the variation caused by their chip-to-chip or device-to-device variation. During device testing, severely unreliable bits cane be detected and eliminated totally while other bit-flips can be detected run-time. Probability of bit-flips also depend on a particular challenge and thus it is useful to find bit-flips run-time rather than making a large table of them beforehand to act as a helper data. Different bits of the challenge vector cause the PUF responses to depend on different memristors in the crossbar and thus, locations of bit-flips are highly depend on particular bit values of challenge vectors. The key generation process including TRNG, XbarPUF, and reliability enhancement operation are described in algorithm 1 [70].

After generating a secure and reliable cryptographic key, the data is then being encrypted and save in a persistent memory or NVM. In the simplest case, XORing is performed as the encryption operation while the key is changed every time to maintain perfect secrecy. A secure tag or hash is also generated from data integrity verification purposes using sneak-path based tag generation method [40]. This is also a hardware security module which helps to validate the integrity of data during restore. The tag is saved in the NVM as well. The sequence of operations when there is low power warning or when a power failure occurs are described in algorithm 2 [70].

The duration of power failure or energy-saving mode can be long and depend on particular device and its power/energy source. We have used memristors as NVM in our design and they can have long retention time without any power source [80], data can be retained for a sufficiently long duration of time for a particular IoT. When the power comes back onto the processor based system, the system first checks if there is enough power to resume the operation or not. If enough power is available, a new tag is first generated from the back-up data and compared with the stored tag. If they do not match, the data is said to be corrupted by either illegal modification, data-error, or key-error and the processor simply flushes all the data and restarts operation. If the tags match, then key generation protocol is activated again to generate the same key as before and decrypt the data into the processor memory. The sequence of operation for secure restore when there is available power in the system is described in algorithm 3 [70].

Result: Cryptographic key: **KEY**; status: *VALID_KEY*;
Initialization:
T ← no. of samples for reliability enhancement ;
m ← no. of parallel small PUF blocks ;
n ← no. of response bits from each small PUF block ;
resp ← **m**×**n** ; `// total response`
Function `Key_Generator():`
    Enable(PUFs) ;
    C = TRNG() ; `// random challenge`
    for $i \leftarrow 0$ **to** T **do**
        for $j \leftarrow 0$ **to** m **do**
            $R_{i,j}$ = apply_challenge_to_PUF(**C**) ;
            save_in_SRAM($R_{i,j}$) ;
        end
    end
    for $k \leftarrow 0$ **to** *resp* **do**
        if *bit_error* **then**
            discard_bit(**R**[k]) ;
        else
            KEY.append(**R**[k]) ;
        end
        if *length(KEY)==nKey* **then**
            VALID_KEY ← *TRUE* ;
            break ;
        end
    end
    if *length(KEY)<nKey* **then**
        VALID_KEY ← *FALSE* ;
    end
    return KEY
End_Function ;

Algorithm 1: Designed secure and error-free key generation method from PUF [70]

Data: States and other data to be backed-up, **DAT**
Result: Encrypted data, **enc_dat**, and **tag** in NVM
Secure_backup:
if *Low-power-warning is asseted* **then**
    Enable(HS_block) ;
    KEY = Key_Generator(**C**) ;
    enc_dat = Encrypt(**KEY**, **DAT**) ; // XOR(KEY,DAT)
    write_in_NVM(**enc_dat**) ;
    tag = gen_tag(**enc_dat**) ;
    save(**C**, **tag**) ;
    Disable(HW_block) ;
else
    Continue_regular_operation ;
end

Algorithm 2: Designed secure data back-up operation

Data: Encrypted data: **enc_dat**, and **tag**
Result: Restored data, **DAT** back in registers
Restore_data:
while $P_{available} < P_{threshold}$ **do**
    wait ;
end
Enable(HS_block) ;
enc_dat = read_from_NVM(stored_data) ;
C = read_from_NVM(stored_challenge) ;
old_tag = read_from_NVM(tag) ;
new_tag = gen_tag(**enc_dat**) ;
if *new_tag = old_tag & VALID_KEY = TRUE* **then**
    KEY = Key_Generator(**C**) ;
    dec_dat = Decrypt(**KEY**, **enc_dat**) ; // XOR(KEY,enc_dat)
    write_back_in_registers(**dec_dat**) ;
else
    flush_data() ;
    restart_processor() ;
end
Disable(HS_block) ;

Algorithm 3: Designed secure data recovery operation [70]

Figure 5.3 connects all of these three key generation, secure backup, and restore algorithms and shows the overall working mechanism of our designed security protocol. Key generation algorithm is a part of both secure and restore operations and produces a run-time unique random key from the XbarPUF. The secure back-up is activated when there is a low power warning and can be designed to be activated by an interrupt of the IoT processor. The secure restore is activated when the available power is greater than the minimum required power. This whole control flow graph is illustrated in Figure 5.3.

5.5 System Implementation

To get a very accurate idea about the actual real-world implementation, we have designed our whole security protocol at transistor level with 65nm technology on Cadence Virtuoso platform. We have used RRAM as the NVM and simple D flip-flop based registers as the processor memory which needs to be backed-up. We have calculated all our security metrics, overhead analysis from this low level implementation. In this particular prototype, our system generates uses a 32×32 XbarPUF to generate a reliable 16-bit cryptographic key. Figure 5.4 displays the system block diagram of this system and its different components are described next in this section.

5.5.1 One Time Pad Implementation

The one time pad (OTP) or also often called the Vernam cipher is the perfect cryptographic algorithm providing maximum security [24]. In this cryptosystem, the plaintext is paired with a random secret key and that key is changed every time a new encryption is needed, thus theoretically ensuring the best security. However, this one time pad or OTP is not used much in practice due to difficulty in its requirements. The key must be (1) truly random, (2) as long as the plaintext, (3) cannot be used more than once. The distribution and secure storage of the key are also big issues. However, here the requirements are fulfilled or relaxed. First, in embedded system or IoT domain, the plaintext or the data to be backed up is small and, therefore, the key can actually be made as large as the data. Second, in theory, PUFs can produce true random number as responses if it receives pseudo-random

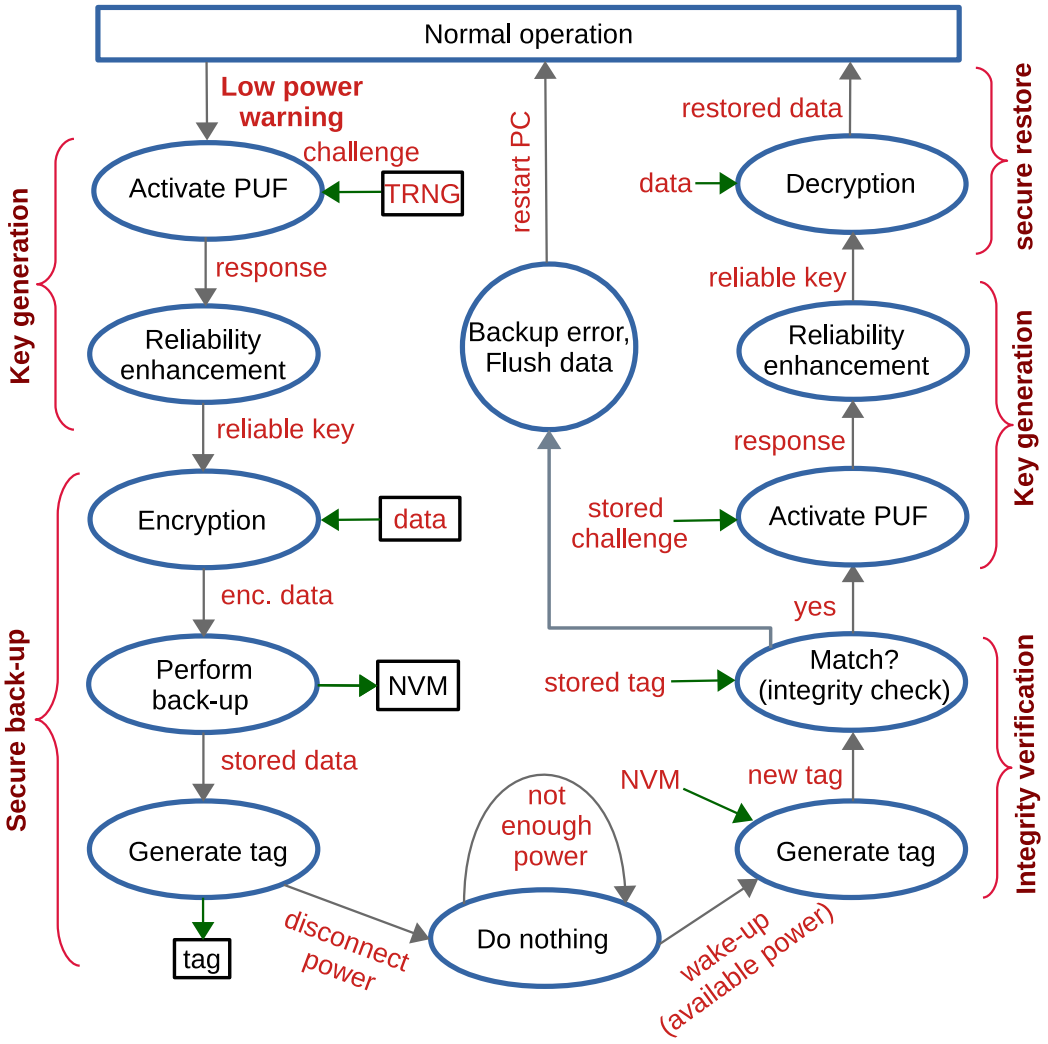


Figure 5.3: Control flow graph for our secure back-up and restore protocol [70]

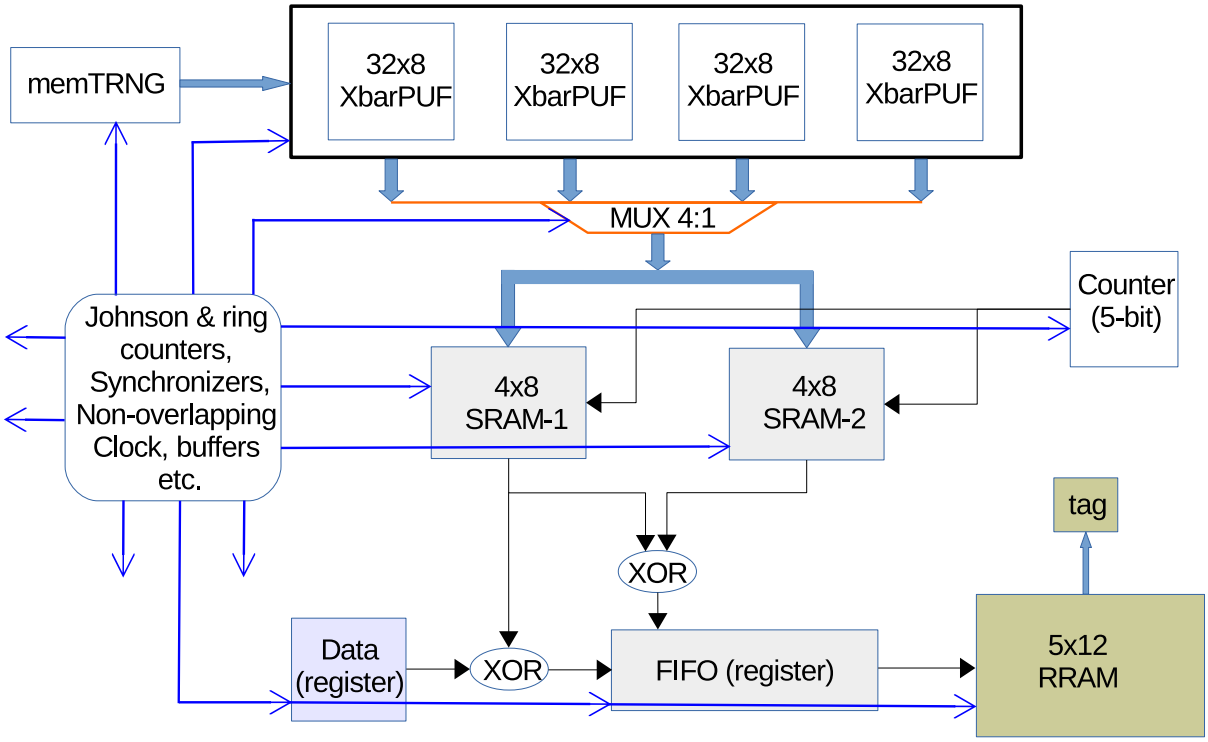


Figure 5.4: System block diagram showing different components of our designed security protocol [70]

number as challenges and thus no need to provide truly random number separately. Third, the key need not to be stored since PUF can generate the key anytime if the same challenge is applied. By ensuring different challenge input, the requirement of producing different key can be fulfilled.

5.5.2 True Random Number Generator (TRNG)

To generate a unique key during each back-up operation, we need to provide the PUF with a unique challenge. We can use a TRNG to produce that random challenge from the hardware itself. This would also help avoid repetition of generating the same challenge and same key in an unpredictable fashion, thereby fulfilling the requirement of an OTP system. TRNG output should be independent of any external influence like temperature, supply voltage noise and should be robust against process variation. Specifically, IoT device can go through extreme environmental changes and thus robustness against such changes is even more important in this domain. Researchers have successfully built TRNG from memristive devices [26]. In this work, we are using our custom-designed designed TRNG [67] which is expected to be very robust against environmental and process variations unlike existing designs. During each back-up, TRNG would produce a random challenge vector which is applied to generate the response from a PUF during both back-up and restore. Therefore, TRNG output is saved in the NVM during back-up to be used during data restore.

5.5.3 Redesigned XbarPUF for IoT

The memristor crossbar PUF or the XbarPUF is the heart of operation of this system. In a regular XbarPUF [69], challenges are applied for a very short period of time, much smaller than the actual switching time of a memristor, to nudge the memristance towards HRS or LRS depending on the magnitude and direction of applied challenge voltage. Then a small read voltage pulse is applied to read the memristance of both crossbar column memristors. This whole process is repeated until one of the columns reach to either HRS or LRS first and thus declared the winner. This is the so-called read-monitored-write approach [42] where knowledge of precise switching time is not important. The problem with this approach is

that the time to get a valid response is not predictable and thus not suitable to use within a processor based system.

To overcome these issues, we do not use the gradual read-monitored-write approach and use a single fixed pulse to nudge the memristors midway between HRS and LRS. Although, it requires prior knowledge of exact switching time of memristors, it makes the input-to-output sample time fixed and predictable and thus useful to incorporate in any processor based system. The memristor switching time can be determined beforehand from fab or we can change the clock period to try with different frequencies and select whichever gives the most stable output. We choose the clock period to be around half the switching time of the memristor. Choosing a fixed low frequency could make both the column memristors reach to either LRS or HRS and thus eventually having little difference between them. Choosing very high frequency could make the change in memristance very small and thus could be impractical to differentiate in a circuit. For the switching time on the range of 50-100ns, we have chosen the pulse width of the clock to be 25ns. This technique doesn't require the use of arbiters or flip-flops and sense amplifiers are used instead to generate responses. The design of the sense amplifier is presented in this next subsection.

5.5.4 RRAM for Non-Volatile Storage

Memristor based memory or RRAM is used as the NVM of our designed system. As we'll later in this chapter, we have implemented a prototype system with 16-bit data and thus we require minimum a 16-bit storage to store the encrypted data. To account for bit-errors, we have designed the XbarPUF to have a 32-bit response. However, to keep a generic design, and also that NVM would also store challenge, tag, and other necessary information, the NVM is sized 5×12. The LRS of the memristor is considered as logic '1' while the HRS is considered as logic '0'. Each cell of the RRAM is a 1T1R logic element where there are one large access transistor in series with a memristor. The HRS/LRS ratio and the dimension of the overall RRAM determine the size of the transistor to minimize area and power overhead while maximizing noise margin between two logic levels. More details i.e. address decoder, column decoder, counter etc. designs are skipped for brevity.

5.5.5 Time Multiplexing of PUF to Lower Power Consumption

The first phase of XbarPUF requires a complete reset of all the memristors of the XbarPUF. Since this involves states change of around half of all the memristors, this requires a lot of static power. For any small system, this would mean a much higher peak power demand during this time which might not be feasible. Therefore, we have decided to use time multiplexing to reduce the peak power demand and distribute energy over several cycles. But we can not distribute this indefinitely since that might increase the overall energy requirement considerably because some logic blocks would need to activated all the time and this also adds additional delay. To trade-off between peak power demand and delay, we have to choose an optimal PUF size with required no. of multiplexing unit. For our prototype design, we have decided to divide our 32×32 XbarPUF to four 32×8 XbarPUF which adds 3× additional delay but also reduces the peak power by a factor of almost 4. For system with different circuit configuration, size and different application requirements, further research is needed to find a optimum power-delay point.

5.5.6 SRAM as Part of Reliability Enhancement Block

As mentioned earlier, any PUF is prone to errors which limit their usage as a key generator. Thus there needs to be some form of error correction so that the key can be considered stable over one cycle of encryption and decryption. The current idea is to use a pair of SRAMs to hold two sets of the same response and then compare to find any errors. These two SRAMs are custom designed and to hold two sets of temporary 32-bit response of the XbarPUF, two 4×8 SRAMs (32 bits each) are used. The SRAMs can be written with 8 bits at a time. Each SRAM cell is a standard 6-T cell, with PMOS and NMOS transistors sized accordingly to ensure fast write as well as reliable read operation. Each pair of bit-lines (regular and complimentary line) contains one custom designed fast and high resolution 9-T sense amplifiers to detect bit-line voltage differences, 3-PMOS bit-line precharge and equalization circuit, and 2-inverter 2-NMOS input logic circuit. Address decoders, MUXes, and counters are also designed to be integrated with each SRAM as well for proper addressing and data input-output.

5.5.7 Sneak-Path based Integrity Checking

If an adversary tries to alter any data stored in the back-up non-volatile memory, then to detect that unauthorized change, a sneak-path based tag generation is also employed. The details of the sneak path of a memristor crossbar and how to generate tag from it is discussed in [41]. After each backup operation, a tag is made and saved in a secure non-volatile memory. During a wake-up, a new tag is generated from the encrypted data and compared with the stored tag. If the two tags are different from each other, then an error/ alteration of data is found and instead of using the back-up state information, the processor flushes all the data and restarts the whole process. This system would also detect if there is any mismatch due to the PUF's reliability issues since if the two response keys generated during back-up and wake-up are different, then processor would simply discard that data and restart everything. In this way, although the forward progression of processor is halted, any kind of these errors would not make their way into the processor data to result in an faulty calculation.

5.5.8 Control Circuit Design

We have also designed control circuits for our security protocol so that it can be "plugged-in" to any processor based system. This whole security protocol would be activated once where is a low power warning (e.g. for batteryless systems) or once the system decides to go in a low power mode i.e. sleep or hibernation. In both cases, data may be left on a unsecured memory and our goal is to secure this data while the processor is in a low power state and load data back into registers and other memory locations when it wakes up. We design and implement all required state machines at transistor level for the prototype system that we are building here. To accommodate various power/energy, area, and speed requirements of different devices, we can do simple overhead analysis to find the best sets of parameters for a particular application.

When a low power warning is issued, a random challenge is generated from a TRNG. A low power and robust TRNG design using memristive technology is discussed in a previous chapter. This challenge can also be generated beforehand to reduce the time and energy it

would take during this low power mode. This challenge is then applied to an XbarPUF block which would generate the responses required to create the secret key. The XbarPUF requires a RESET phase where all memristors are reset from their LRS to HRS. Thus during this phase, static current flows through the whole crossbar, consuming a large amount of power. This even though the overall system may have a low average power, this phase would demand a large peak power demand, making it impractical for many systems. That's why we are diving the whole crossbar into multiple smaller crossbars to reduce this peak power demand during RESET phase. The number of smaller crossbar PUFs and their individual sizes can be determined based on a particular memristive technology along with circuit level parameters and design requirements. For this work, we envision memristors to have very high LRS and HRS values along with low threshold voltages to reduce the amount of energy required to switch their states. For this reason, we have decided to use the TiO_x memristor shown in [48] as the example memristor to build this system. The parameters for this memristor is given in 4.4 and again shown in here in Table 5.1.

To reduce this peak power demand, we have divided our 32×32 XbarPUF into four 32×8 XbarPUF, thus effectively reducing the static power by a factor of 4. However, this increases the delay in producing responses from each of these four separate PUFs with some added area because of their peripheral circuitry. Each PUF is activated one by one, as they each generates one portion of the final response vector. These partial responses are written into a temporary memory as soon as they are available. For this particular design, we have designed an SRAM to act as the temporary storage. During each cycle, one XbarPUF is active while the other remains disabled, and one row of the SRAM is written by this XbarPUF. Here, our designed SRAM is sized 4×8 so that each row can hold 8-bit response that each smaller (32×8) XbarPUF produces. This makes their control circuit neat and easier to decode. The SRAM is custom designed too, with the same 65nm technology like the rest of this security architecture. Now, we have implemented an all-agree voting or majority voting (discussed in a later section) as reliability enhancement technique here where a response bit is taken as a valid key bit only if that bit doesn't change among a certain number of measurements. More measurements increase the accuracy of the reliability enhancement technique but it also adds delay and energy overhead. As for this system, we are implementing the simplest

Table 5.1: Switching Parameters (mean value) for TiO_x Memristors [48]

HRS	**LRS**	V_{tp}	V_{tn}	t_{swp}	t_{swn}
$2M\ \Omega$	$500k\Omega$	0.5V	0.5V	10ns	10ns

2-vote all-agree voting scheme. To implement this, PUF responses are generated twice and saved in two different SRAMs. After comparing the contents of these two SRAMs and removing erroneous bits, each key bit is XORed with data bit and saved in an RRAM bit by bit. This phase is long in terms of delay but the average power required is very small compared to PUF response generation steps. We can reduce the time required by storing multiple bits at once (e.g. byte access or word access). It could also increase the required number of response bits for a particular size of key, however.

We have used standard Johnson counters and ring counters to implement state machines in our circuit design. In the first state after security blocks are enabled, a complete response vector is generated from all four 32×8 XbarPUFs. Thus, there are four sub-states under this first state too. A single 32×8 PUF is activated during sub-state. This involves resetting all the memristors, applying a random challenge to the PUF, and reading the response. Clock pulses with the same period but smaller duty cycles with different phases are used to control these operations. Moreover, during each sub-state, response from each PUF is saved in one row the first SRAM. Thus at the end of first state, this SRAM-1 contains 32 response bits from four XbarPUFs. The second state is very similar to the first state, when responses are generated from these four PUFs and saved in an SRAM. Little difference is that now the responses are saved in a different SRAM. In the third state, contents from these two SRAMs i.e. saved responses are compared against each other by XORing and only valid or unchanged response bits are considered as a part of the final key. Each key bit is XORed i.e. encrypted with one bit from the data and written in an RRAM. Thus depending on the number of key/data bits, this state can be long. In our designed state diagram, this state is actually much longer than the first two states to account for the errors in in the response vector. Each erroneous bit means one clock cycle lost as it is discarded. For this system, we need a minimum of 16 cycles to read 16 correct response bits or keys. To account for the maximum number of errors, we make this state 32 bits long as the whole response is 32 bit

itself. State 1 and state 2 are 4 cycles each, where each clock cycle is used to generate one set of response from each individual PUF. The clock period, pulse width, supply voltages etc. are chosen to minimize the power consumption while ensuring a good window of operation for the particular types of memristors that we have chosen. Specifically, we have chosen a VDD of 0.85V with a clock period of 1μs for this simulated design. The analog MUXes, buffers, and other components are sized accordingly to minimize power consumption and voltage drop across them.

At the last step before the power is turned off during a power failure or low power mode, a tag is generated from the stored memory content. This tag along with the PUF challenge is saved in a secure small memory. When power comes back on or when the processor starts to get out of low power mode its wake-up protocol takes place. First, a new tag is generated from the stored (and encrypted) data. Then the old stored tag is read and compared with the new tag. If they do not match, then the data is considered to be corrupted or altered illegally and the processor restarts its operation, flushing all the data. This way any illegal modification of data or any key or data error won't propagate their way into the processor. If both tags match, then key generation and reliability enhancement processes take place. Just like before, the PUFs are activated one by one to generate and store a 32-bit response into the SRAMs. Then erroneous key bits are discarded to get a clean 16-bit key. Finally, encrypted data from the RRAM is read bit by bit and XORed with the corresponding key bit for decryption and then saved back into registers again. Except for the tag generation and matching part, all the other operations are done at the same sequence during wake-up as in during back-up.

It is very important that no states are active at the same time i.e. there are no glitches at all, especially for time-critical time-multiplexing of PUFs to generate responses. To ensure that no two consecutive states have glitches, two phase non-overlapping clock generators are used. They are used at the outputs of state decoders between two successive states to prevent any metastability. Moreover, due to the presence of different clocks (e.g, SRAM clock, RRAM clock, PUF RESET and READ signals), synchronizers are needed to ensure reliable capture between two different clock domain. As the clock frequency is known and

all these clocks share the same frequency with just different phases, a simple two-phase synchronizer based on two flip-flops are used.

5.6 Initial Design without Reliability Correction

Initially we have developed a small prototype without the error correction or tag generation method, only to demonstrate PUF based key generation and encryption-decryption technique [74]. Figure 5.5 shows the conceptual block diagram of this small system. The secret key generation block of this figure is essentially a PUF. Before each power down or sleep phase, a random challenge is applied to the PUF to generate a unique random response. Then the data to be backed up can be encrypted with any very lightweight cryptographic technique and saved in a non-volatile memory (NVM). This way the data would be safe against malicious read. For this demonstration, we have used a simple bitwise XOR to encrypt the data. To restore the data, the same challenge is applied to the PUF again to generate the key. In this case, simply XORing the encrypted data with the PUF key would give back the original data. The idea is to provide robust security like an one time pad where they key is random and changed on each encryption-decryption operation. The added benefit of using a PUF is that the key is inherently random with pseudo-random challenge input, need not to be stored physically and can be generated on demand.

5.6.1 Overview

For this demonstration, we have implemented a 4×4 modified XbarPUF which takes 4 challenge bits as input and produces a 4-bit response as the cryptographic key. This is shown in Figure 5.6 In the first phase of operation, this PUF takes one cycle to generate the 4-bit key and that key is saved in a register. In second phase, key is XORed with the data register and this encrypted data is written in an RRAM one bit at a time. Thus this phase takes 4 clock cycles for a 4-bit key. During decryption, the PUF is provided with the same challenge to produce the same key and encrypted data is read from the RRAM bit by bit. Decryption is done by XORing the encrypted data with the register again and the decrypted data is saved back in the original data register. Here, the encryption and

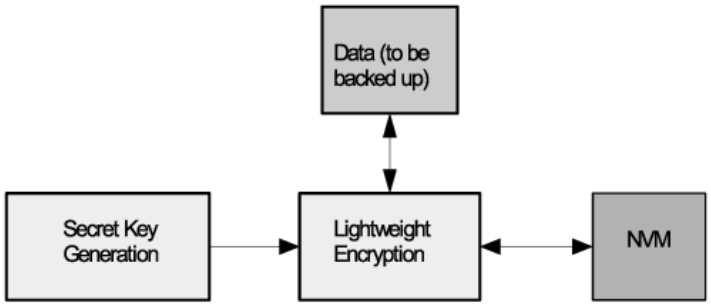


Figure 5.5: High-level conceptual block diagram of the initial security system [74]

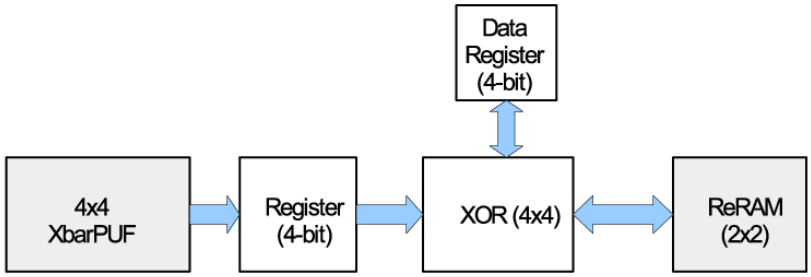


Figure 5.6: Block diagram of the initially designed prototype system [74]

decryption operation are symmetric and share the same XOR block to reduce area. The idea of this security design is to try to implement as close as possible to a hardware one time pad [24]. If the PUF is provided with pseudo-random challenges as input, the responses would be random. Each time a power-down occurs and backup operation is needed, the PUF is provided with a new challenge and thus the resulted response or the key would be new too. Therefore, just like an one time pad, the cryptographic key would be changed each time it is used. Moreover, since the length of data to be backed up is not large in these IoT devices, the key can actually be made as large as the data and thus any repetition based attack would not reveal the key.

5.6.2 Results for Initial Prototype

First, we have evaluated our PUF by running Monte Carlo statistical analysis. The results for important PUF metrics are listed in Table 5.2. Although, it is only a demonstration,

Table 5.2: Performance of modified XbarPUF used in this design [74]

Security metrics	**Uniqueness(%)**	**Bit-aliasing(%)**	**Reliability(%)**
Average	50.090	50.833	99.904

using Monte Carlo simulation, we have analyzed 500 chip instances, and each for 500 cycles. As we can see, our XbarPUF shows excellent uniqueness and bit-aliasing which indicates the random nature of the response bits. Reliability is also very good, although not 100% as a typical cryptographic algorithm demands. However, in this system, the PUF only needs to be reliable in a single set of encryption and decryption cycles and causes no problem if the PUF produces different response in a completely different time and can actually increase the entropy of the PUF this way.

Table 5.3 shows the overhead of this initial design. The encryption and decryption phases involve a lot of memristor read-write operations and consume larger amount of power compared to write-only and read-only stages for the ReRAM. Also, we have used HfO_2 memristors for this design which have relative lower HRS and LRS values. Later we have switched to TiO_2 memristors with much higher HRS and LRS values to reduce the power consumption significantly. Each of the key generation phase has just a single cycle delay. Where for an N-bit key and thus N-bit memory, it takes N cycles each to write and read all the bits.

The total GE (gate equivalent) count compared to a unit-sized NAND of our whole system is approximately 547. Majority of this number comes from the large pass-gates used for read-write control circuits of RRAM. Fortunately, these control blocks can be shared among many row-columns and thus GE count won't increase much with increasing RRAM sizes. There are also a total of 64 memristors in the XbarPUF (4×4) and 4 memristors in the ReRAM(2×2). Memristors in crossbar architecture takes very little area compared to even a single transistor and thus they contribute to a large reduction in overall area.

Table 5.3: Overhead in different phases of the system [74]

Overhead	**Enc. key generation**	**Memory write**	**Memory read**	**Dec. key generation**
Power (μW)	24.61	0.79	0.44	24.96
Delay (clock cycles)	1	4	4	1

5.7 Reliability Enhancement Technique

After designing and demonstrating the low overhead with security properties of our design, we have worked on improving the reliability of the PUF for practical key generation applications. Since a PUF's response depends on tiniest manufacturing process variation which can be affected by noise or other environmental effects, and we have already seen that our XbarPUF doesn't have ideal 100% reliability. As traditional error-correcting methods can huge area, delay, and power overhead, we have considered two fast, low overhead and run-time reliability enhancement techniques in our work. First one depends on isolating and eliminating bit-flips from the PUF response to determine the final key and we call it all-agree or veto-voting technique. The second approach is well-known majority voting technique. Both of these are described below.

5.7.1 All-Agree Voting/Veto Technique

All-agree voting or veto technique depends on finding and eliminating bits which flips between multiple responses of a PUF even for the same applied challenge. To implement this, the same challenge is applied for an 'N' number of times to produce 'N' sets of responses. Ideally, without any bit-error, all of these should be the same. However, in practice, there would be some unstable bits which would flip among these evaluations. In this all-agree voting technique, we only consider bits that do not flip even once in 'N' evaluations to be a part of the final cryptographic key. This technique, therefore, requires the no. of bits in response to be longer than the key to accommodate for bit-flips. This number of additional bits and the optimal number of evaluations ('N') are determined experimentally.

In this all-agree voting technique, a bit would be only be considered as part of the key if that bit remains stable during both encryption and decryption, resulting in a different key.

Suppose, the probability of a particular bit being stable at a particular logic value (either at '1' or '0') is p. Thus the probability of that bit being the opposite value (not stable) is 1-p. Thus for an 'N' of evaluations, the probability of a bit being stable is p^N while the probability of bit flips at least once is (1-p^N). Now this bit-flip would propagate to cause an error without being detected by this voting technique is given by equation 5.1. PE denoted probability of error.

$$PE_1 = p^N * (1 - p^N) \tag{5.1}$$

There is another less-likely way of an error being propagated into the system with detection. This is when a bit remains stable in its less-stable state during encryption but flips during decryption. The probability of that happening is given by equation 5.2.

$$PE_2 = (1 - p)^N * (1 - (1 - p)^N) \tag{5.2}$$

Now in both cases (equation 5.1, 5.2), we consider that a particular bit remains stable during encryption but flips during decryption. These same two equations hold for a case where a bit can flip during encryption but remains unchanged during decryption. Thus the probability of an bit-flip being propagated, and resulting in a part of the cryptographic key is given by the following equation 5.3:

$$PE_{1,2} = 2 * [(p^N * (1 - p^N)) + ((1 - p)^N * (1 - (1 - p)^N))] \tag{5.3}$$

However, we need slight modification as this equation counts two special cases twice. This situation happens when the PUF produces all '1's during encryption, but all '0's during decryption for a particular bit and vice-versa. The probability of such a case is:

$$PE(allzeros + allones) = 2 * [(p^N * (1 - p)^N))] \tag{5.4}$$

Both equations 5.1 and 5.2 take this situation into account and thus this is being considered twice while finding out the overall error propagation probability of the all-agree voting technique as given by equation 5.3. Thus by subtracting this case from equation 5.3,

we get the final expression for a bit-flip being undetected and propagated into the final key with all-agree voting system and is shown shown here.

$$PE(\textit{all-agree voting}) = 2 * [(p^N * (1 - p^N)) + ((1 - p)^N * (1 - (1 - p)^N)) - (p^N * (1 - p)^N))] \tag{5.5}$$

For a few different suitable numbers of evaluations, 'N' and for different probability values of 'p' (denoted the probability of a particular bit being stable at a fixed binary logic i.e. represents bit-flipping probability), we have used equation 5.5 to show a bit-error probability plot in Figure 5.7.

In any PUF, there could be some bit positions in the response where there is a high bit-flipping probability while most of the other bits in that response remain stable at a particular logic value for the same challenge. In our XbarPUF, as we have shown later from exhaustive simulations that on average there are less than only 2 bit-flips out of the 32-bit PUF response over a long period of time with the same challenge being applied. Thus if it is possible to eliminate these few bits having high bit-flip probabilities, then the system would have overall a smaller probability of generating different keys between a pair of encryption and decryption cycles.

From Figure 5.7, it is evident that with larger number of samples or evaluations (N), all-agree voting scheme is more efficient at eliminating highly flipping bits. Its implementation is also simpler with just XORing to find detect bit-flips and two separate storage to contain both actual response and bit-flip information.

5.7.2 Majority Voting

Majority voting scheme is a very well-known and established error correction technique. To implement this technique for PUF error correction, the same challenge would be applied to a PUF for a fixed number of times and a bit would be considered either a '1' if it produces '1' more than '0' among those evaluations or '0' where it produces more '0'. Thus unlike all-agree voting technique, no response bits are not discarded here, rather majority voting is used to

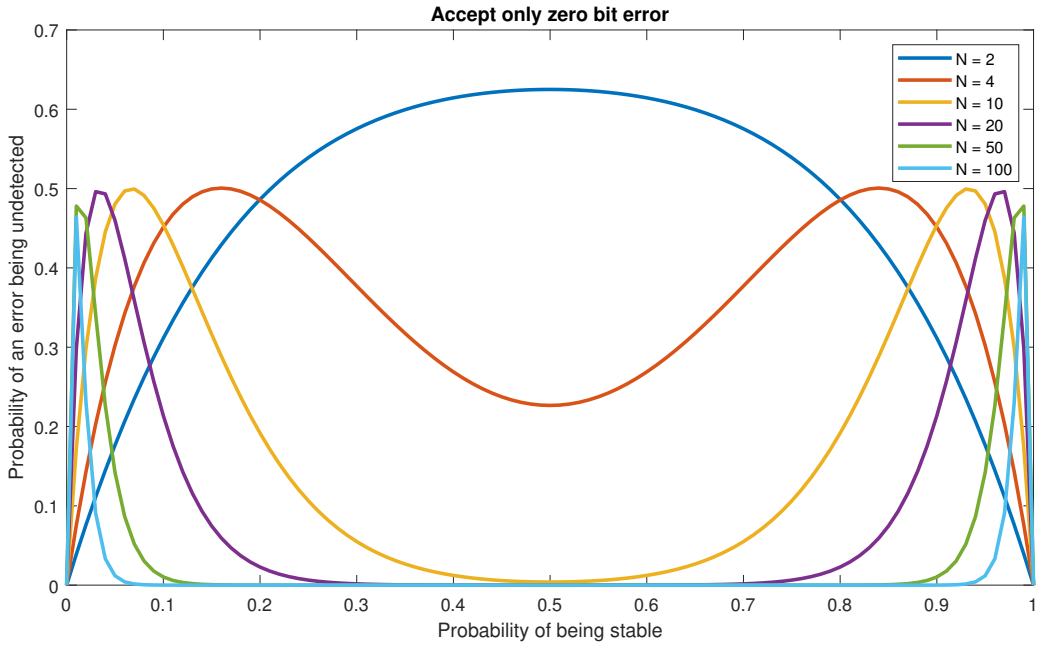


Figure 5.7: 'All-agree' reliability enhancement technique for different number of evaluations for different bit-flip probability. It is more efficient at detecting highly unstable bits with increasing number of evaluations [70]

determine their more stable or prominent binary value. The probability of determining the correct binary value of a response bit using this technique with 'N' number of evaluations are given by the following equation:

$$\begin{aligned} P(majority\ voting) &= \sum_{r=\frac{N}{2}+1}^{N} \binom{N}{r} p^r (1-p)^{N-r} \\ \implies P.E. &= 1 - \sum_{r=0}^{\frac{N}{2}} \binom{N}{r} (1-p)^r p^{N-r} \end{aligned} \tag{5.6}$$

Derivation of this equation is provided in Appendix E. Figure 5.8 shows a plot using this equation for different number of samples ('N') and for different bit-flip probability ('p'). With larger number of 'N', the curve gets narrower around the center i.e. it can detect bit-flips more effectively. This technique, however, is not effective for bits which already have

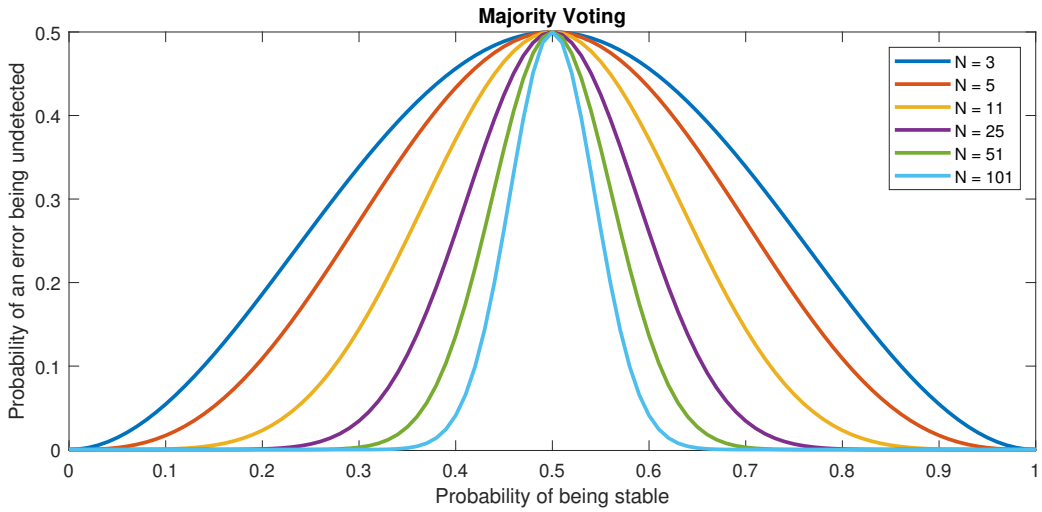


Figure 5.8: Effectiveness of majority voting for reliability enhancement technique for different number of evaluations and with different bit-flip probability [70]

very high flipping probability, p = 0.4-0.6 as it is evident from the figure since the peak of the curve remains the same for any number of evaluations.

One of the other things for consideration of majority voting technique is the overhead. The overhead could be high for larger number of evaluations, 'N'. For example, for an N-sample majority voting, we would need counters that can count up to N for each response bit, and memory capacity of $log_2 N$ bits to store the results for each bit of the PUF response.

5.7.3 Chosen Reliability Enhancement Technique

If we look carefully at both Figure 5.7 and 5.8, we see that all-agree voting is more effective at detecting (and eliminating) high bit-flips while majority voting is better at detecting bits with smaller bit-flipping probability. Delay and area overhead increase with increasing number of evaluations for majority voting where all-agree voting can be implemented simply with using XORs and one extra bit per response bit for any number of evaluations. To reap the benefit of both of these techniques, one idea is to use all-agree voting with large value of 'N' during chip functionality testing to eliminate bits with high flipping probability (0.4-0.6) and only use majority voting with small 'N' during run-time to reduce bit-error. If all-agree

voting can eliminate highly unreliable bits before regular operation, then majority can be used with only a few samples during run-time, thereby also ensuring lightweight operation.

The choice of reliability enhancement technique heavily relies on the bit-flipping profile of all the response bits of a PUF. Other factors include the overhead associated with the implementation i.e. allowable power, area, delay constraints and security requirements of a system. Depending on all of these, we can choose to implement a particular reliability enhancement technique and not the other. One key thing to remember is that we are not implying to eliminate all bit-errors altogether, but to reduce its probability to such extent that the cost or associated overhead when there is a key error (i.e. when the processor restarts) would be insignificant compared to the total savings in overhead associated with a particular reliability enhancement technique.

5.8 Working Principle

We have designed the whole system at transistor level in Cadence Virtuoso with $65nm$ CMOS technology and memristive technology. Figure 5.9 shows the waveform for different signals of this design. The X axis shows time in microseconds (μs). As mentioned before, each clock cycle is 2 μs long. Asserting signal 'EN' starts the sequence of back-up operation maintaining our security protocol. When the control signal 'state1' of Figure 5.9 is high, XbarPUF block and SRAM-1 are enabled. Signals 'en0' to 'en3' indicates activation of four different smaller XbarPUFs one by one. Thus in this state, each of the four XbarPUFs are reset, challenge is being applied, and read with the help of signals 'CLK', 'Reset', 'Challenge', 'WE' (write enable). During 'state2', almost exactly the same thing happens as these XbarPUFs are activated once more to generate responses. However, unlike 'state1', the responses are now saved in the second SRAM. Signals 'en0' to 'en3' also enable different rows of these two SRAMs to help writing PUF responses there. The next state is large and we call it 'state3to10'. During this state, contents from the two SRAMs are read and compared to find valid key bits, data is encrypted and then written into the RRAM. Signals 'dout-a' and 'dout-b' denote the output from SRAM1 and SRAM2, respectively. Without any error, they should display exactly the same values. However, for the purpose of demonstration, we have

intentionally produced 8 bit errors as shown in Figure 5.9. The signal 'MATCH' also show when there is a mismatch among these two SRAM output i.e. among key bits. By counting the occurrence of such events during this state ('state3to10') from Figure 5.9, we can also verify that the total number of bit error is indeed 8. Now as mentioned before, this demo design would use a 16-bit key. Thus we only need to extract first 16 'good' bits from the 32 bit PUF response. Since there are 8 erroneous bits, added intentionally, the system would read from SRAMs 16+8 = 24 times to get 16 'good' bits. Signals 'count0' though 'count4' represent the number of bits read from SRAM. From these signals, if we notice in Figure 5.9, we can see the count is '11000' or 24 at the end of this state which validates our claim. The signal 'stop-count' also shows when the counter is being paused i.e. when there is a bit-error. The signal 'key15' shows the key bit used for encryption in different clock cycle. The bottom three signals of Figure 5.9 are used for controlling the RRAM write. The signal 'valid-writ' is high only when there is no bit-error and if it is high, either 'reset-on-0' or 'set-on-1' signal is activated to write either a '0' or a '1' in the RRAM, respectively. Back-up data is represented using a 16-bit register which is XORed with the key and then written in the RRAM. Tag generation is done as soon as the system finishes writing 16 bits to the RRAM. During decryption, the XbarPUF blocks and SRAMs are activated again to generate the same valid key as before and RRAM is read bit by bit this time to XOR with the key and then write back in data and state registers.

5.9 Possible Attack Scenarios

5.9.1 Malicious read

The main motivation behind this designed security protocol is to prevent an attacker from reading out the contents from the backup NVM, thereby gaining sensitive information about the overall system. Our PUF-key based one time pad (OTP) encryption scheme encrypts the data before doing backup to prevent direct readout of sensitive information. As we have explained before, we are effectively implementing an OTP here. OTP is theoretically the most secure encryption system if we can fulfill its three main requirements: (1) random key,

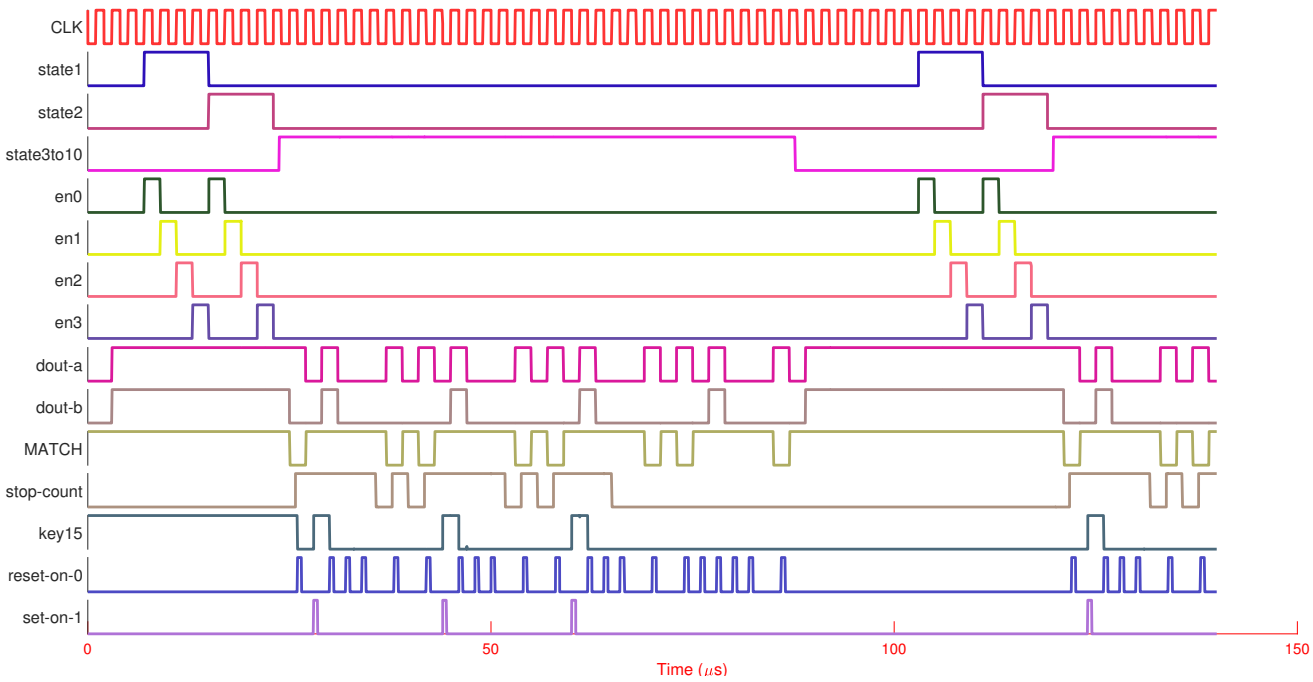


Figure 5.9: Waveform for a complete secure backup and restore cycle showing how different states control different operations of the system. Counter keeps track of the number of valid or legal key bits and stops each time a bit error occurs

(2) unique key during each operation, and (3) key being as large as the data. Our generated key is random as it comes from a strong PUF and in the expected embedded or IoT system where the amount of backup needed is small, the key can actually be made as large as the data. Moreover, since the backup operation could be infrequent and the overall state-space of a strong PUF could be very large, a PUF key is unlikely to be repeated predictably in a practical time-frame of an IoT device and thus replay attack should be improbable. Key sharing is another weakness of an OTP, however, this is not a concern here as the key is only used in-place within a same device. Therefore, we are fulfilling all the requirements of an OTP and ensuring maximum security with a unique random key.

5.9.2 Malicious Write

An adversary might try to illegally modify the contents of the backup NVM arbitrarily, thereby creating an erroneous calculation in the processor. An error in original data or key itself can also result in an erroneous backup, especially during power failure or sleep and time

sensitive back-up operation. A memory integrity checking based on the sneak-path currents of an RRAM crossbar should be able to detect any kind of data alteration. During wake-up, this system calculates a tag from the backup memory and verify against its saved tag. If they do not match, the processor simply rejects all backup data and restarts its operation. This would result in an increase in overhead as the processor can't use previous states and backup data, but this makes the overall system robust against any errors or harmful injection of information.

5.9.3 Readout or Alteration of the PUF Challenge or Secure Tag

Since we need to produce the same response during encryption, the random challenge to generate the PUF response during back-up is also saved in the NVM, to be used during restore operation. The tag generated for integrity verification purpose is also save in a secure NVM. Here, we assume that attacker would have more difficulty to access these small bits of secure memory. If the PUF is robust against machine learning based modeling attack for XbarPUF [71], just getting access to the challenge shouldn't compromise the security. Moreover, continuous access to the PUF is not permitted in this system as the PUF is only used at some certain stages. Finally, illegally modifying the saved PUF challenge would almost definitely change the PUF response as well as the encryption key, which would inevitably result in an erroneously restored data. Pre-computing a tag or hash with the encrypted data during backup and restore would help to prevent such scenarios and detect such illegal modifications.

The tag is calculated from the sneak-path currents of the memristor crossbar [40], i.e. this is similar to an analog in-memory computation. Because of the analog nature of memristor's switching itself and its die-to-die and cycle-to-cycle variations, it should be very difficult, if not impossible, to repeat exact resistive states of a crossbar of memristors to regenerate the same sneak-path current and the same tag. Thus once the encrypted data and tag are written into NVM, any attempt of further modification should corrupt the data and the tag and thus processor would be able to detect it by doing an integrity verification during restore.

5.9.4 Modeling Attacks

Machine learning based modeling attack is one of the primary concerns of a strong PUF, especially PUFs which can be modeling using linear additive linear delay model as shown in [62]. The idea is to gather a subset of CRPs of a strong PUF and build a model using machine learning (classification) algorithms to predict responses for unknown challenges. This effectively reduces the randomness of PUF responses. We have shown in Chapter 4 that XbarPUFs are also vulnerable to modeling attacks while mitigation techniques are also developed. However, it is not possible to apply continuous challenges to the PUF in this system, unlike authentication applications. In our system, the only way to build a database for modeling attack is to observe many cycles of backup and restore and perform illegal read of PUF challenges and responses, thereby increasing the timeframe of the attack considerably.

In an extreme case, what an adversary can do is to gain control of a device, causing continuous back-up and restore with known data value so that it can gather information about the responses by doing malicious read of the encrypted back-up data and challenge. To prevent this from happening altogether, our XbarPUF should be able to resist modeling attacks. A simple and lightweight modification which was introduced in [71] is to add response bit XORing in combination with column shuffling/swapping technique which can drastically reduce the accuracy of modeling the XbarPUF, thereby increasing the robustness against such attacks. These techniques are analyzed in details in [71]. Moreover, in any sensitive IoT device could also have a tamper detection mechanism to keep an eye on the expected number of back-ups in a given time-frame and make sure that number doesn't exceed a certain value. This would considerably increase the time it takes an attacker to build a database.

5.10 Results: PUF Security Analysis

As we already mentioned, we have implemented a 32×32 XbarPUF as the key generator for our designed security architecture. Since PUF is the key component and basically the heart of operation of this security architecture, we have evaluated XbarPUF first in terms of these security metrics listed below.

- Uniqueness
- Bit-Aliasing
- Reliability
- Uniformity
- Diffuseness
- Steadiness

All of these metrics are defined and explained elaborately in [39, 23]. The formal definition, the characterizing equations are also presented in Chapter 2. Uniqueness measures the degree of variation among responses produced by different chips for the same challenge. Bit-aliasing determines how random the distribution of 1's and 0's for each bit position of a response of a PUF for different chips. Thus these two metrics define the randomness of PUF response across different devices. We have performed Monte Carlo simulation for 500 different chips, each with 25 different challenges to evaluate uniqueness and bit-aliasing. Figure 5.10 shows the uniqueness results for 25 different challenges. As evident from this figures, our designed XbarPUF displays almost ideal (50%) uniqueness for all these different challenges which strengthens the claim of this PUF based security architecture as an unclonable hardware security module. Bit-aliasing results for each of the 32 bits of the response vector of the XbarPUF for different challenges are shown in details in Figure 5.11. For each of the 32 bits, the bit-aliasing value is within the range of [0.45 0.55] and the average bit-aliasing value is 0.5 for all these bits, very close to the ideal.

Uniformity and diffuseness metrics measure a PUF's performance across the challenge space. For a single chip, if different challenges are applied, the responses should be different from each other so that ideally these metrics becomes equal to 0.5. Steadiness is another metric that evaluates the stability of a PUF's response over multiple challenges. It represents bias of individual response bits of a PUF on average. Specifically, it measures the degree of bias of a response bit towards either a '1' or '0' over many cycles as defined in [39]. We have run Monte Carlo simulations for 500 different random challenges, and for 25 different chips to evaluate uniformity and bit-aliasing. Steadiness is evaluated for 25 different chips for 500 cycles each. Uniformity, diffuseness, and steadiness results are shown together in Figure 5.12. Although for different chips these numbers deviate from this ideal value, they do not show very large deviation from their average values and display an average uniformity and diffuseness of near 0.5 as can be seen from this figure. All of these metric prove the applicability of this XbarPUF as a strong PUF, capable of generating unique and random

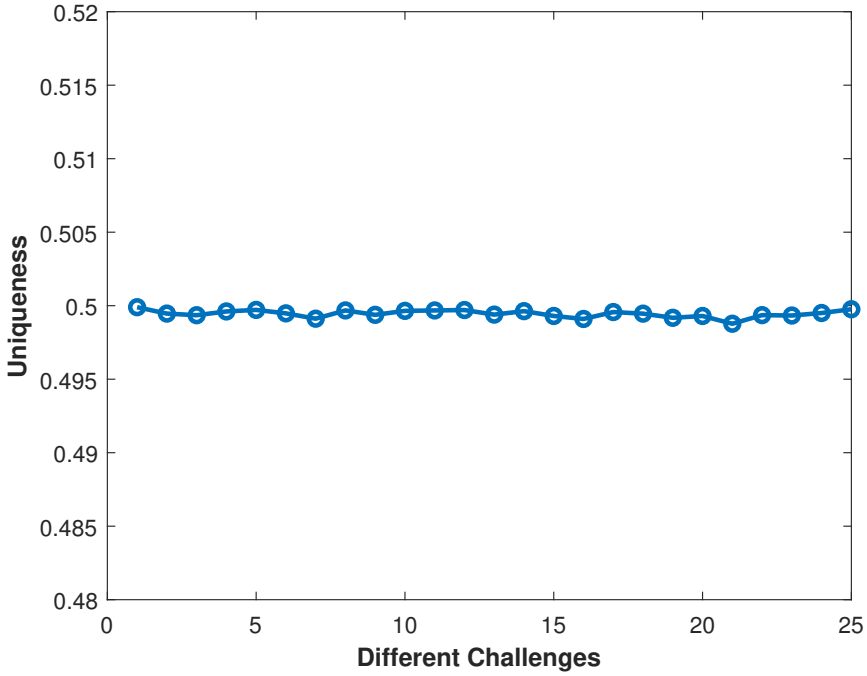


Figure 5.10: This figure shows average uniqueness results for 500 different chips [70]. Even for different challenges, this value is very close to ideal value of 0.5.

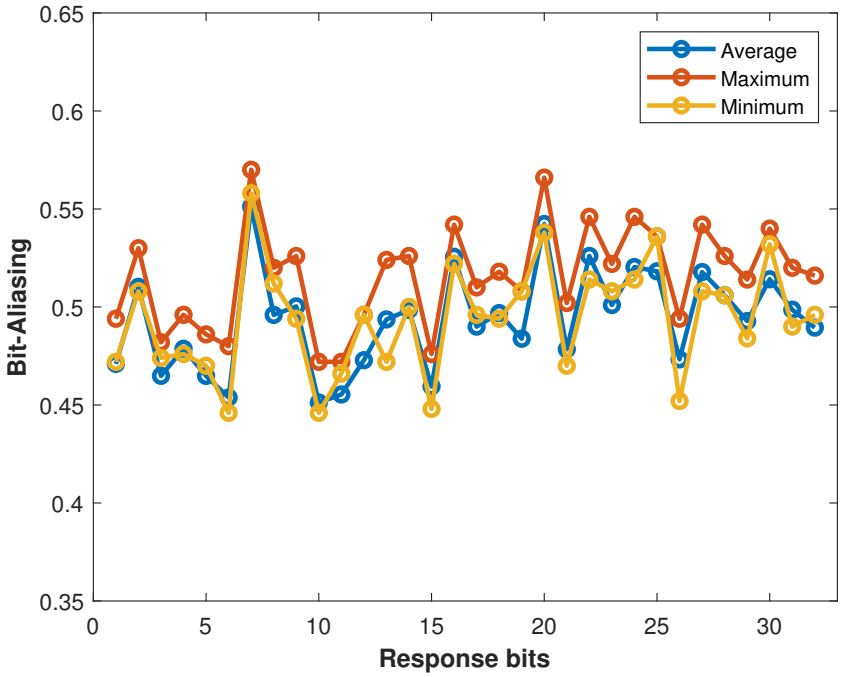


Figure 5.11: Summary of results for bit-aliasing for all 32 bits from 500 different chips. The minimum and maximum value for each bit are also shown [70].

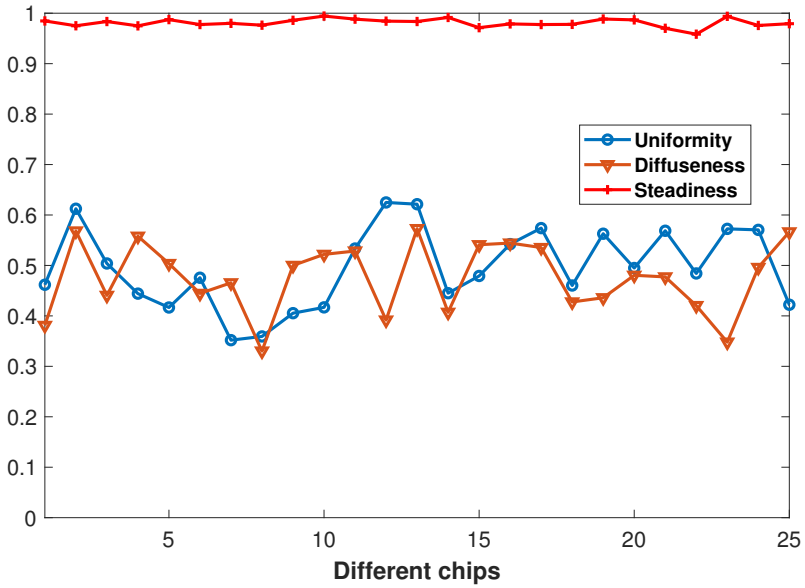


Figure 5.12: Summary of results for average uniformity, diffuseness and steadiness from 500 different challenges and for 25 different chips [70].

keys for different challenges and different chips. Also from the figure 5.12, it can be seen that steadiness value is very close to the ideal value of 1.

Reliability is one of the major concerns around PUF to justify their applicability in practical designs. As PUF depends on tiniest process variation to produce device-specific signature, any small change in environment or the presence of noise can make a PUF's response prone to change undesirably. Reliability, somewhat similar to stability (distinction is explained in [39]) is used to describe how reliable or stable a PUF's response is when the same challenge is applied again and again [39]. Because, reliability could be the most crucial metric to evaluate a PUF's practicability, we have performed a detailed reliability analysis using Monte Carlo simulations for 500 different clock cycles each, for 25 different challenges, and each for 25 different chips. This result is shown in Figure 5.13. This 3-D plot shows that for all different challenges and for all chips, the XbarPUF shows a minimum of 92% and on average 98% reliability. However, it is to be noted that this result is obtained from the regular XbarPUF directly, before applying any error correction technique. Using our chosen reliability enhancement technique, the idea is to eliminate unreliable bits from the response and only take the error-free bits to form a cryptographic key.

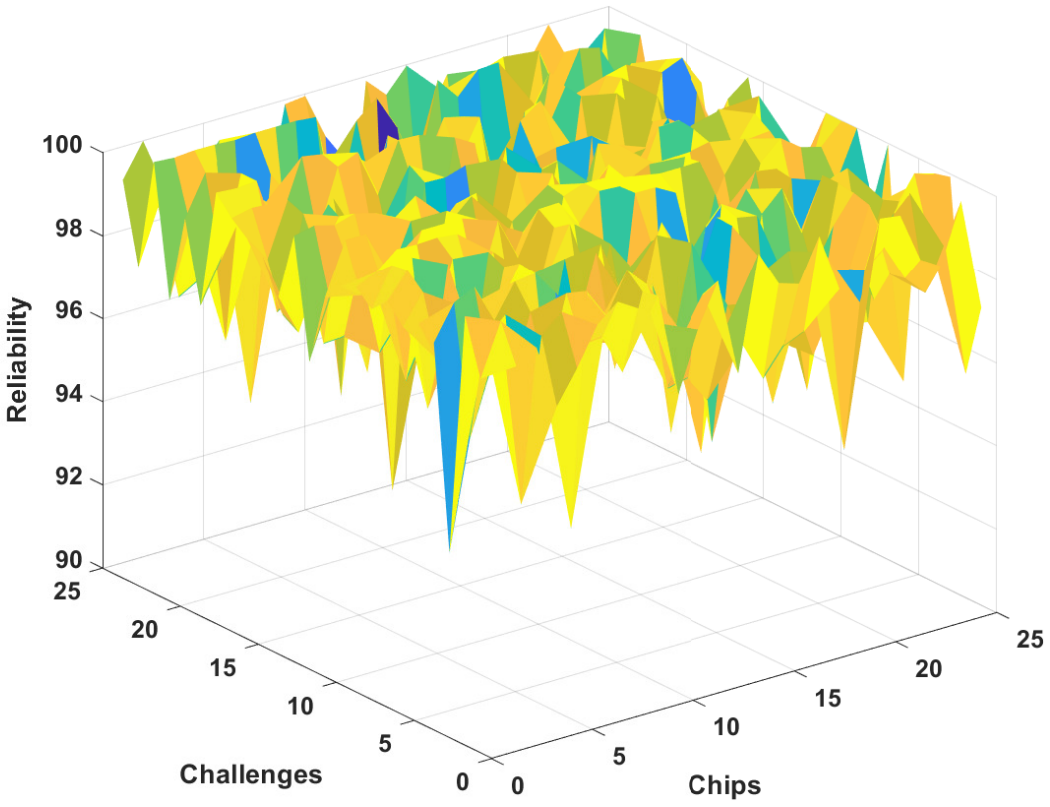


Figure 5.13: Detailed reliability results generated from 500 different cycles for 25 different chips and 25 different challenges [70]

After analyzing the security of the XbarPUF based key generation, we are now interested in evaluating the security of our sneak-path based tag generation method. This sneak-path based tag generation method is specific to memristive memory or RRAM and corresponding details can be found in [40]. Three metrics for the evaluation of any tag generation or hashing method are: uniformity (different than PUF's uniformity), diffusion, and avalanche effect. Uniformity dictates that the probability of a tag bit being either a '0' or '1' should be equal to each other. To fulfill the diffusion property, if a single bit of the data is changed, each bit of the tag should have an equal probability (0.5) of being flipped. Avalanche effect is another property that defines that the tag would be very different even for two very similar data, maybe differing by just a single bit. This means even if just 1-bit of data is changed, around half of the tag bits should be changed. Table 5.4 presents the results for this sneak-path tag generation method used in this work. This result is a little different from the one presented in [40] as this is regenerated for the memristor type and crossbar used for this work.

5.11 Overall Security Evaluation

5.11.1 Malicious Read

The goal of this attack is to read stored back-up data and gain sensitive information. Since we encrypt the data before backup, this would increase the complexity of learning anything from the data. As mentioned before, we use OTP based encryption. Since in an OTP based encryption, the key is random, has the same size as data, for a 16-bit data, any of the whole possible space of 2^{16} combinations are equally likely to be a key. Thus OTP is considered to not vulnerable against brute force attacks because the attacker doesn't gain any new information from the data encrypted using OTP. This is explained in [77]. For example, with an N-bit key, the possible number of key combinations, N_{key}, before and after brute force attacks are:

$$\begin{aligned} N_{key}(before) &= 2^N \\ N_{key}(after) &= 2^N \end{aligned} \tag{5.7}$$

Table 5.4: Security properties of the tag generation method

Tag size	**Uniformity**	**Avalanche**	**Diffusion**
6	0.9869	0.4953	0.4971
8	0.9637	0.4951	0.5062

Thus for any two different input data or plaintext, d_1 and d_2 in data space D, the probability of any ciphertext, c being equal to either d_1 or d_2 the same as shown in this equation 5.8.

$$P[(d_1 \epsilon D] = c) = P[(d_2 \epsilon D) = c] \tag{5.8}$$

Since brute force across the key or data space doesn't add any new information to an attacker which wasn't already available to him, OTP is said to be able to maintain perfect secrecy.

5.11.2 Malicious Write

Verifying the integrity of the backup data before restore is another security feature of this work. An attacker might launch a spoofing attack by connecting the NVM from a different power source in the network in order to perform malicious write. However, our sneak-path current based integrity checking method described earlier should be able to detect such offline modifications to this memory by generating a secure tag. The goal of the attacker thus should be to modify the memory in a way that it would produce the same tag. The success rate of this attack depends on the uniformity property of the tag and the number of trials the attacker can perform during this power failure duration. Since we have already shown that the tag generation method exhibits a good uniformity, we have have analyzed the probability of successful spoofing attack in terms of number of trials. This is shown in Figure 5.14. With increasing number of trials, the probability of finding a tag match increases. For a given number of trials, the tag match probability depends on the tag size. A hypothetical scenario that an attacker may leverage is that the backup data stored in

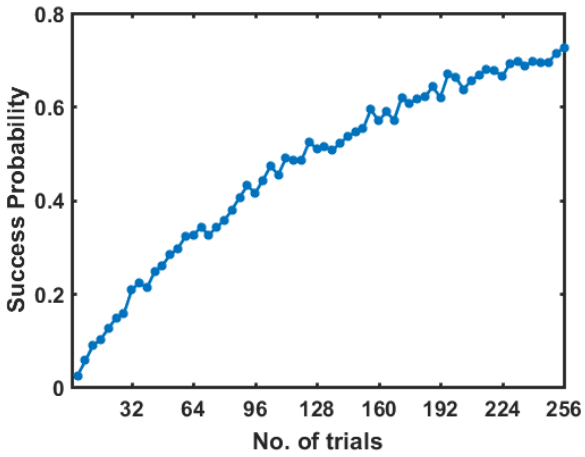


Figure 5.14: Probability of success in a spoofing attack with number of trials. The more trials an attacker can perform, the higher the chance that the data matches the tag. No. of effective trial is 1 in this protocol since the tag is updated on each cycle [70].

the NVM was unchanged for a few cycles of backup and restore. In that case, an attacker can perform multiple numbers of spoofing trials on the data in order to be successful where the data matches the tag. However, our tag generation protocol randomly reconfigure its reserved bits [40] as a timestamp before every backup stage and generate a new tag even if the data remains the same. Therefore, in our security protocol the effective number of trials to perform a spoofing attack is 1. The success probability for a 8-bit tag with a single trial is nearly $1/2^8 = 1/256$. This is sufficient due to the consideration that the tag is updated on each backup and restore cycle regardless of the data and the attacker cannot perform multiple trials on guessing a data-tag pair. For the the same reason, this protocol can also prevent replay attack where an attacker remembers a previous data and tag and replace the present (data, tag) pair with that. Since each data has a large number of variants for the tag depending on the timestamp, a tag from a data at one time would be very different than the tag from the same data at a different time.

5.11.3 Modeling Attacks

In this work, we are assuming that the attacker can read anything from NVM which means he can gain access to the stored PUF challenge as well. Researchers have already shown strong

PUFs can be modeled using only a subset of the total CRPs [62, 25]. We have also shown before (in Chapter 4) how a 32×2 XbarPUF (with HfO_2 memristors) can also be modeled and predicted with high accuracy and also developed mitigation techniques to reduce that accuracy significantly [71]. Here, we recreate the work with the TiO_2 memristor models and circuit parameters used to design this whole security architecture. We have used python's machine learning toolbox, the scikit-learn [54] and presented the results using four different classification models in Table 5.5. We have collected 5000 CRPs from an abstract model of the XbarPUF, developed beforehand in [71] and then used 2/3rd of the data for training and the rest for testing. It is clear from Table 5.5 that the accuracy is almost like a random guess of a coin flip (50%) for all of these models, namely support vector machine (SVM) with Gaussian or radial basis function (RBF) kernel, logistic regression (LR), naive Bayes with Gaussian kernel, and AdaBoost ensemble.

5.11.4 Readout/Alteration of Secure Information

The random challenge to generate PUF response and the tag generated from the backup data are also saved in NVM. Here we pessimistically assume that an attacker can also gain access to these sensitive information from the NVM. Now without a good prediction model, knowing the PUF challenge wouldn't compromise the PUF response. We already know that the prediction accuracy using modern modelings models against our XbarPUF is almost like a random guess and thus wouldn't reveal any information to the attacker. Moreover, changing the challenge itself would change the PUF response significantly which can be inferred from the very good uniformity and bit-aliasing values of this PUF. Also, because of the good collision property of the tag using our tag generation protocol, it is very difficult to find the data that produces the same tag even if an attacker can read the stored tag from the NVM. Finally, because of good avalanche property of the tag, even if a single tag bit is changed, almost half of the data bits would change, thus making it resilient against such adversarial modifications.

Table 5.5: Results from modeling attacks using different machine learning algorithms against TiO_2 memristor based XbarPUF [70]

	SVM (RBF)		L. R.		Gauss. N. B.		AdaB. Ensem.	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
Accuracy (%)	51.27	49.48	56.17	50.16	56.31	50.25	56.16	50.24

5.12 Overall Performance Analysis

Generating a reliable key from the partially-reliable XbarPUF response is one of the most critical components of this design. We are accepting the fact that there would be some unreliable PUF response bits and we size the XbarPUF to get at least the required number of reliable bits (16-bit here) for the key from the PUF response (32-bit). These extra bits present an overhead to our design. To determine and predict the minimum required number of extra bits, we have run Monte Carlo simulation. For our XbarPUF, we know that memristors' cycle-to-cycle (C2C) variation is the root cause of producing unstable bits or bit-flips. The bit-flipping probability would be higher in cases where a pair of memristor's C2C variation dominate over their process variation. Moreover, since our target application here is deployable embedded system or IoT, we also need to consider drastic environmental changes. To emulate this situation, we have considered the case when temperature changes rapidly from room temperature to 50^0C above room temperature between successive clock cycles. A change in supply voltage changes the switching speed of memristors but since we are using a large enough switching time to ensure complete state transition, this voltage change shouldn't an effect on the final memristive states and thus the PUF response. Therefore, we have only considered C2C variation and temperature change for this analysis.

The average numbers of response bits it takes to produce 16 'clean' key bits are shown in Figure 5.15 for three different C2C variation parameter of memristors. As we have already discussed, C2C variation is the main culprit behind producing unreliable responses from our XbarPUF, we have used three different sets, 2%, 5%, and 10%. From Figure 5.15, we can see that with increasing number of C2C variation, the minimum number of required bits increases. However, because of the resiliency of our XbarPUF design against environmental changes, the amount of extra bits needed is small (≈0.8) even for a 10% C2C variation, with

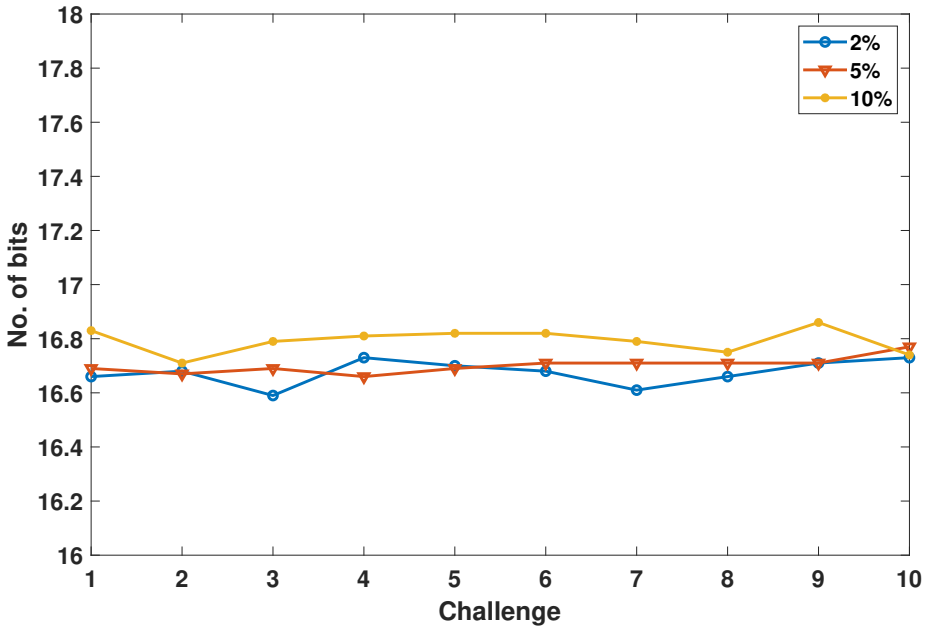


Figure 5.15: Figure shows the average number of bits to produce 16 'good' bits from 50 different chips. The results are generated for three different cycle-to-cycle variations and a 50^0C temperature change [70].

50^0C temperature change. The goal of our design is to produce 16 error-free bits in two consecutive cycles of encryption and decryption in one back-up and restore session.

After getting an idea about the minimum number of extra response bits, we are now interested in learning about bit-flipping probabilities of all the response bits. We have applied multiple different challenges in the same chip, each for 500 cycles using Monte Carlo analysis. The percentage of time that the response bits flipped are shown in Figure 5.16 for 10 different challenges and for a particular chip. As we can clearly see, there are only one or two bits per challenge which have a significantly large bit-flip probability which can be identified and discarded during functionality testing of the chip. The purpose of our all-agree voting based reliability enhancement technique is to mitigate the impact of bit-flips that remain undetected during device testing and later cause a bit-flip during run-time, resulting in a key error. From this figure, we can also see that our key generation method is capable of producing 30 'clean' key bits on average from a 32-bit response. To account for higher bit-errors/bit-flips in some chips, the ratio of number of required key bits with

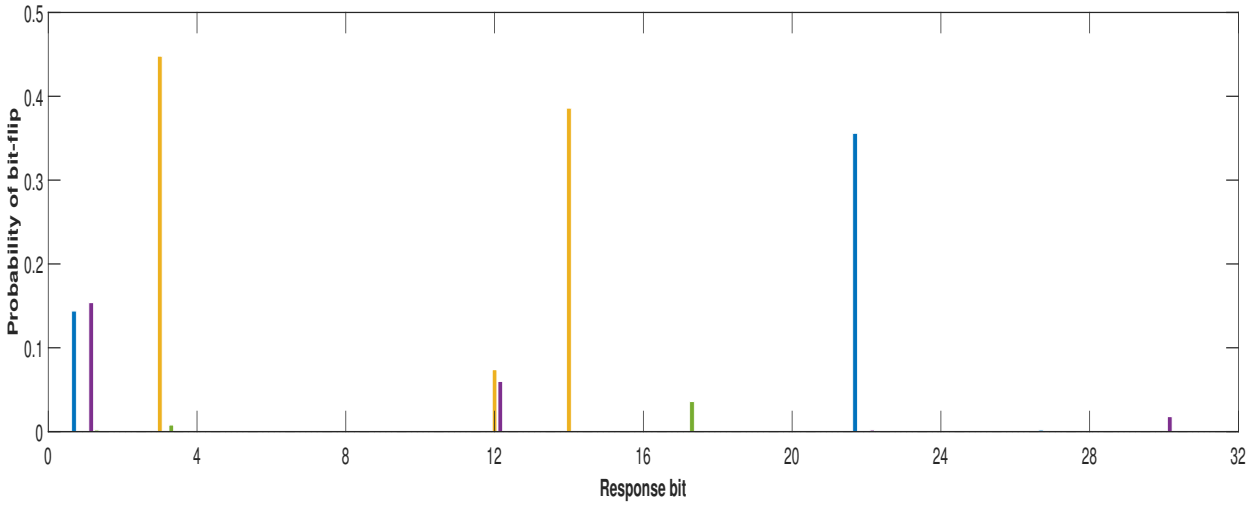


Figure 5.16: Probability of bit-flip for a same chip for 10 unique challenges. Plots showing the bit-flip probabilities for all 32 bits from a PUF response, evaluated for 500 cycles. Different challenges cause different bits to flip i.e. there is no single set of globally unreliable bit [70].

number of response bits may be decreased which would help increase the yield of the design, at the expense of extra overhead.

The area, power, and delay overhead for in different phases and different blocks of our security architecture are shown in Table 5.6, 5.7, and 5.8, respectively. From table 5.6, one can think the required area overhead of our designed system is large. However, if we take a closer look, we can see that except for XbarPUF, most other circuit blocks like SRAM, decoder, RRAM, counters etc. are actually common parts of any regular processor and memory and thus can be reused for our security implementation. We have designed all these blocks in CMOS $65nm$ technology where the minimum length and width of a transistor are chosen as $60nm$ and $120nm$ respectively. All digital circuits are sized to have minimum area while buffers are added to match their drive strengths with corresponding required loads. Analog circuits like sense amplifiers, pass-gates, analog MUXes etc. are sized accordingly, usually larger than digital circuits, to minimize the impact of mismatch and noise. Table 5.7 presents the power consumption in different phases of the security system in terms of overall current. As we have mentioned before, XbarPUFs have large static power consumption in the state where all the memristors are reset to HRS from LRS. Therefore, during state 1

Table 5.6: Total area in terms of transistor count for different components of our security architecture [70]

Design block	Component	Count	Comment
XbarPUF ×4	Memristor	64×16	$10nm \times 10nm$
	Sense amplifier	×8	9-T cell (base width 1.2 μm) [73]
	Row controller	×64	8 pass-gates
	Column controller	×8	10-T 2-R
Enc.-Dec.	XOR	× 16	
Tag gen.	MUX2to1 (regular)	3×12	$120nm$ NMOS and PMOS
	MUX2to1 (wide)	3×12	$12\mu m$ NMOS and PMOS
	Sense Amplifier	×12	9-T cell (base width 1.2 μm) [73]
	Resistor	×12	load resistor = $\sqrt{HRS * LRS}$
memTRNG	Memristor, NMOS	×2	$(10nm \times 10nm)$
	Differential op-amp	×1	5-T cell (base width 1.2μm)
RRAM	1T1R	5×12	1 memristor, 1-NMOS (4.8μm)
	Sense amplifier	×12	9-T cell (base width 1.2 μm) [73]
	Pass gate	×12	large (12μm) NMOS & NMOS
	Decoders	×1	4to16 & 2to4 decoders
SRAM ×2	SRAM cell	×32	6-T cell ($120nm$,$240nm$)
	Pre-charge	×8	3-PMOS (1.2μm)
	Sense Amplifier (SA)	×8	Current Latched SA (9-T) [29]
	Column buffer	×8	4-T (1.2μm), 2-NOT
	Address decoder	×8	2to4 & 3to8 decoder
	Basic gates	×8	(AND, OR, NOT); min. width
Others	Counters, Buffers, flops, Non-overlapping clock generator, state decoders, basic gates etc.		

Table 5.7: Power consumption of the security architecture in different stages (State 1 and 2 involve a reset of the whole memristor crossbar, 64×64 at once and thus have large static current) [70].

State	Average current (μA)	Comment
State 1	143.9	PUF response generation + SRAM-1 write
State 2	151.8	PUF response generation (again) + SRAM-2 write
State3to10	0.143	Both SRAM read + RRAM write
State11	3.28	Tag generation
Total	24.80	Overall average current

Table 5.8: Delay overhead of the system in different stages of operation

State	Clock cycles	Comment
PUF RESET	0.5	8×
PUF challenge	0.25	8× short spike
PUF read	0.25	8×
SRAM write	4	overlapped with PUF read
RRAM write	16+x	for 16 clean bit + 'x' bit error
RRAM read	16	for 16 bit

and 2, the power consumption of the overall system is high but it is fairly low during other times. Thus, the average current of the whole system is roughly 24.80μA overall with a 0.85V supply voltage (VDD). We can further reduce this current by utilizing a dual or multi-voltage scheme, with very small VDD for digital circuits and separate sufficiently larger VDD for analog and memristive components.

The required delay of our security protocol is shown in Table 5.8 in terms of clock cycles. We already mentioned there are four XbarPUFs in our prototype design and each one is activated twice during both encryption and decryption. Thus it takes m×4×2 clock cycles to generate responses from these XbarPUFs where m is the number of clock cycles required to generate a response from one XbarPUF which has a design delay of 1 clock cycle (reset, challenge, and read are done in one clock cycle). Each row of SRAM (8 bits) is written in one cycle and thus our 4×8 SRAM requires 4 clock cycles for a complete write. Since this is overlapped with the XbarPUF response generation, this doesn't incur any additional delay. The slowest operation of our design is when the encrypted data is written one bit at a time into the RRAM. For a 'n'-bit key, it would require n clock cycles ('n' is 16 for this prototype system). In practice, this operation would require more than n clock cycles as 'x' number of unreliable or noisy bits would add 'x' extra clock cycles. However, this delay can be reduced further by allowing to write multiple bits together into the RRAM. Tag generation unit takes 3 clock cycles. During decryption, RRAM is read one bit at a time, resulting in a total of exact 'n' clock cycles (again n=16 here). The time required during decryption thus would be almost the same (minus 'x' clock cycles to account for bit errors) as the encryption as both operations are very similar.

We have also compared the resource overhead of our designed security protocol against several traditional and lightweight security mechanisms suitable in embedded system domain. Table 5.9 lists the resource overhead for several such encryption algorithms, AES [51], nanoAES [44], SIMON [6] and, PRESENT [10]. To calculate the area overhead, we have omitted typical circuit blocks that would already be present in a processor (e.g. counters, SRAMs, registers etc.). However, the delay and power consumption from these blocks are taken into consideration. Although our designed system can generate 30 clean bits (out of 32) on average, one can easily generate a larger key by applying more challenges and appending the responses together. This would inevitably increase the delay and energy requirement while the overall area and average power consumption would be the same. Alternatively, each individual PUF block can be made larger to generate a larger key with no little to no additional delay at the expense of larger area and power overhead.

It is important to note that the overhead for existing cryptographic techniques reported in Table 5.9 are without the overhead associated with generating and storing the key. Thus their actual implementation in a real hardware can be expected to have even larger resource overhead than in Table 5.9. However, this key generation is one of the main contributing factor to overhead in our designed security architecture. This comes at the special advantage of random unique keys for each system while the key doesn't need be stored physically. Overall, our designed security protocol with implemented architecture provides a very secure and lightweight way of performing data backup and restore in the NVM of a resource-constrained IoT system.

Table 5.9: Performance comparison with state-of-the-art lightweight hardware security techniques [70]

Overhead	Encryption-Decryption				
	AES [51]	nanoAES [44]	SIMON [6]	PRESENT [10]	Ours [70]
Avg. Power (μW)	18.5	170	-	-	21.08
Delay (clk. cyc./bit)	1.75	2.62	6.67	0.5	0.27
Area (NAND G.E.)	2400	2090	763	1570	856

Chapter 6

Conclusion and Future Plan

6.1 Future Works

I have designed and improved hardware security primitives like PUF and TRNG built from emerging memristive technology. Then a novel security vulnerability for IoT devices is shown and a complete security protocol as well as security architecture built from these pieces of hardware security is proposed. Different devices with different security requirements, resource limitation, and application space would require different ways of implementing security. Our security protocol is designed in such a way which is, however, can be easily modified to fit into different such systems. Finally, characteristics and security properties of hardware security modules could depend heavily on a particular circuit design and fabrication technology and only after actual physical implementation, we'll be able to test, verify, and explore the full benefits and shortcomings of our design. It would also be interesting to see how this designed security architecture actually performs when added to any existing embedded process in an IoT device. That way, we can also get a real sense of the security benefit as well as performance overhead of our system. Depending on the availability of power source, resource limitation, and required level of security, a lot of research can be done on how to better fit our design into various systems.

6.2 Summary

I have identified a security vulnerability of resource constrained IoT device in this work and designed a very lightweight security protocol using hardware enabled security primitives with emerging memristive technology. Starting from device modeling improvement to circuit design and analyses, I have built a complete security architecture for this domain as well to get a realistic idea about real-world implementation difficulties. The several different components of my work are listed below:

- A memristor model is improved to add environmental impact on the device. Specifically temperature dependence on HRS, LRS and threshold voltages are introduced in the device model as well as LRS and HRS aging. Aging is also added. Data for these relationships are collected from literature for HfO_2 memristors.

- A crossbar memristor PUF is redesigned to improve its performance and to comply with the updated memristor model. A detailed analysis of this PUF is then presented where the security and performance of the design against process variation and varying operating condition are analyzed. Scalability of the circuit, choice of different circuit parameters are explored and improvement of the design with different device parameters are also analyzed to give device engineers a possible future direction.

- Abstract high-level abstract models for the memristor and memristor crossbar PUF are developed. Then machine learning algorithms are used to perform modeling attacks on this PUF. Circuit design of the PUF is also proposed to improve robustness against these machine learning attacks.

- A practical sense amplifier suitable for memristive crossbar circuits is proposed. The circuit topology is first compared with other topology to show its effectiveness in these applications. Then a detailed Monte Carlo yield analysis along with power, delay and area estimation are used to choose the best transistor size for this analog component.

- A comprehensive theoretical analysis of memristor based true random number generator (TRNG) is provided and how randomness is affected by the presence of

process variation and temperature, voltage variation and aging is also discussed using statistical mathematics. Finally an improved twin memristor based TRNG is proposed which works better in varying environmental condition and even where large process variation is present.

- A run-time reliability enhancement algorithm for PUFs is shown as well. Circuit implementation for such technique is also shown. Moreover, detailed analyses are provided on to choose the best of parameters to get a balance between resource usage and reliablity of operation.

- A complete security protocol based on hardware security for resource-limited IoT systems is provided. The architecture is also designed at transistor level to get an accurate overhead estimation and practical implementation difficulties. Our designed security protocol provides a practical level of security which is lightweight as IoT domain requires. Detailed security and overhead analyses are performed and compared with existing lightweight security techniques to show its advantages as well.

Bibliography

[1] Agrawal, D., Archambeault, B., Rao, J. R., and Rohatgi, P. (2003). The EM side—channel(s). In Kaliski, B. S., Koç, ç. K., and Paar, C., editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 29–45, Berlin, Heidelberg. Springer Berlin Heidelberg. 14

[2] Association, S. I. (2013). International technology roadmap for semiconductors (ITRS): Emerging research materials. (MSU-CSE-06-2):1–102. 8, 10

[3] Association, S. I. (2015). International technology roadmap for semiconductors (ITRS) 2.0: Executive report. (MSU-CSE-06-2):1–79. 8

[4] Balatti, S., Ambrogio, S., Wang, Z., and Ielmini, D. (2015). True random number generation by variability of resistive switching in oxide-based devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):214–221. 25

[5] Barenghi, A., Breveglieri, L., Koren, I., and Naccache, D. (2012). Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076. 14

[6] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. (2013). The simon and speck families of lightweight block ciphers. *IACR Cryptology ePrint Archive*, 2013(1):404–449. 143

[7] Beckmann, K., Holt, J. S., Cady, N. C., and Nostrand, J. V. (2015). Comparison of random telegraph noise, endurance and reliability in amorphous and crystalline hafnia-based ReRAM. In *2015 IEEE International Integrated Reliability Workshop (IIRW)*, pages 107–110. 31

[8] Beckmann, K., Manem, H., and Cady, N. (2016). Performance enhancement of a time-delay PUF design by utilizing integrated nanoscale ReRAM devices. *IEEE Transactions on Emerging Topics in Computing*, page 1. xii, 27, 28, 30, 48

[9] Benoist, A., Blonkowski, S., Jeannot, S., Denorme, S., Damiens, J., Berger, J., Candelier, P., Vianello, E., Grampeix, H., Nodin, J. F., Jalaguier, E., Perniola, L., and Allard, B.

(2014). 28nm advanced cmos resistive RAM solution as embedded non-volatile memory. In *2014 IEEE International Reliability Physics Symposium*, pages 2E.6.1–2E.6.5. 31

[10] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., and Vikkelsoe, C. (2007). Present: An ultra-lightweight block cipher. In *International workshop on cryptographic hardware and embedded systems*, pages 450–466. Springer. 143

[11] Cabout, T. et al. (2013). Temperature impact (up to 200C) on performance and reliability of hfo_2-based RRAMs. In *5th IEEE International Memory Workshop*, pages 116–119. 30, 66

[12] Cassuto, Y., Kvatinsky, S., and Yaakobi, E. (2013). Sneak-path constraints in memristor crossbar arrays. In *IEEE Int. Symp. on Inform. Theory Proc. (ISIT)*, pages 156–160. 80

[13] Chang, M. F., Wu, C. W., Kuo, C. C., Shen, S. J., Lin, K. F., Yang, S. M., King, Y. C., Lin, C. J., and Chih, Y. D. (2012). A 0.5v 4mb logic-process compatible embedded resistive RAM (ReRAM) in 65nm CMOS using low-voltage current-mode sensing scheme with 45ns random read time. In *IEEE International Solid-State Circuits Conference*, pages 434–436. 47

[14] Chen, A. (2016). A review of emerging non-volatile memory (NVM) technologies and applications. *Solid-State Electronics*, 125:25–38. 8

[15] Chua, L. O. (1971). Memristor-the missing circuit element. *IEEE Trans. on Circuit Theory*, 18(5):507–519. 11

[16] Chua, L. O. and Kang, S. M. (1976). Memristive devices and systems. *Proc. of the IEEE*, 64(2):209–223. 11

[17] Department, S. R. (2019). Internet of things - number of connected devices worldwide 2015-2025. 5

[18] Fang, Z., Yu, H. Y., Liu, W. J., Wang, Z. R., Tran, X. A., Gao, B., and Kang, J. F. (2010). Temperature instability of resistive switching on HfO2 -based RRAM devices. *IEEE Electron Device Lett.*, 31:476–478. 66

[19] Fujisaki, Y. (2013). Review of emerging new solid-state non-volatile memories. *Japanese Journal of Applied Physics*, 52(4R):040001. 8, 9

[20] Gao, S., Zeng, F., Wang, M., Wang, G., Song, C., and Pan, F. (2015). Implementation of complete boolean logic functions in single complementary resistive switch. *Scientific reports*, 5:15467. 10

[21] Gassend, B., Clarke, D., van Dijk, M., and Devadas, S. (2002). Silicon physical random functions. In *Proc. of the 9th ACM Conf. on Comput. and Commun. Security*, pages 148–160. 16

[22] Guajardo, J., Kumar, S. Schrijen, G., and Tuyls, P. (2007). Physical unclonable functions and public-key crypto for FPGA IP protection. In *Proc. of the IEEE Int. Conf. on Field Programmable Logic and Applicat.*, pages 189–195. 18

[23] Hori, Y., Yoshida, T., Katashita, T., and Satoh, A. (2010). Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs. In *2010 International Conference on Reconfigurable Computing and FPGAs*, pages 298–303. IEEE. 20, 23, 130

[24] Horstmeyer, R., Vellekoop, I. M., Assawaworrarit, S., Judkewitz, B., and Yang, C. (2013). Physical key-protected one-time pad. *Scientific Reports, Nature*, 3(3543). 107, 118

[25] Hospodar, G., Maes, R., and Verbauwhede, I. (2012). Machine learning attacks on 65nm arbiter PUFs: Accurate modeling poses strict bounds on usability. In *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 37–42. 137

[26] Jiang, H., Belkin, D., Savelev, S. E., Lin, S., Wang, Z., Li, Y., Joshi, S., Midya, R., Li, C., Rao, M., et al. (2017). A novel true random number generator based on a stochastic diffusive memristor. *Nature communications*, 8(1):882. xiv, 10, 25, 61, 62, 110

[27] Kavehei, O., Hosung, C., Ranasinghe, D., and Skafidas, S. (2013). mrPUF: A Memristive Device based Physical Unclonable Function. *ArXiv e-prints*. 20

[28] Kim, J., Ahmed, T., Nili, H., Duy Truong, N., Yang, J., Jeong, D. S., Sriram, S., Ranasinghe, D. C., and Kavehei, O. (2017). Nano-Intrinsic True Random Number Generation. *ArXiv e-prints*. 25

[29] Kobayashi, T., Nogami, K., Shirotori, T., and Fujimoto, Y. (1993). A current-controlled latch sense amplifier and a static power-saving input buffer for low-power architecture. *IEEE J. of Solid-State Circuits*, 28(4):523–527. 47, 55, 141

[30] Kocher, P., Jaffe, J., and Jun, B. (1999). Differential power analysis. In *Annual International Cryptology Conference*, pages 388–397. Springer. 14

[31] Kocher, P., Jaffe, J., Jun, B., and Rohatgi, P. (2011). Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1):5–27. 14

[32] Koeberl, P., Kocabas, U., and Sadeghi, A.-R. (2013). Memristor PUFs: A new generation of memory-based physically unclonable functions. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 428–431. 19

[33] Lee, A., Lin, C.-C., Yang, T.-C., and Chang, M.-F. (2015). An embedded ReRAM using a small-offset sense amplifier for low-voltage operations. In *VLSI Design, Automation and Test (VLSI-DAT), 2015 Int. Symp. on*, pages 1–4. IEEE. 47

[34] Lee, J. W., Lim, D., Gassend, B., Suh, G. E., van Dijk, M., and Devadas, S. (2004). A technique to build a secret key in integrated circuits for identification and authentication applications. In *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, pages 176–179. 17

[35] Lim., D. (2004a). Extracting Secret Keys from Integrated Circuits. MSc thesis. Master's thesis, MIT, Massachusetts, USA. 17

[36] Lim., D. (2004b). Extracting Secret Keys from Integrated Circuits. MSc thesis. Master's thesis, MIT, Massachesetts, USA. 23

[37] Ma, K., Li, X., Li, S., Liu, Y., Sampson, J. J., Xie, Y., and Narayanan, V. (2015a). Nonvolatile processor architecture exploration for energy-harvesting applications. *IEEE Micro*, 35(5):32–40. 98

[38] Ma, K., Zheng, Y., Li, S., Swaminathan, K., Li, X., Liu, Y., Sampson, J., Xie, Y., and Narayanan, V. (2015b). Architecture exploration for ambient energy harvesting nonvolatile processors. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 526–537. IEEE. 98

[39] Maiti, A., Gunreddy, V., and Schaumont, P. (2013). A systematic method to evaluate and compare the performance of physical unclonable functions. In Athanas, P., Pnevmatikatos, D., and Sklavos, N., editors, *Embedded Systems Design with FPGAs*, pages 245–267. Springer New York. 20, 23, 130, 132

[40] Majumder, M. B., Hasan, M. S., Uddin, M., and Rose, G. S. (2018). A secure integrity checking system for nanoelectronic resistive ram. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 1–14. 25, 104, 128, 134, 136

[41] Majumder, M. B., Uddin, M., Rose, G. S., and Rajendran, J. (2016). Sneak path enabled authentication for memristive crossbar memories. In *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, pages 1–6. 113

[42] Manem, H. and Rose, G. S. (2011). A read-monitored write circuit for 1T1M multi-level memristor memories. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2938–2941. 20, 34, 36, 110

[43] Manifavas, C., Hatzivasilis, G., Fysarakis, K., and Rantos, K. (2013). Lightweight cryptography for embedded systems–a comparative analysis. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 333–349. Springer. 15

[44] Mathew, S., Satpathy, S., Suresh, V., Anders, M., Kaul, H., Agarwal, A., Hsu, S., Chen, G., and Krishnamurthy, R. (2015). 340 mv–1.1 v, 289 gbps/w, 2090-gate NanoAES hardware accelerator with area-optimized encrypt/decrypt gf (2 4) 2 polynomials in 22 nm tri-gate cmos. *IEEE Journal of Solid-State Circuits*, 50(4):1048–1058. 143

[45] Mazady, A., Manem, H., Rahman, M., Forte, D., and Anwar, M. (2015). Memristor PUF - a security primitive: Theory and experiment. *IEEE J. on Emerging and Selected Topics in Circuits and Syst.*, 5(8):222–229. 20

[46] McDonald, N. R. (2012). Al/Cu_xO/Cu memristive devices: Fabrication, characterization, and modeling. Master's thesis, College of Nanoscale Science and Engineering, University at Albany, Albany, NY. 65

[47] McDonald, N. R., Bishop, S. M., Briggs, B. D., Van Nostrand, J. E., and Cady, N. C. (2012). Influence of the plasma oxidation power on the switching properties of Al/Cu_xO/Cu memristive devices. *Solid-State Electronics*, 78:46–50. 28, 65, 89

[48] Medeiros-Ribeiro, G., Perner, F., Carter, R., Abdalla, H., Pickett, M. D., and Williams, R. S. (2011). Lognormal switching times for titanium dioxide bipolar memristors: origin and resolution. *Nanotechnology*, 22(9):095702. xi, 86, 114, 115

[49] Mohammad, B., Dadabhoy, P., Lin, K., and Bassett, P. (2012). Comparative study of current mode and voltage mode sense amplifier used for 28nm SRAM. In *24th Int. Conf. on Microelectronics (ICM)*, pages 1–6. IEEE. 47, 49, 50

[50] Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117. 1

[51] Moradi, A., Poschmann, A., Ling, S., Paar, C., and Wang, H. (2011). Pushing the limits: A very compact and a threshold implementation of AES. In Paterson, K. G., editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 69–88, Berlin, Heidelberg. Springer Berlin Heidelberg. 143

[52] Naous, R., Al-Shedivat, M., and Salama, K. N. (2016). Stochasticity modeling in memristors. *IEEE Transactions on Nanotechnology*, 15(1):15–28. 25

[53] Pappu, R., Recht, B., Taylor, J., and Gershenfeld, N. (2002). Physical one-way functions. *Science*, 297(5589):2026–2030. 1, 16

[54] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. 91, 94, 137

[55] Pouyan, P., Amat, E., and Rubio, A. (2014). Reliability challenges in design of memristive memories. In *5th European Workshop on CMOS Variability (VARI)*, pages 1–6. 29, 30, 31

[56] Rennie, D. and Sachdev, M. (2013). SRAM sense amplifier. US Patent 8,536,898. 55

[57] Rose, G. and Meade, C. (2015). Performance analysis of a memristive crossbar PUF design. In *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. 10, 20, 32, 36

[58] Rose, G. S., Majumder, M. B., and Uddin, M. (2017). Exploiting memristive crossbar memories as dual-use security primitives in iot devices. In *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 615–620. 5

[59] Rose, G. S., McDonald, N., Yan, L., and Wysocki, B. (2013a). A write-time based memristive PUF for hardware security applications. In *Proc. of the IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pages 830–833. 19, 37

[60] Rose, G. S., McDonald, N., Yan, L., Wysocki, B., and Xu, K. (2013b). Foundations of memristor based PUF architectures. In *Proc. of the IEEE/ACM Int. Symp. on Nanoscale Architectures (NANOARCH)*, pages 52–57. 19, 37

[61] Rose, G. S., Rajendran, J., Manem, H., Karri, R., and Pino, R. E. (2012). Leveraging memristive systems in the construction of digital logic circuits. *Proceedings of the IEEE*, 100(6):2033–2049. 10

[62] Ruhrmair, U., Solter, J., Sehnke, F., and Xu, X. (2013). PUF modeling attacks on simulated and silicon data. *IEEE Trans. on Inform. Forensics and Security*, 8:1876–1891. 23, 89, 96, 129, 137

[63] Strukov, D. B., Snider, G. S., Stewart, D. R., and Williams, R. S. (2008). The missing memristor found. *Nature*, 453:80–83. 1, 12

[64] Su, Y., Holleman, J., and Otis, B. P. (2008). A digital 1.6 pj/bit chip identification circuit using process variations. *IEEE Journal of Solid-State Circuits*, 43(1):69–77. 23

[65] Suh, G. and Devadas, S. (2007). Physical unclonable functions for device authentication and secret key generation. In *44th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 9–14. 18

[66] Suh, G. E., O'Donnell, C. W., Sachdev, I., and Devadas, S. (2005). Design and implementation of the AEGIS single-chip secure processor using physical random functions. In *Proc. of the 32nd Annual Int. Symp. on Comput. Architecture (ISCA)*, pages 25–36. 17, 23

[67] Uddin, M., Hasan, M. S., and Rose, G. S. (2019). On the theoretical analysis of memristor based true random number generator. In *Accepted for the Proceedings of the 2019 on Great Lakes Symposium on VLSI*, pages –. x, xiv, 62, 71, 72, 110

[68] Uddin, M., Majumder, B., and Rose, G. S. (2018). Nanoelectronic security designs for resource-constrained internet of things devices: Finding security solutions with nanoelectronic hardwares. *IEEE Consumer Electronics Magazine*, 7(6):15–22. xii, 5, 6

[69] Uddin, M., Majumder, M. B., Beckmann, K., Manem, H., Alamgir, Z., Cady, N. C., and Rose, G. S. (2017a). Design considerations for memristive crossbar physical unclonable functions. *J. Emerg. Technol. Comput. Syst.*, 14(1):2:1–2:23. x, xii, xiv, xv, 28, 30, 31, 32, 36, 48, 61, 65, 67, 74, 75, 76, 77, 78, 80, 81, 83, 84, 110

[70] Uddin, M., Majumder, M. B., Hasan, M. S., and Rose, G. S. (2020). A secure back-up and restore for resource-constrained IoT based on nanotechnology. *IEEE Internet of Things Journal*. xi, xv, xvi, 104, 105, 106, 108, 109, 123, 124, 131, 132, 133, 136, 138, 139, 140, 141, 143

[71] Uddin, M., Majumder, M. B., and Rose, G. S. (2017b). Robustness analysis of a memristive crossbar PUF against modeling attacks. *IEEE Transactions on Nanotechnology*, 16(3):396–405. x, xi, xii, xv, 25, 33, 84, 86, 87, 90, 94, 95, 97, 128, 129, 137

[72] Uddin, M., Majumder, M. B., Rose, G. S., Beckmann, K., Manem, H., Alamgir, Z., and Cady, N. C. (2016). Techniques for improved reliability in memristive crossbar PUF

circuits. In *IEEE Comp. Society Annual Symp. on VLSI (ISVLSI)*, pages 212–217. xii, 32, 33, 34, 36, 65, 86, 89, 90

[73] Uddin, M. and Rose, G. S. (2018). A practical sense amplifier design for memristive crossbar circuits (PUF). In *2018 31st IEEE International System-on-Chip Conference (SOCC)*, pages 209–214. x, xiii, xiv, 51, 52, 55, 56, 57, 141

[74] Uddin, M., Shanta, A., Majumder, M. B., Hasan, M. S., and Rose, G. S. (2019). Memristor crossbar PUF based lightweight hardware security for IoT. In *IEEE International Conference on Consumer Electronics (ICCE)*. xi, xv, 117, 118, 119, 120

[75] Walczyk, C., Walczyk, D., Schroeder, T., Bertaud, T., Sowinska, M., Lukosius, M., Fraschke, M., Wolansky, D., Tillack, B., Miranda, E., and Wenger, C. (2011). Impact of temperature on the resistive switching behavior of embedded HfO_2 -based RRAM devices. *IEEE Trans. on Electron Devices*, 58:3124–3131. 30

[76] Wang, H., Forte, D., Tehranipoor, M. M., and Shi, Q. (2017). Probing attacks on integrated circuits: Challenges and research opportunities. *IEEE Design & Test*, 34(5):63–71. 14

[77] Wikipedia (2020). One-time pad. 134

[78] Xu, X. and Burleson, W. (2014). Hybrid side-channel/machine-learning attacks on PUFs: A new threat? In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6. 23

[79] Yang, C., Liu, B., Wang, Y., Chen, Y., Li, H., Zhang, X., and Sun, G. (2016). The applications of nvm technology in hardware security. In *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*, pages 311–316. ACM. 25

[80] Yang, J. J., Zhang, M., Strachan, J. P., Miao, F., Pickett, M. D., Kelley, R. D., Medeiros-Ribeiro, G., and Williams, R. S. (2010). High switching endurance in TaOx memristive devices. *Applied Physics Letters*, 97(23):232102. 83, 86, 104

[81] Yeung, J. and Mahmoodi, H. (2006). Robust sense amplifier design under random dopant fluctuations in nano-scale CMOS technologies. In *IEEE Int. SOC Conf.*, pages 261–264. 50, 54

[82] Zhang, D., Zeng, L., Gao, T., Gong, F., Qin, X., Kang, W., Zhang, Y., Zhang, Y., Klein, J. O., and Zhao, W. (2017). Reliability-enhanced separated pre-charge sensing amplifier for hybrid CMOS/MTJ logic circuits. *IEEE Transactions on Magnetics*, 53(9):1–5. 47

[83] Zhang, L., Fong, X., Chang, C.-H., Kong, Z. H., and Roy, K. (2014a). Highly reliable memory-based physical unclonable function using spin-transfer torque MRAM. In *IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, pages 2169–2172. IEEE. 19

[84] Zhang, L., Kong, Z. H., Chang, C., Cabrini, A., and Torelli, G. (2014b). Exploiting process variations and programming sensitivity of phase change memory for reconfigurable physical unclonable functions. *IEEE Transactions on Information Forensics and Security*, 9(6):921–932. 19

[85] Zhao, W., Chappert, C., Javerliac, V., and Noziere, J. P. (2009). High speed, high stability and low power sensing amplifier for MTJ/CMOS hybrid logic circuits. *IEEE Trans. on Magnetics*, 45(10):3784–3787. 47, 48, 55

Appendices

A Calculation of Optimal Load Resistance for XbarPUF

We already know these relations from Chapter 3.

$$R_1 = \frac{HRS}{2N}, R_2 = \frac{HRS}{N} \parallel \frac{LRS}{N}. \tag{1}$$

$$V_{eq,R_1} = \frac{R_{ld}}{R_1 + R_{ld}} * V_{read}, \quad V_{eq,R_2} = \frac{R_{ld}}{R_2 + R_{ld}} * V_{read} \tag{2}$$

$$\begin{aligned} \Delta V_R = V_{eq,R_2} - V_{eq,R_1} &= (\frac{R_{ld}}{R_2 + R_{ld}} - \frac{R_{ld}}{R_1 + R_{ld}}) * V_{read} \\ &= (\frac{R_2}{R_2 + R_{ld}} - \frac{R_1}{R_1 + R_{ld}}) * V_{read} \end{aligned} \tag{3}$$

By differentiating ΔV_R from equation 3 with respect to R_{ld} and setting the expression to zero, we get the equation for optimal load resistance.

$$\begin{aligned} & \frac{d}{dR_L}(\Delta V_R) = 0 \\ \implies & \frac{d}{dR_L}(\frac{R_2}{R_2 + R_{ld}} - \frac{R_1}{R_1 + R_{ld}}) * V_{read} = 0 \ \ [from\ equation\ (3))] \\ \implies & \frac{-R_2}{(R_2 + R_{ld})^2} + \frac{R_1}{(R_1 + R_{ld})^2} = 0 \ \ [after\ differentiation] \\ \implies & \frac{R_1}{(R_1 + R_{ld})^2} = \frac{R_2}{(R_2 + R_{ld})^2} \\ \implies & (R_2 + R_{ld})^2 = \frac{R_2}{R_1}(R_1 + R_{ld})^2 \\ \implies & R_2 + R_{ld} = \pm\sqrt{\frac{R_2}{R_1}}(R_1 + R_{ld}) \\ \implies & R_{ld}(1 - \sqrt{\frac{R_2}{R_1}}) = \sqrt{R_1 R_2} - R_2 \\ \implies & R_{ld}(\frac{\sqrt{R_1} - \sqrt{R_2}}{\sqrt{R_1}}) = \sqrt{R_2}(\sqrt{R_1} - \sqrt{R_2}) \\ \implies & R_{ld} = \sqrt{R_1 R_2} \end{aligned} \tag{4}$$

By substituting R_1 and R_2 from equation 1, we get

$$R_{ld,best} = \sqrt{R_1.R_2} = \sqrt{\frac{HRS}{2N} * (\frac{HRS}{N} \parallel \frac{LRS}{N})} \tag{5}$$

B Memristor (HfO_2) Model Used in This Work

B.1 Verilog-A Model for HfO_2 Memristor:

```
// bipolar model for HfOx memristors
// Adapted from MATLAB
// written by: Nathan McDonald, AFRL/RITB, Rome, NY
//
// Adaptation to Verilog-A:
// Garrett S. Rose, AFRL/RITA, Rome, NY
// 30-May-2014
//
// contribution:  2014_06_03 1704est Nathan McDonald, AFRL/RITB
// contribution:  2014_08_04 1515est Jillian Hallak, Univ. of Rochester
// mod.: 2014_08_05 0930est Garrett S. Rose, Univ. of Tennessee
// mod.: 2014_09_03 1540est Garrett S. Rose, Univ. of Tennessee
// mod.: 2015_09_03 1110est Harika Manem, CNSE, SUNY Polytechnic Institute
//       -- Adaptation to Hafnium Oxide ReRAM from CNSE
////HfO2 Memristor model. This model includes HRS Aging
////


`include "constants.vams"
`include "disciplines.vams"


module memr_hfox(p, n);
  inout          p;         //positive pin
  inout          n;         //negative pin
```

```
electrical     p, n;

parameter real HRS           =  1.5e5;     // high resistance state
parameter real LRS           =  1e4;       // low resistance state
parameter real Vtp           =  0.75;      // positive threshold voltage
parameter real Vtn           = -1.0;       // negative threshold voltage
parameter real tsw_p         =  1e-8;      // time to switch under +V bias
parameter real tsw_n         =  1e-6;      // time to switch under -V bias

parameter real Rinit         =  1e4;//1e6;

parameter real HRS_rate      =  2e6;

parameter real HRS_nom_spr   =  0.001;

parameter real THRS_sp_rel   =  0.1;//0.01;
parameter real TLRS_sp_rel   =  0.1;//0.01;
parameter real Ttsw_n_sp_rel =  0.05;//0.01;
parameter real Ttsw_p_sp_rel =  0.05;//0.01;
parameter real TVtn_sp_rel   =  0.1;//0.01;
parameter real TVtp_sp_rel   =  0.1;//0.01;

parameter real LRS_Aging_Rate = 0.25e6;     //Aging rate of LRS....should be a paramet
parameter real LRS_nom_spr = 0.001;             //this also should be a parameter


      // local variables
real delR;                        // resistance spread (HRS - LRS)
real delt;                        // simulation time step
real HRS_rnd;                     // ith HRS
```

```
real LRS_rnd;                    // ith LRS
real Rm;                         // memristance
real Rm_tmp;                     // temp memristance variable
//real switch_val;                 // denote state of memristor switching
real time_last;                  // previous simularion time reading
real tsw_p_rnd;
real tsw_n_rnd;
real Vtn_rnd;                    // negative th voltage for ith switch
real Vtp_rnd;                    // positive th voltage for ith switch
real Vwr;                        // input voltage

real HRS_nom;                    // nominal high res, trends down w/ time
real LRS_nom;
real tsw_p_nom;
real tsw_n_nom;
real Vtn_nom;
real Vtp_nom;

real HRS_spread;
real LRS_spread;
real tsw_p_spread;
real tsw_n_spread;
real Vtp_spread;
real Vtn_spread;



real HRS_min_after_Aging;     // impose a lower bound on the HRS due to the Aging rate
real HRS_after_Aging;     // value of HRS after Aging phenomena
real LRS_max_after_Aging;
```

```
  real LRS_after_Aging;     //value of LRS after Aging phenomena

  //temperature parameters
  real temp_room;
  real HRS_temp_change_rate;
  real LRS_temp_change_rate;
  real HRS_spread_temp_change_rate;
  real Vt_temp_change_rate;
  //
  real HRS_spread_rate_temp;
  real temp_curr;
  real deltemp;



  integer rnd;                      // temp storage for random number seeds

analog begin
  @ ( initial_step or initial_step("dc") ) begin
    delt            = 0;
    time_last       = 0;
    //switch_val      = 0;

    rnd             = $random;      // generate random number for seed

    HRS_nom         = HRS;
    LRS_nom         = LRS;
    HRS_rnd         = HRS_nom;
    LRS_rnd         = LRS;
    tsw_p_rnd       = tsw_p;
```

```
tsw_n_rnd       = tsw_n;


Rm              = Rinit;
delR            = HRS_rnd - LRS_rnd;


HRS_spread      = abs(HRS * THRS_sp_rel);
LRS_spread      = abs(LRS * TLRS_sp_rel);
tsw_p_spread    = abs(tsw_p * Ttsw_p_sp_rel);
tsw_n_spread    = abs(tsw_n * Ttsw_n_sp_rel);
Vtp_spread      = abs(Vtp * TVtp_sp_rel);
Vtn_spread      = abs(Vtn * TVtn_sp_rel);

HRS_min_after_Aging = 4*LRS; //arbitary choice for minimum HRS
LRS_max_after_Aging = 3*LRS; //arbitary choice for maximum LRS
//LRS_Aging_Rate = 0.25e6; //Aging rate of LRS...should be slower then HRS aging
//LRS_nom_spr   =  0.001;



//initialize the parameters
temp_room = 300.15;
HRS_temp_change_rate = -0.1; //negative temperature coefficient of HRS, slower than
LRS_temp_change_rate = 0.3;
HRS_spread_temp_change_rate = 0.02; //HRS variance increases with increasing tempera
Vt_temp_change_rate = -0.01; //threshold voltage decreases with increasing temperatu

temp_curr = $temperature;
```

```
  Vtp_nom = Vtp;
  Vtn_nom = Vtn;
  Vtp_rnd     = Vtp_nom;
  Vtn_rnd     = Vtn_nom;
  HRS_spread_rate_temp = THRS_sp_rel;
  //$display("\n\nbefore s\ntemperature=%3.2r vtp= %r vtn= %r hrs spr= %r hrs_sp_rate

  ////////////////// nominal threshold voltage changes for different temperature...Lat
  Vtp_nom = Vtp_nom + Vtp_nom * Vt_temp_change_rate * ($temperature-temp_room);
  Vtn_nom = Vtn_nom + Vtn_nom * Vt_temp_change_rate * ($temperature-temp_room);
  Vtp_rnd     = Vtp_nom;
  Vtn_rnd     = Vtn_nom;


  //////change in nominal value of HRS and LRS
  HRS_nom = HRS + HRS * HRS_temp_change_rate * ($temperature-temp_room);
  LRS_nom = LRS + LRS * LRS_temp_change_rate * ($temperature-temp_room);

  //////change in nominal value of HRS spread rate
  HRS_spread_rate_temp = HRS_spread_rate_temp + HRS_spread_rate_temp * HRS_spread_temp
  HRS_spread  = abs(HRS * HRS_spread_rate_temp);


  //$display("\nafter s\ntemperature=%3.2r vtp= %r vtn= %r hrs spr= %r hrs_sp_rate = %


end

delt       = $abstime - time_last;
time_last  = $abstime;
```

```
Vwr         = V(p,n);

///// for temperature changes that occurs within a simulation
deltemp     = $temperature-temp_curr;
temp_curr   = $temperature;

Vtp_rnd     = Vtp_rnd + Vtp_rnd * Vt_temp_change_rate * deltemp;
Vtn_rnd     = Vtn_rnd + Vtn_rnd * Vt_temp_change_rate * deltemp;
HRS_spread_rate_temp = HRS_spread_rate_temp + HRS_spread_rate_temp * HRS_spread_temp_c
HRS_spread = abs(HRS * HRS_spread_rate_temp);
HRS_nom     = HRS_nom + HRS_nom * HRS_temp_change_rate * deltemp;
LRS_nom     = LRS_nom + LRS_nom * LRS_temp_change_rate * deltemp;

//$display("\nnow V=%r \ntemperature=%3.2r vtp= %r vtn= %r hrs spr= %r hrs_sp_rate = %
//


Rm          = Rm;
delR        = HRS_nom - LRS_nom;


if (Vwr >= Vtp_rnd && Rm != LRS_rnd) begin
// this is unbounded! Rm can go negative if delR is small enough!
  //switch_val = -1;
  Rm_tmp = Rm - ((delR * delt * abs(Vwr)) / abs(tsw_p_rnd * Vtp_rnd));
  if (Rm_tmp <= LRS_rnd) begin
    rnd             = $random;           // generate random seed


    ////////impose a upper bound of LRS aging factor
```

```
    LRS_after_Aging = LRS_nom + LRS_Aging_Rate*time_last;
    //$display("age = %r\n",LRS_after_Aging);
    if(LRS_after_Aging>LRS_max_after_Aging)
      LRS_after_Aging = LRS_max_after_Aging;

    // update trending upward (Aging) nominal low resistance value
    LRS_nom   = $rdist_normal(rnd,LRS_after_Aging,LRS_after_Aging*LRS_nom_spr);
    LRS_spread = abs(LRS_nom * TLRS_sp_rel);

    if (LRS_nom >= LRS_max_after_Aging)
      LRS_nom = LRS_max_after_Aging;



    // regenerates LRS, Vtn and tsw_n ONLY when switching complete...
    LRS_rnd         = $rdist_normal(rnd,LRS_nom,LRS_spread);
    Vtn_rnd         = $rdist_normal(rnd,Vtn,Vtn_spread);
    tsw_n_rnd       = $rdist_normal(rnd,tsw_n,tsw_n_spread);

    // update parameters, set switch_val to stop switching
    delR            = HRS_rnd - LRS_rnd;
    Rm_tmp          = LRS_rnd;
  end
end
else if (Vwr <= Vtn_rnd && Rm != HRS_rnd) begin
  Rm_tmp = Rm + ((delR * delt * abs(Vwr)) / abs(tsw_n_rnd * Vtn_rnd));
  if (Rm_tmp >= HRS_rnd) begin
    rnd             = $random;           // generate random seed

    //impose a lower bound of HRS aging factor
    HRS_after_Aging = HRS_nom - HRS_rate*time_last;
```

```
      if(HRS_after_Aging<HRS_min_after_Aging)
        HRS_after_Aging = HRS_min_after_Aging;

      // update trending downward (Aging) nominal high resistance value
      HRS_nom   = $rdist_normal(rnd,HRS_after_Aging,HRS_after_Aging*HRS_nom_spr);
      HRS_spread = abs(HRS_nom * HRS_spread_rate_temp);

      if (HRS_nom <= HRS_min_after_Aging)
        HRS_nom = HRS_min_after_Aging;

      // regenerate HRS, Vtp and tsw_p ONLY when switching complete...
      HRS_rnd         = $rdist_normal(rnd,HRS_nom,HRS_spread);
      Vtp_rnd         = $rdist_normal(rnd,Vtp,Vtp_spread);
      tsw_p_rnd       = $rdist_normal(rnd,tsw_p,tsw_p_spread);

      // update parameters, set switch_val to stop switching
      delR            = HRS_rnd - LRS_rnd;
      Rm_tmp          = HRS_rnd;
    end
  end
  else begin
    Rm_tmp = Rm;
  end

  Rm = Rm_tmp;

  //$display("\nRm = %r\nasdfsadf\ntemperature=%3.2r vtp= %r vtn= %r hrs spr= %r hrs_sp_

  I(p,n) <+ Vwr / Rm;
end         // end analog
```

```
endmodule   // memr_hfox
```

B.2 MATLAB Model of a Memristor for Quick Behavioral Simulation:

```
function [I,Mout] = memristor(Vmag, delt,Minit)
%% no aging, temperature dependence
%% Major parameter list, mean (default)
u_Vthp =  0.7;
u_Vthn = -1.0;
u_HRS = 300e3;
u_LRS = 30e3;
u_tswp = 1e-8;
u_tswn = 1e-6;

Vthp = u_Vthp;
Vthn = u_Vthn;
HRS = u_HRS;
LRS = u_LRS;
tswp = u_tswp;
tswn = u_tswn;

%% cycle-to-cycle distribution parameter list(default value) in percentage
Vtp_sp_rel_time = 0.005;
Vtn_sp_rel_time = 0.005;
HRS_sp_rel_time = 0.2;
LRS_sp_rel_time = 0.02;
tswp_sp_rel_time = 0.01;
tswn_sp_rel_time = 0.01;
```

```
    %% creating parameters using the mean and distribution with a normal
    %%%distribution function
    Vthp = normrnd(u_Vthp,abs(u_Vthp*Vtp_sp_rel_time));
    Vthn = normrnd(u_Vthn,abs(u_Vthn*Vtn_sp_rel_time));
    HRS = normrnd(u_HRS,abs(u_HRS*HRS_sp_rel_time));
    LRS = normrnd(u_LRS,abs(u_LRS*LRS_sp_rel_time));
    tswp = normrnd(u_tswp,abs(u_tswp*tswp_sp_rel_time));
    tswn = normrnd(u_tswn,abs(u_tswn*tswn_sp_rel_time));

    %%
    delR = HRS-LRS;

    Mout = Minit;
%     if Vmag <Vthp && Vmag>Vthn          % won't switch
%         Mout = Minit;
%     else

    %% Switching logic of memristor
    if Vmag >= Vthp && Mout > LRS
        Mout = Minit - abs(Vmag)*delR*delt/abs(tswp*Vthp);    % didn't consider magtitud
        if Mout < LRS
            Mout = LRS;
        end
    elseif Vmag <= Vthn && Mout < HRS
        Mout = Minit + abs(Vmag)*delR*delt/abs(tswn*Vthn);
        if Mout > HRS
            Mout = HRS;
        end
```

```
    end

    %%
    %Mout;
    I = Vmag/Mout;
    %I = Mout;
end
```

C Behavioral XbarPUF Model Used for Response Generation and Power Calculation

```
%% Mesbah Uddin, UT    June,2016

%% To measure uniqueness, you have to measure the same response for same challenge acros
% variable 'MC' and keep Ncycle = 1 and 'C' constant to calculate uniqueness.
% MC=1, Ncycle variable and 'C' fixed for reliability.
% MC=1, Ncycle fixed and 'C' variable for uniformity
% MC variable and others fixed for bit-aliasing

clc;clear all;close all;

directory = 'C:\Users\muddin6\Documents\machine-learning-ex2\ex2\testResult_new';
extension = '.csv';
avgunif = 0;

%% Global memristor parameters
 %% cycle-to-cycle distribution
var_in_time = [.02 .02 .02 .02 .02 .02]; %in percentage
%var_in_time = [HRS_sp_rel_time LRS_sp_rel_time Vtp_sp_rel_time Vtn_sp_rel_time tswp_sp_
```

```
  %% mean of major parameters
mean_param = [300e3 30e3 0.7 -1.0 1e-6 1e-6];
% mean_param = [u_HRS u_LRS u_Vtp u_Vtn u_tswp u_tswn];

  %% Statistical variations for mean of major parameters
stat_memr_param = [.2 .1 .1 .1 .05 .05]; % in percentage, same order as cycle-to-cycle v

%%
RROW = 32; CCOL = 0; %extra 3 bits for column swapping
ROW = 32; COL = 4;
row = ROW*2;
col = COL*2;
dim = row*col;
Rmem = zeros(1,dim);

%% monte carlo run
%C = randi([0 1],RROW,1); %same challenge for all PUFs
MC = 3;
totResp = zeros(MC,COL);
for run = 1:MC
custom_mean_param = mean_param;
%k=1;j=1;
%% generating the crossbar matrix by using instances of randomly created memristors
rng('shuffle');
for k=1:dim
    if rem(k,4) ==0
        rng('shuffle'); % to seed different numbers
    end
```

```
    custom_mean_param = create_statistical_distribution(mean_param,stat_memr_param);
    Rmem(k) = custom_mean_param(1);   %set to HRS initially
    mem(k) = Memristor(custom_mean_param,Rmem(k),var_in_time);
end
mem = reshape(mem,[row col]);
Rmem = reshape(Rmem,[row col]);
%[I, Mout] = memristance(mem(1,1),Vmag,step)

backup_mem = mem;
%% Start applying challenge
C = randi([0 1],ROW,1);

NCycle = 2;
totResp2 = zeros(NCycle,COL);

bigcsv = zeros(NCycle,RROW+CCOL);   %extra 1 column for saving the response bits
bigcsv_cm = zeros(NCycle,RROW+CCOL); % cm = column mix
bigcsv_xor = zeros(NCycle,RROW+CCOL/2);
for iter = 1:NCycle %increase this number to calculate uniformity, reliability

mem = backup_mem;

%C = []
%C = input('');
C = randi([0 1],RROW,1);
%disp(C')
% save challenges to a big matrix to write in a csv file later
bigcsv(iter,1:end-CCOL) = C';
bigcsv_cm(iter,1:end-CCOL) = C';
```

```
%% %%%%%%%%%% Beginning of experiment
%% Major circuit parameter
Vwr = 1.3;
Vrd = 0.60;


%% voltage applied across the row and load resistance
v = zeros(row,1);
Rload = zeros(1,col);
Rload(Rload==0) = calcRload(mean_param(1),mean_param(2),ROW);
Vresp = zeros(1,col);
Resp = zeros(1,COL);
Resp2 = zeros(1,COL);
%%
tper = 1e-6;
points_per_cycle = 50;
step = tper/points_per_cycle ;         %50 points per cycle

%% RESET
TReset = 2e-6;
no_reset_cycle = 1;
t_res = step:step:TReset*no_reset_cycle;
v(v==0) = -Vwr;

reset_time = length(t_res);

for k=1:reset_time
    for rr = 1:row
        for cc = 1:col
            [I, Rmem(rr,cc)] = mem(rr,cc).memristance(v(rr),step,Rmem(rr,cc));
```

```
            mem(rr,cc).Mout = Rmem(rr,cc);
            %Rmem(rr,cc)
        end
    end
end

%% CHALLENGE
TChallenge = 0.6e-6;
no_chal_cycle = 1;
t_ch = step:step:TChallenge*no_chal_cycle;

%%choose a random challenge
%C = zeros(row,1);
%C = randi([0 1],row/2,1); used as input
Cb = C;%zeros(row/2,1); %minus 3 cause of control challenges
%size(Cb)
C(C==1) = Vwr;
Cb(C==0) = -Vwr;
%disp('here')
v = zeros(row,1);
%size(Cb)
v(1:2:row) = C(1:end);
v(2:2:row) = Cb(1:end);

C;
Cb;
v;
%C = [0;0;0;0]

%%apply challenge
```

```
chal_time = length(t_ch);
%fprintf('\n---\n\n\n');
for k=1:chal_time
    for rr = 1:row
        for cc = 1:col
            [I, Rmem(rr,cc)] = mem(rr,cc).memristance(v(rr),step,mem(rr,cc).Mout);
            mem(rr,cc).Mout = Rmem(rr,cc);
            %Rmem(rr,cc)
        end
    end
end

%% READ
vr = zeros(1,col);
vr(vr==0) = Vrd;
v = vr;

TRead = 1e-7;
t_rd = step:step:TRead;
format short
Rmem;
Req = 1./sum(1./(Rmem));
%fprintf('Req = %3.3f \n',Req);
%Req = 1./sum(1./([Rmem;Rload]));
% fprintf('Req = %3.3f\n',Req);
Vresp = Vrd*Rload./(Rload+Req);
for k=1:COL
    Resp(k) = Vresp(2*k-1)>Vresp(2*k);
end
%fprintf('%d',Resp);
```

```
%fprintf('\n');

%% perform swap
Req2 = columnSwap(logical(C(end-3:end)),Req); %last 3 bits of challenge for swap control
Vresp = Vrd*Rload./(Rload+Req2);
for k=1:COL
    Resp2(k) = Vresp(2*k-1)>Vresp(2*k);
end



totResp(iter,:) = Resp;
for i=0:CCOL-1
    bigcsv(iter,end-i) = Resp(i+1); %Resp;
end

totResp2(iter,:) = Resp2;
%fprintf('%d',xor(Resp(1),Resp(2)));
for i=0:CCOL-1
    bigcsv_cm(iter,end-i) = Resp2(i+1); %Resp;
end

fprintf('%d iteration finished...\n',MC*(run-1) + iter);
end

totResp(run,:) = Resp;
%fprintf('\n-------------%3.2f%% complete-------------\n',run*100/MC);

  csvfile = strcat(directory,MC+'0',extension);
  csvwrite(csvfile,bigcsv);
```

```
  csvfile2 = strcat(directory,MC+'0','swap',extension);
  csvwrite(csvfile2,bigcsv_cm);

  %avgunif = avgunif + uniformity(totResp2)
end
```

D How to Perform Tests on the Chip

We have fabricated a complete chip with 65nm CMOS technology containing HfO_2 memristors. There are several stand-alone 'test structures' on this chip that can be probed and tested separately. For example, we have single memristor circuit, forming and set-reset circuit, sense amplifier circuit, 1-bit PUF circuit, and so on. Figure 1 shows a snapshot of the layout of our designed chip where some of the test structures are clearly visible. There are several things that we need to setup correctly to properly test and verify functionality of any of these circuits. Taking the sense amplifier circuit as an example, the different steps of testing are described in this section.

D.1 Setting up the Probepad Connection

Each of the individual stand-alone test structures contain 24 pads, arranged as two face-to-face rows of 12 pads each. The dimension of each of these pads are 60μm $\times$ μm while having a 40μm gap between consecutive pads. These are landing pads, pure chunks of metal, meaning a 12×2 probepad can be connected directly to these pads to apply different input patterns and also get the output from the underlying test circuits. After inserting a test chip into the probe station and after setting up all 24 probes, they are moved slowly to be just on top of the desired test structure and then pushed down until the probes slightly touch the landing pads on the chip to form electrical connections. Figure 2 shows one such connection.

D.2 PSOC Microcontroller and Source Meter to Generate Inputs

We have used a PSOC (programmable system on chip) microcontroller board and a source meter to apply different input patterns and connected them with the probepad connecting wires. The PSOC microcontroller that we have used for our experiment has quite a few input/output (IO) pins, more than the maximum 24 that a probepad might need. Moreover, the PSOC board provides multiple direct connections to 1.2V, 3.3V, and many connections to GND, thereby easing the need to produce these signals separately from the microcontroller. The digital I/O pins of the microcontroller use 3.3V as the VDD so any digital pulse can be produced using these pins. The number of available analog pins, however, is limited as at

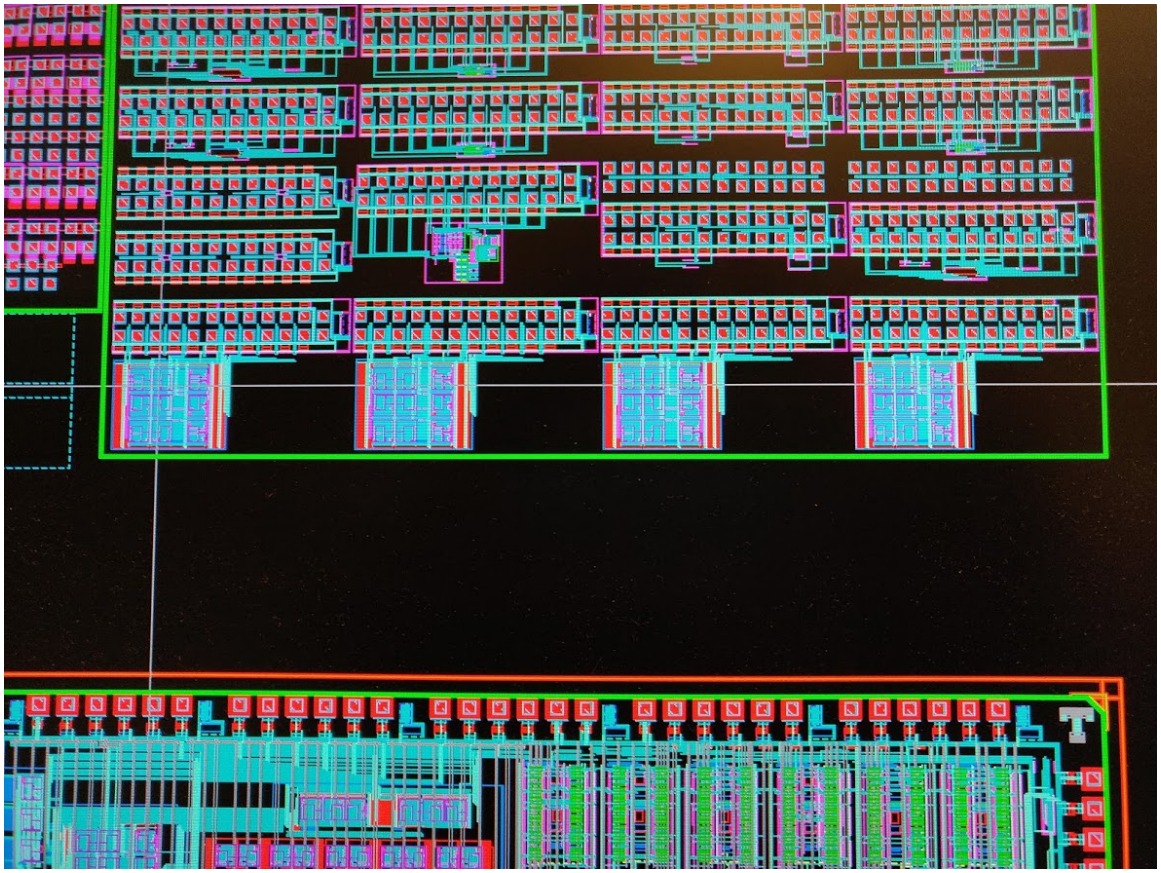


Figure 1: Partial snapshot of the layout of the chip where it contains a 12×2 probepad for the sense amplifier circuit.

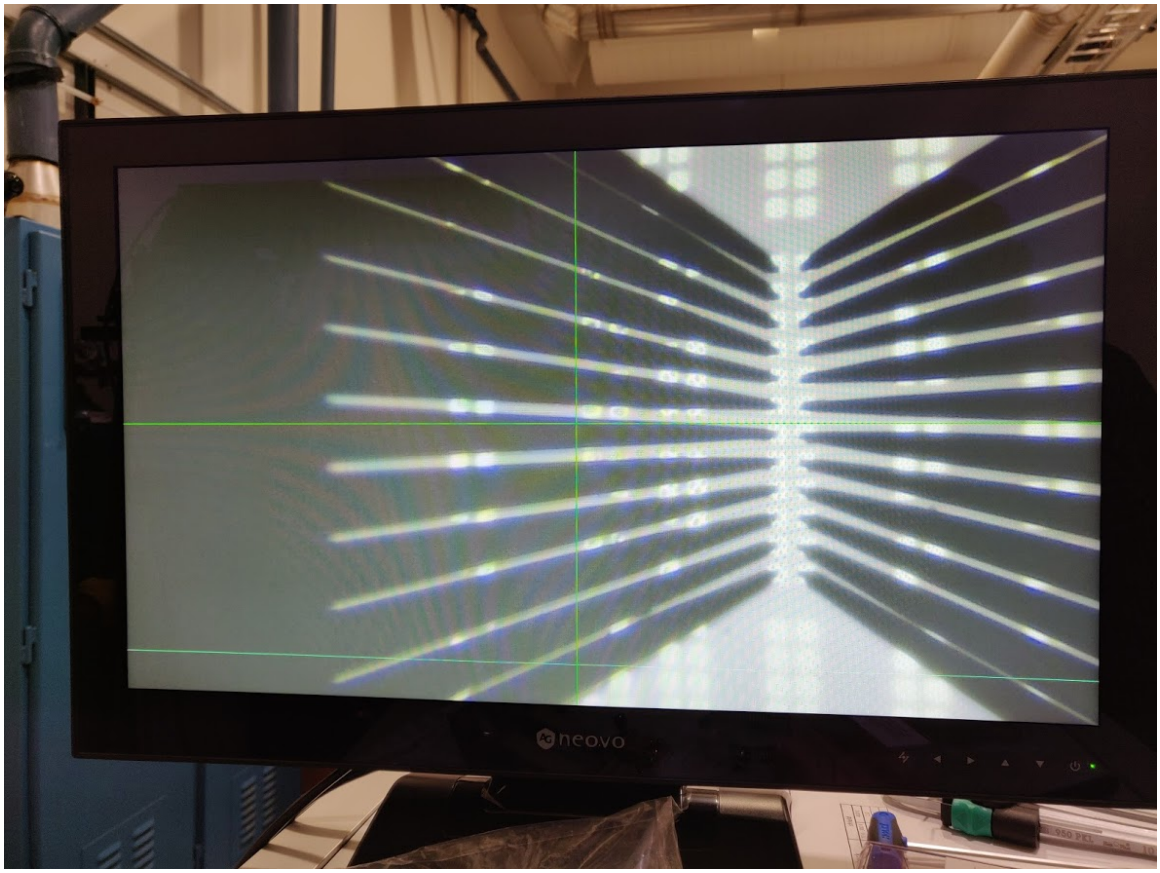


Figure 2: 12×2 probepad connection to a test circuit inside of a probe station, the image is taken from a microscopic view.

most four analog signals can be generated at the same time. The voltage range for analog signals can be between 0 to 4V. Moreover, since 65nm technology only allows maximum 1.2V VDD, these digital signals can not be produced using the regular digital I/O pins of the PSOC as those I/O can only use 3.3V as VDD. Thus we need to use the on-chip DAC (digital-to-analog converter) to generate these pulses with 1.2V VDD similar to any analog signals.

Most of our circuits are designed to have much smaller number of pin counts than the maximum of 24 for each test structure. For example, an SA test circuit only need 7 pins, VDD, GND, sense enable or 'SE', two inputs B and Bbar, two outputs O and Ob. GND and VDD pins can be directly connected to the on-board GND and 1.2V, respectively. The output pins can be connected directly to display instruments like the oscilloscope. The analog voltages B, Bar and digital SE (wih 1.2V) signal can be generated using the DAC and analog I/O of the PSOC.

The source meter that we have used has two separate channels, capable of providing arbitrary input voltage while it can also measure the current drawn from this two voltage source. Thus if we need to provide accurate analog fixed DC voltage input, we can use this source meter. Since this also measures the current supplied by the voltage as well, this is very useful to detect state change in memristors by observing the current. That's why the source meter was specifically useful at setting up different input voltages to test the memristor read-write-form circuit.

D.3 Setting up the Connection with Probepads

Only the minimum required number of pins out of 24 from probepad are connected to the PSOC board. A snapshot of this is shown in Figure 3.

D.4 Performing Simple Functionality Test

The detailed results from testing of the SA circuit are provided in Chapter 3 and we skip that here. Because of the very limited frequency range of DAC of the PSOC, we have performed very slow signals. For example, the SA is tested at only 10Hz frequency. The DC voltage

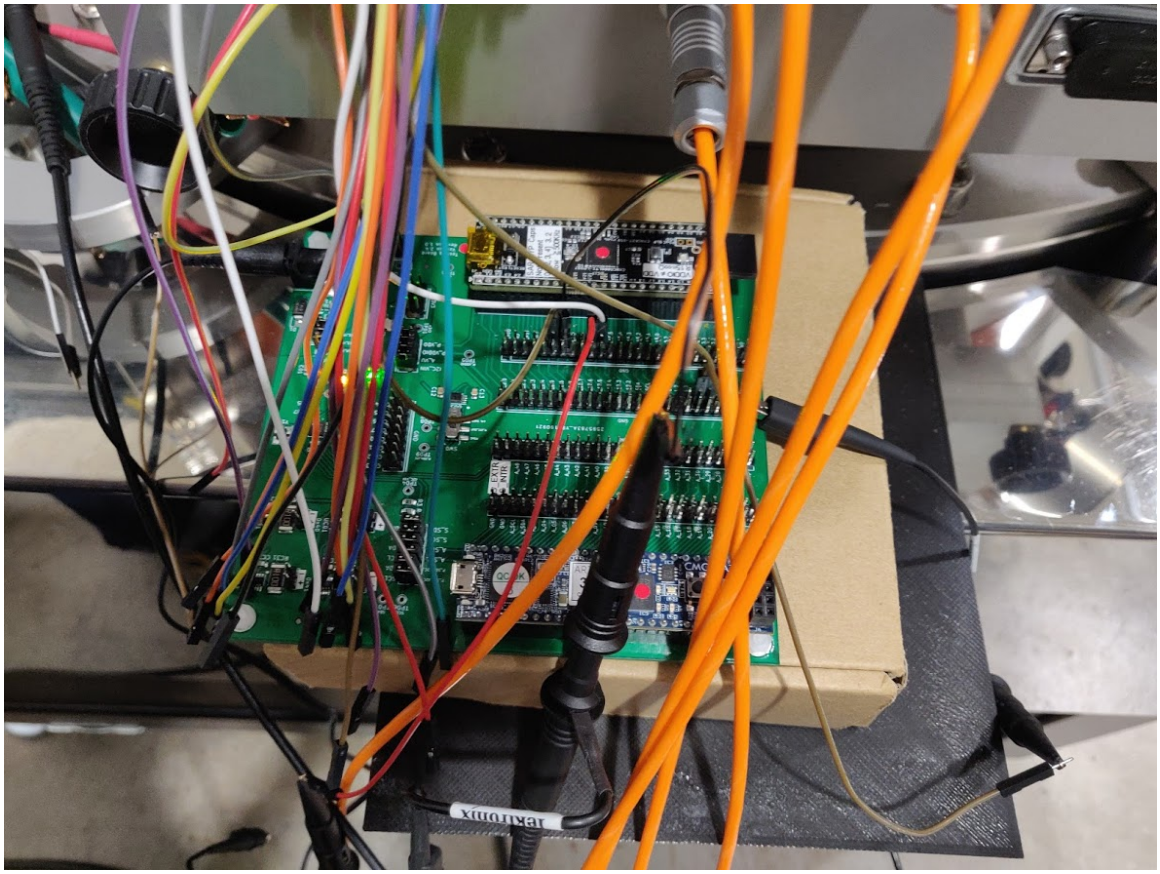


Figure 3: Connection between the inputs generated by the PSOC microcontroller and wires coming from the probepad

level generated by PSOC has a noise of around 50-100mV and thus the voltage difference between two bit-line inputs are kept around 200-400mV. This doesn't provide a full testing capability but ensures the electrical connection and the circuit itself work properly.

D.5 Oscilloscope to View and Collect the Output

The SA circuit has two inverting outputs. They have basically the same node as the inputs, but they are asserted when the 'SE' signal goes to high. They are shorted with the two inputs in other times. The outputs are connected directly to two channels of an oscilloscope and a snapshot is shown in Figure 4. Both the outputs are found to be correct all the times in this test setup. These waveforms are also saved and exported as CSV files in a computer to perform data analysis later on.



Figure 4: Output shown on a oscilloscope during testing of a sense amplifier circuit from a fabricated chip.

E Probability of Error with Majority Voting Technique

In majority voting technique, suppose a response bit is evaluated 'N' number of times. Also let's consider the probability of producing a stable output of a particular response bit is p. Thus the probability of being unstable is (1-p). Now the probability of a particular response bit to produce its valid logic state r times during N evaluation is:

$$\textbf{Prob} = p^r(1-p)^{N-r} \tag{6}$$

The number of ways to produce the stable response of a particular bit 'r' times during N evaluations can be expressed as $\binom{N}{r}$. Thus the probability of producing a valid response by producing its valid binary state exactly 'r' times among N evaluations is:

$$\textbf{Prob} = \binom{N}{r} p^r(1-p)^{N-r} \tag{7}$$

Now depending on the value of this 'r', the response cane be either valid or erroneous. This bit would be considered logic '0' if it produces '0' more than half (r>N/2) of the time. Thus '0' would be the valid state for this particular response bit. However, an error would occur if this bit fails to produce '0', its valid state N/2 or more number of times. Thus there would be error when it produce its valid state 0 times, or 1 times, or more upto less than N/2 number of times. Therefore, the probability of errors (P.E.) can be found by adding all the situations where 'r' is not greater than N/2 and is expressed here:

$$\begin{aligned} PE = & \binom{N}{0} p^0(1-p)^{N-0} + \binom{N}{1} p^1(1-p)^{N-1} + \binom{N}{2} p^2(1-p)^{N-2} + ... \\ & ..... + \binom{N}{N/2} p^{N/2}(1-p)^{N-N/2} \\ & = \sum_{r=0}^{\frac{N}{2}} \binom{N}{r} p^r(1-p)^{N-r} \end{aligned} \tag{8}$$

The probability of producing a correct output with majority voting on the other hand can be expressed similarly as:

$$\begin{aligned} P(Maj.vote) &= \binom{N}{\frac{N}{2}+1} p^{\frac{N}{2}+1}(1-p)^{N-(\frac{N}{2}+1)} + \binom{N}{\frac{N}{2}+2} p^{\frac{N}{2}+2}(1-p)^{N-(\frac{N}{2}+2)} + ... \\ &..... + \binom{N}{N} p^{N}(1-p)^{N-N} \\ &= \sum_{r=\frac{N}{2}+1}^{N} \binom{N}{r} p^{r}(1-p)^{N-r} \end{aligned} \tag{9}$$

Vita

Mesbah Uddin received his B.Sc. (2013) in Electrical and Electronic Engineering (EEE) from Bangladesh University of Engineering and Technology in Dhaka, Bangladesh. Since August, 2015, he has worked for his Ph.D. in Computer Engineering at University of Tennessee, Knoxville, Tennessee. His active area of research during this period includes hardware security, IoT security, and VLSI circuit design using emerging nanotechnology.