ABSTRACT

Title of dissertation: DESIGN, FABRICATION, AND RUN-TIME STRATEGIES FOR HARDWARE-ASSISTED SECURITY

Domenic J. Forte, Doctor of Philosophy, 2013

Dissertation directed by: Dr. Ankur Srivastava
Department of Electrical and Computer Engineering

Today, electronic computing devices are critically involved in our daily lives, basic infrastructure, and national defense systems. With the growing number of threats against them, hardware-based security features offer the best chance for building secure and trustworthy cyber systems. In this dissertation, we investigate ways of making hardware-based security into a reality with primary focus on two areas: Hardware Trojan Detection and Physically Unclonable Functions (PUFs).

Hardware Trojans are malicious modifications made to original IC designs or layouts that can jeopardize the integrity of hardware and software platforms. Since most modern systems critically depend on ICs, detection of hardware Trojans has garnered significant interest in academia, industry, as well as governmental agencies. The majority of existing detection schemes focus on test-time because of the limited hardware resources available at run-time. In this dissertation, we explore innovative run-time solutions that utilize on-chip thermal sensor measurements and fundamental estimation/detection theory to expose changes in IC power/thermal

profile caused by Trojan activation. The proposed solutions are low overhead and also generalizable to many other sensing modalities and problem instances. Simulation results using state-of-the-art tools on publicly available Trojan benchmarks verify that our approaches can detect Trojans quickly and with few false positives.

Physically Unclonable Functions (PUFs) are circuits that rely on IC fabrication variations to generate unique signatures for various security applications such as IC authentication, anti-counterfeiting, cryptographic key generation, and tamper resistance. While the existence of variations has been well exploited in PUF design, knowledge of exactly how variations come into existence has largely been ignored. Yet, for several decades the Design-for-Manufacturability (DFM) community has actually investigated the fundamental sources of these variations. Furthermore, since manufacturing variations are often harmful to IC yield, the existing DFM tools have been geared towards suppressing them (counter-intuitive for PUFs). In this dissertation, we make several improvements over current state-of-the-art work in PUFs. First, our approaches exploit existing DFM models to improve PUFs at physical layout and mask generation levels. Second, our proposed algorithms reverse the role of standard DFM tools and extend them towards improving PUF quality without harming non-PUF portions of the IC. Finally, since our approaches occur after design and before fabrication, they are applicable to all types of PUFs and have little overhead in terms of area, power, etc.

The innovative and unconventional techniques presented in this dissertation should act as important building blocks for future work in cyber security.

DESIGN, FABRICATION, AND RUN-TIME STRATEGIES FOR HARDWARE-ASSISTED SECURITY

by

Domenic J. Forte

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Professor Ankur Srivastava, Chair/Advisor
Professor Rama Chellappa
Professor Joseph F. JaJa
Professor Shuvra S. Bhattacharyya
Professor Amitabh Varshney

© Copyright by
Domenic J. Forte
2013

Dedication

To my family

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Ankur Srivastava, for his guidance and patience through my Ph.D. His advice, support, and friendship have been invaluable on both academic and personal levels, for which I am extremely grateful. Without him as a continuous source of inspiration and motivation, I would not have been able to improve and achieve all that I have. It has been a pleasure to work with and learn from him.

I would also like to extend my gratitude to the members of my dissertation committee for their time and service. I also offer special thanks to Professor Rama Chellappa, Professor Joseph JaJa, Professor, Professor Shuvra Bhattacharyya, and Professor Andre Tits for their support, encouragement, and advice in applying to jobs in academia.

My colleagues in Prof. Srivastava's research group have enriched my graduate life in many ways and deserve a special mention here as well. I owe my gratitude to Bing Shi, Caleb Serafy, Tiantao Lu, and Chongxi Bao not only for our fruitful discussions about research, but also the stress relieving conversations and jokes we shared throughout our time together in the lab.

I would also like to acknowledge the financial support provided by the ECE department and Northrop Grumman for three semesters of teaching assistantship and one year of research fellowship respectively.

Last, but by no means least, I thank all of my family and friends for their support and encouragement throughout my graduate studies. I would especially

like to express my heart-felt gratitude to my parents. I would not be the person that I am today without their unwavering support, encouragement, and love. I also owe many thanks to Stephen and Mary Ellen for being my family away from home and always being there for me.

Table of Contents

List of Figures viii

List of Abbreviations x

List of Publications xi

1 Introduction 1
1.1 Evolution of Computing Devices and Need for Cyber Security 1
1.2 Software vs. Hardware-assisted Security 2
1.3 Hardware-based Attacks 4
1.3.1 Sources of Attack in the IC Supply Chain 5
1.3.1.1 Design Phase 5
1.3.1.2 Synthesis Phase 6
1.3.1.3 Fabrication Phase 7
1.3.2 Post-deployment Attacks 8
1.4 Theme: A Comprehensive Strategy 10
1.5 Summary and Thesis Organization 14

2 Background 17
2.1 Hardware Trojan Attacks and Mitigation 17
2.1.1 Trojan Anatomy and Taxonomy 18
2.1.1.1 Anatomy of a Trojan 18
2.1.1.2 Trojan Taxonomy 19
2.1.1.3 Real-life Prototypes and Benchmarks 22
2.1.2 Trojan Mitigation Techniques and Associated Challenges . . . 22
2.1.2.1 Destructive Methods 23
2.1.2.2 Test-time Verification Methods 24
2.1.2.3 Run-time Monitoring Methods 26
2.1.2.4 Design-time Methods 27
2.1.3 Summary 28
2.2 Physically Unclonable Functions (PUFs) 30
2.2.1 PUF Structures 32
2.2.1.1 Arbiter PUF 33
2.2.1.2 Ring Oscillator PUF (RO-PUF) 35
2.2.1.3 SRAM PUF 35
2.2.2 PUF Applications in Hardware Security 37
2.2.3 PUF Quality and Metrics 38
2.2.4 IC Variations and Impact on PUFs 40
2.2.4.1 Manufacturing Process, Variations, and DFM 40
2.2.4.2 Temporal Variations 43
2.2.5 Existing PUF Research 44
2.2.5.1 Architectures 44
2.2.5.2 Circuits 45

2.2.5.3 Post-fabrication 46
2.2.6 Summary 47

3 Temperature Tracking for Run-time Detection of Trojans 49
3.1 Introduction 49
3.1.1 Motivation 49
3.1.2 Main Contributions 52
3.2 Preliminary 54
3.2.1 Thermal Sensors 54
3.2.2 RC Thermal Model 55
3.3 Problem Definition and Challenges 57
3.4 Temperature-Based Detection 59
3.4.1 Design Phase 60
3.4.2 Test Phase 61
3.4.3 Run-time Phase 63
3.4.3.1 Local Sensor Approach 63
3.4.3.2 Global (Kalman Filter-based) Approach 67
3.4.3.3 Qualitative Comparison 72
3.5 Experiments and Discussion 73
3.5.1 Setup 73
3.5.2 Results 75
3.6 Summary 77

4 Mask Generation for Physically Unclonable Functions 80
4.1 Introduction 80
4.2 Preliminary 85
4.2.1 Optical Lithography 85
4.2.2 Optical Proximity Correction (OPC) 89
4.3 Contributions and Discussion 93
4.3.1 Wafer Variability Model 93
4.3.2 Impact of Variations on PUFs 95
4.4 Optical Proximity Correction for PUFs 97
4.4.1 PUF-aware OPC (P-OPC) 98
4.4.1.1 P-OPC Objective Cost 98
4.4.1.2 P-OPC Optimization Algorithm 99
4.4.1.3 Ensuring Functional Correctness with P-OPC 101
4.4.1.4 Dealing with Systematic Variations 102
4.4.2 Systematic Variation Compensation OPC (SVC-OPC)) 103
4.4.2.1 SVC-OPC Objective Cost 105
4.4.2.2 SVC-OPC Algorithm 106
4.4.3 Qualitative Comparison of P-OPC and SVC-OPC 107
4.5 Simulation Experiments 109
4.5.1 Simulation Setup 109
4.5.1.1 Simulation Models 109
4.5.1.2 Mask Generation Algorithms 110

4.5.1.3 PUF Evaluation 111
4.5.1.4 ROs and RO-PUF Response Extraction 112
4.5.2 Results and Discussion 113
4.5.2.1 Ranking Approach 113
4.5.2.2 Decoupled Neighbor Approach 114
4.5.2.3 All Pairs Approach 116
4.5.2.4 Mask Generation Algorithm Overheads 118
4.5.3 Summary of Results 119
4.6 Summary 120

5 Custom Cell Layouts for Physically Unclonable Functions 122
5.1 Introduction 122
5.2 Preliminary 123
5.2.1 Self-compensation 123
5.2.2 Impact of Variations on PUFs. 125
5.3 Proposed Approach 127
5.3.1 Self-compensated Cell Layouts for PUFs 127
5.3.2 Reliability Enhancement 130
5.3.3 Combined Approach 134
5.4 Simulation Experiments 135
5.4.1 Experimental Setup 135
5.4.2 Results and Discussion 139
5.5 Summary 143

6 Conclusion and Future Research Directions 144
6.1 Future Research 145
6.1.1 Defense Against Trojans 146
6.1.2 Opportunities and Challenges in PUF Manufacturing 148

Bibliography 150

List of Figures

1.1 Attacks on the IC Supply Chain/Process 5
1.2 Summary of Hardware Attacks. The lefthand side is organized based on location of attack in the IC supply chain. The righthand side is organized based on the type of post-deployment attack. 11
1.3 Comprehensive Strategy for Self-sustaining Security 12

2.1 Trojan circuity: Trigger and Payload 19
2.2 Trojan Taxonomy 20
2.3 Trojan Mitigation Techniques 23
2.4 (a) Switch block constructed with MUXES controlled by challenge bit; (b) Arbiter PUF; (c) Effects of challenge bits on paths to the arbiter. 34
2.5 Ring Oscillator (RO) and Ring Oscillator PUF (RO-PUF) 36
2.6 SRAM cell and parameter mismatch between M1 and M3 (ΔL, ΔV_{th}) 37

3.1 Avg. power consumption (nW) in a 250μs time window across Trojan-inactive and Trojan-active ICs for the RS232-T900 benchmark. 51
3.2 Basic thermal sensor circuit 54
3.3 IC broken into grids and RC thermal model within dotted region . . . 56
3.4 Phases of the Proposed Approach 60
3.5 Local Hypothesis Testing (HT) Approach with z sensors and $m \times n$ grid 66
3.6 KF-based Temperature Tracking 70
3.7 Global Kalman Filter (KF) Approach with z sensors and $m \times n$ grid 71
3.8 Average autocorrelation ($\hat{a}$) over time with 4 and 32 sensors for s38417-T300 77

4.1 Summary of existing research geared towards improving PUF quality. Columns represent the three main areas of research: Systematic Variation (SV) Compensation, Random Variation (RV) Enhancement, and Environmental Variation (EV) Resistance. Rows represent the steps in the IC design/fabrication process. Starred rows denote steps where research is lacking. New approaches at mask and layout levels are the focus of Chapters 4 and 5 in this dissertation. 81
4.2 Optical Lithography System 86
4.3 Basic OPC Algorithm 90
4.4 (a) Mask: White (gray) areas correspond to transparent (opaque) pixels; (b) Fragmentation; (c) Mask perturbation 1; (d) Mask perturbation 2 93
4.5 Systematic portion of defocus across the wafer (i.e. $p_{\text{aw}}(x_w, y_w)$ + $p_{\text{af}}(x'_f, y'_f)$ where x'_f, y'_f represent the field location in wafer coordinates) 95

4.6 Keep Out Zone Illustration: Non-PUF region, PUF region, and keep out zone are shown in white, dark gray, and light gray respectively. δ denotes the length of the keep out zone. Note the figure is not drawn to scale. 101

5.1 Systematic and opposing behavior of dense and iso patterns with defocus. 124
5.2 Effective channel lengths (dotted) for cells in series. (a) dense and iso cells in series; (b) two dense cells in series 124
5.3 Standard and current starved inverters 132
5.4 Channel length vs. Defocus for cell layout types: (1) Very Dense VD; (2) Dense D; (3) Isolated I; (4) Very Isolated VI; (5) Self-Compensated S 136
5.5 Quadtree partitioning for a chip. The depth of the tree shown is 3 levels. 138

List of Abbreviations

AES	Advanced Encryption Standard
CMP	Chemical Mechanical Polishing
DFM	Design for Manufacturability
DoS	Denial of Service
ECC	Error Correcting Code
EKF	Extended Kalman Filter
EM	Electromigration
EV	Environmental Variation
FIB	Focused Ion Beam
FV	Fabrication Variation
HCI	Hot Carrier Injection
HI	Logical High (1)
IC	Integrated Circuit
ICUT	Integrated Circuit Under Test
IP	Intellectual Property
KF	Kalman Filter
LO	Logical Low (0)
NBTI	Negative Bias Temperature Instability
OPC	Optical Proximity Correction
OS	Operating System
PDF	Probability Distribution Function
PSM	Phase Shift Mask
PUF	Physically Unclonable Function
RO	Ring Oscillator
RO-PUF	Ring Oscillator Physically Unclonable Function
RV	Random Variation
RTL	Register Transfer Level
SEM	Scanning Electron Microscope
SRAM	Static Random Access Memory
SV	Systematic Variation
TDDB	Temperature-Dependent Dielectric Breakdown
TPM	Trusted Platform Module

List of Publications

Journal Publications (accepted)

1. D. Forte and A. Srivastava, "Thermal-Aware Sensor Scheduling for Distributed Estimation", to appear *ACM Transactions on Sensor Networks (TOSN)*, 2013.

2. D. Forte and A. Srivastava, "Energy and Thermal-Aware Video Coding via Encoder/Decoder Workload Balancing", to appear *IEEE Transactions on Embedded Computing Systems (TECS)*, 2013.

3. D. Forte and A. Srivastava, "Resource-Aware Architectures for Adaptive Particle Filter Based Visual Target Tracking", to appear *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2013.

Journal Publications (submitted)

1. D. Forte and A. Srivastava, "Improving the Quality of Delay-based PUFs via Optical Proximity Correction," with minor revisions to *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, March 2013.

Conference Publications (accepted)

1. D. Forte and A. Srivastava, "Manipulating Manufacturing Variations for Better Silicon-Based Physically Unclonable Functions", *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp.171-176, August 2012.

2. D. Forte and A. Srivastava, “On Improving the Uniqueness of Silicon-Based Physically Unclonable Functions Via Optical Proximity Correction”, *Design Automation Conference (DAC)*, pp.96-105, June 2012. [**DAC-2012 Best Paper Nomination**]

3. D. Forte and A. Srivastava, “Adaptable Architectures for Distributed Visual Target Tracking”, *IEEE International Conference on Computer Design (ICCD)*, pp.339-345, October 2011.

4. D. Forte and A. Srivastava, “Energy-Aware and Quality-Scalable Data Placement and Retrieval for Disks in Video Server Environments”, *IEEE International Conference on Computer Design (ICCD)*, pp.457-458, October 2011.

5. D. Forte and A. Srivastava, “Energy-aware video storage and retrieval in server environments,” *International Green Computing Conference and Workshops (IGCC)*, pp.1-6, July 2011.

6. D. Forte and A. Srivastava, “Resource-aware architectures for particle filter based visual target tracking,” *International Green Computing Conference and Workshops (IGCC)*, pp.1-6, July 2011.

7. D. Forte and A. Srivastava, “Adaptable video compression and transmission using lossy and workload balancing techniques”, *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp.145-152, June 2011. [**Awarded AHS-2011 Best Student Paper**]

8. D. Forte and A. Srivastava, “Energy-aware video coding of multiple views via workload balancing”, *NASA/ESA Conference on Adaptive Hardware and*

Systems (AHS), pp.295-302, June 2011.

9. D. Forte and A. Srivastava, “Energy and Thermal-Aware Video Coding via Encoder/Decoder Workload Balancing”, *International Symposium on Low Power Electronics and Design 2010 (ISLPED)*, pp. 207-212, August 2010.

10. D. Forte and A. Srivastava, “Thermal-Aware Sensor Scheduling for Distributed Estimation”, *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 116-129, June 2010.

Technical Reports

1. D. Forte, C. Bao, and A. Srivastava, “Temperature Tracking: An Innovative Run-Time Approach for Hardware Trojan Detection”, *Institute for Systems Research (ISR) Technical Reports (TR_2013-03)*, UMCP.

Chapter 1

Introduction

1.1 Evolution of Computing Devices and Need for Cyber Security

Continued semiconductor scaling and outsourcing of the integrated circuit (IC) design/fabrication process have resulted in electronic computing devices with greater performance at faster time-to-market and lower prices. These factors combined with the advances in connectivity between devices have revolutionized the technology landscape and enabled previously unimaginable applications. The computing systems of today are universal tools and platforms that play an increasing role in our daily lives and basic infrastructure. For instance, smart phones and other mobile devices are now widely used for communication, personal organization, online banking, navigation, and internet. Embedded computing devices are also critical components in larger systems of the financial, commercial, and military sectors. Automobiles, air traffic control systems, the power grid, national defense systems, etc. all critically depend on the computing infrastructure provided by integrated circuits (ICs).

While most of these advances have improved our everyday lives, our vulnerability to cyber attacks has also increased dramatically. For example, outsourcing of various steps of the IC design/fabrication process has made it easier for untrusted third parties to insert malicious circuits (hardware Trojans), steal intellectual property (IP), and make counterfeit electronics. At the same time, the ever decreasing

size of IC features and increasing complexity of modern designs are making it nearly impossible to detect the fake and modified ICs.

Given the pervasiveness of computing devices in commerce and defense, attacks launched against them can have potentially devastating consequences. For example, consider how much of our personal and confidential information is stored on smartphones, PCs, and remote servers. If this information falls into the wrong hands, it could put individuals at risk for identity theft. Even more alarming are the cyber and denial-of-service (DoS) attacks on networks and safety-critical systems (power grid, emergency response, defense, etc.) that can result in deaths. With the growing number of attacks looming, the need for sophisticated technology that ensures the confidentiality, integrity, and reliability of our computing systems, stored data, and IP has never been larger.

1.2 Software vs. Hardware-assisted Security

Traditionally, computing systems have relied upon software-assisted security in the form of user passwords, key encryption algorithms, anti-virus software, anti-malware software, etc. However, software-based solutions provide only limited security and may still leave systems susceptible to intelligent and well-funded attackers. In the case of encryption, the cryptographic key is a single point of failure and can be leaked by various vulnerabilities of Operating Systems (software Trojans, key-loggers, etc.). Antivirus and anti-malware software are attempts to prevent such attacks, but are also inherently flawed because they rely solely on feedback

from developers. Developers must first recognize the existence of an attack and then address it with an antivirus/anti-malware database entry or patch. However, zero-day exploits (i.e. the attacks that occur prior to developer intervention) can still subvert system security [1]. Finally, any software-assisted security solution can often be subverted by a physical attack on hardware, the platform on which the software is running. For example, an adversary can de-package a chip, drill into the inner layers of the circuit, and directly probe signal lines with the help of advanced semiconductor tools (Scanning Electron Microscope, Focused Ion Beam, etc.) [2].

Due to the limitations of software-based security measures, the role of hardware in security has been growing in recent years. For instance, secure cryptoprocessors are dedicated devices embedded in a tamper resistant package which can be used to carry out cryptographic operations and handle passwords/keys more securely. Examples include smartcards [3] and Trusted Platform Modules (TPMs) [4]. Physically Unclonable Functions (PUFs) [5] are promising solutions to many security issues due their ability to generate IC unique identifiers that are resistant to cloning attempts as well as physical tampering. Recently, Intel and McAfee have also been working on security solutions that move beyond Operating Systems (OS) and rely more heavily on underlying hardware [1].

The trend towards hardware-assisted security is driven by two advantages hardware has over software:

- *Level of Abstraction:* Since computer hardware operates at the lowest abstraction level, hardware-based solutions can be faster, more energy efficient, and

tougher to counter than those in software [1]. This also allows hardware to play roles against both hardware- and software-based attacks. Software, on the other hand, can do little against attacks on hardware (eg. microprobing).

- *Degree of Modifiability:* The ability to easily update software is both an advantage and disadvantage. On the one hand, developers can fix and patch problems. On the other hand, the same flexibility can also be exploited by attackers to install and upgrade viruses, Trojans, and malware. In contrast, hardware chips cannot be (easily) modified by an attacker once they leave the foundry.

1.3 Hardware-based Attacks

While the assistance of hardware can dramatically improve system security, even the hardware-based solutions discussed above have a flaw: they rely on the assumption that the ICs which make up computing platforms are *trustworthy.* While in the recent past this assumption was reasonable, continued outsourcing in the semiconductor industry has made it increasingly possible for counterfeit and Trojan-infected ICs to be inserted into the supply chain.

Counterfeiting is a practice that causes irrecoverable loss to the IP holder and can harm the reputation of authentic providers. Unreliability of counterfeits can also render systems that unknowingly use them *unreliable.* A hardware Trojan is "a malicious, undesired, intentional modification of an electronic circuit or design, resulting in the incorrect behavior of an electronic device when in operation - a

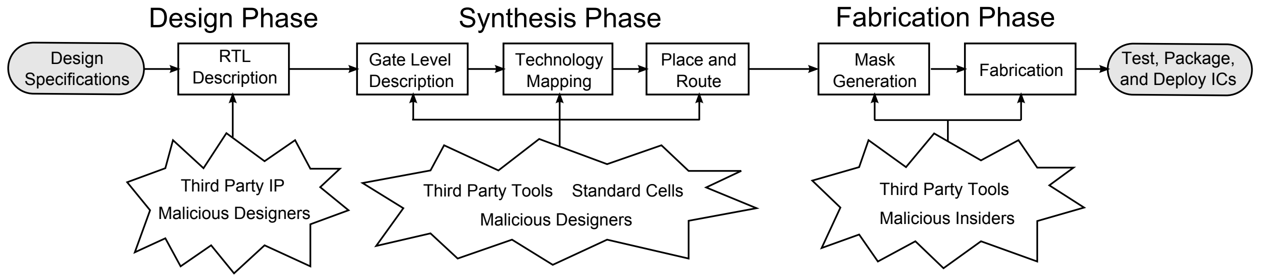


Figure 1.1: Attacks on the IC Supply Chain/Process

back-door that can be inserted into hardware" [6]. Such attacks represent a considerable danger because they allow adversaries to exploit the advantages possessed by hardware. By modifying the hardware *before* it leaves the factory, attackers can essentially *circumvent* many hardware- and software-based security features [6].

Hardware attacks occurring during IC design/fabrication and after IC deployment are discussed in more detail in the sections below.

1.3.1 Sources of Attack in the IC Supply Chain

The wide spectrum of attacks on hardware and the ease with which attacks can now occur within the IC supply chain is very concerning. Essentially, attacks on hardware are possible at any stage of the design and manufacturing process. We highlight the attackers and attacks in Figure 1.1, and we summarize them in the sections below.

1.3.1.1 Design Phase

The IC supply chain/process starts with the design phase where a designer or design team writes a register transfer level (RTL) description of the IC according

to design specifications. In the past, an IC company might have designed every component of the RTL itself. However, today most companies typically integrate third party Intellectual Property (IP) along with their own IP. Attacks in the design phase can occur through this third party IP, by adversaries who infiltrate the design process, and by malicious insiders. Infiltrators and insiders are particularly dangerous since they may have full access to the design and source files.

There are two main attacks that are possible during the design phase:

1. *Trojan insertion*: If the attacker has write access, he or she can maliciously add or remove hardware components from the original design (so-called hardware Trojan attacks). Such attacks can leak sensitive data such as cryptographic keys or reduce the reliability of the design (discussed in greater detail in Chapter 2). The third party IP may also behave unexpectedly and maliciously.

2. *IP Theft*: If the attacker only has read access, he or she can still analyze the design or steal the company's IP. Stolen IP can be used by the attacker to produce counterfeit instances of the design. Analysis of the stolen IP can be used to aid in future software or hardware-based attacks as well.

1.3.1.2 Synthesis Phase

When the design phase is complete, the RTL design is synthesized using third party software tools from Cadence, Synopsys, Mentor Graphics, etc. Synthesis begins by translating the high-level RTL to a technology-independent gate level logic diagram. The technology-independent design is then mapped to a standard

cell library and converted to a transistor-level netlist. Finally, place-and-route tools determine the physical layout of the entire design. Performance-based optimization algorithms (number of literals/gates, area, timing, power, etc.) are often employed in each abstraction level.

During the synthesis phase, an attacker can once again perform two attacks:

1. *Trojan insertion*: If the attacker has write access, he or she can insert Trojans by adding or removing gates and transistors at the various steps of the synthesis process. The attacker can also reduce the performance and reliability of the design by manipulating interconnects in the layout (increase capacitive coupling, aging effects, etc.). Even if the attacker does not have direct access to the synthesis process, the third party tools and standard cell libraries might also be untrustworthy and execute similar attacks.

2. *IP Theft*: If the attacker only has read access, he or she can gather information from netlists, layout, etc. to aid in future attacks or to create counterfeit devices.

1.3.1.3 Fabrication Phase

When synthesis is complete, the layout files are sent to a foundry, which generates lithography masks and then fabricates the chips. In the past, an IC company could use an in-house foundry (i.e. fab) facility. However, the costs to not only establish but maintain a full-scale fab have become prohibitively expensive. Thus, the use of contract (untrusted) foundries has become a necessity for all but a few

IC companies (eg. Intel). Attackers in the foundry may accomplish three different kinds of attacks:

1. *Trojan Insertion:* An attacker at the foundry will have access to the layout geometry as well as the mask generation, which will enable him or her to insert Trojans before fabrication.

2. *Overbuilding:* Current IC fabrication practices provide little to limit the number of ICs produced by the fab. Therefore, once the fab has produced the amount of ICs requested by the IC company, an attacker may be able to create additional ICs to sell on the black market. This is referred to as an overbuilding attack and results in irrecoverable loss to the IP holder.

3. *IP theft:* With the layout or masks, it is feasible (albeit expensive) for an attacker to reverse-engineer the original design. Once again, the attacker could then analyze the design for reference in future attacks or use the information to produce counterfeit instances of the design.

1.3.2 Post-deployment Attacks

Aside from vulnerabilities during design and fabrication, there are also a variety of attacks that can occur once an IC has been deployed:

1. *Trojan Activation:* While Trojans are inserted during the design or manufacturing process, the malicious circuitry is often such that the Trojan remains dormant until some triggering event (more details to come in Chapter 2). In

doing so, the adverse effects caused by the Trojan can be hidden from post-manufacturing tests, thereby increasing the likelihood that a Trojan-infected IC passes tests and is deployed into the field. Later on, when Trojan is activated, it can execute its attack (leak information, open back-doors, etc.) on an unsuspecting system.

2. *Counterfeiting:* Aside from the IP theft discussed above, counterfeiting can also be accomplished in other ways. For instance, in *IC recycling attacks*, old ICs are refurbished and promoted as new products. This is dangerous because the refurbishing process itself can harm the IC and result in premature failure of any system dependent on the IC. *Failure to adequately test* an IC for long term use and reliability (a process known as upscreening) can lead to similar issues for unsuspecting consumers and systems as well.

3. *Reverse-Engineering:* Well-funded adversaries can spend weeks using specialized equipment to de-package, de-layer, and image an IC [7]. While this process is invasive (i.e. destroys the chip), the recovered information can be exploited by attackers to understand the IC's internal features at various levels of abstraction (system level, transistor level, etc.), thereby enabling both counterfeiting and future Trojan insertion attacks.

4. *Microprobing:* The same equipment used for reverse-engineering can also be used to access the chip surface directly and observe nodes of the IC. For example, one can exploit vulnerabilities (i.e. data remanence) in "erased" non-volatile and "unpowered" volatile memory to steal sensitive/secret data

[8].

5. *Fault Injection:* Malfunctions in an IC can be triggered by exposing it to abnormal environmental conditions (intensive light pulses, radiation, local heating, etc.) and then used to infer secret information [8].

6. *Side Channel Attacks:* Side channels of a device are parameters that characterize the device's physical implementation, such as delay, power consumption, glitches, etc. Correlation existing between side channels and the data being processed by the device can be exploited by attackers to gain information about the IC. Such attacks have been shown to successfully recover keys from smartcards on the order of seconds [9], which is a far cry from the billions of years it would take to crack most encryption standards (eg. AES) by brute force methods.

1.4 Theme: A Comprehensive Strategy

The attacks discussed above are summarized in Figure 1.2 with supply chain attacks on the left and post-deployment attacks on the right. One can see that they are quite *diverse* and *interdependent*, which makes providing assurances against all of them very challenging. For instance, even if one designs a PC with a TPM [4], there's no guarantee that a Trojan won't be inserted into the TPM during synthesis or fabrication steps. When the Trojan is activated in the field, the security of the PC will be compromised. Furthermore, even if one could guarantee trust in the entire supply chain, this would not provide assurances against all post-deployment

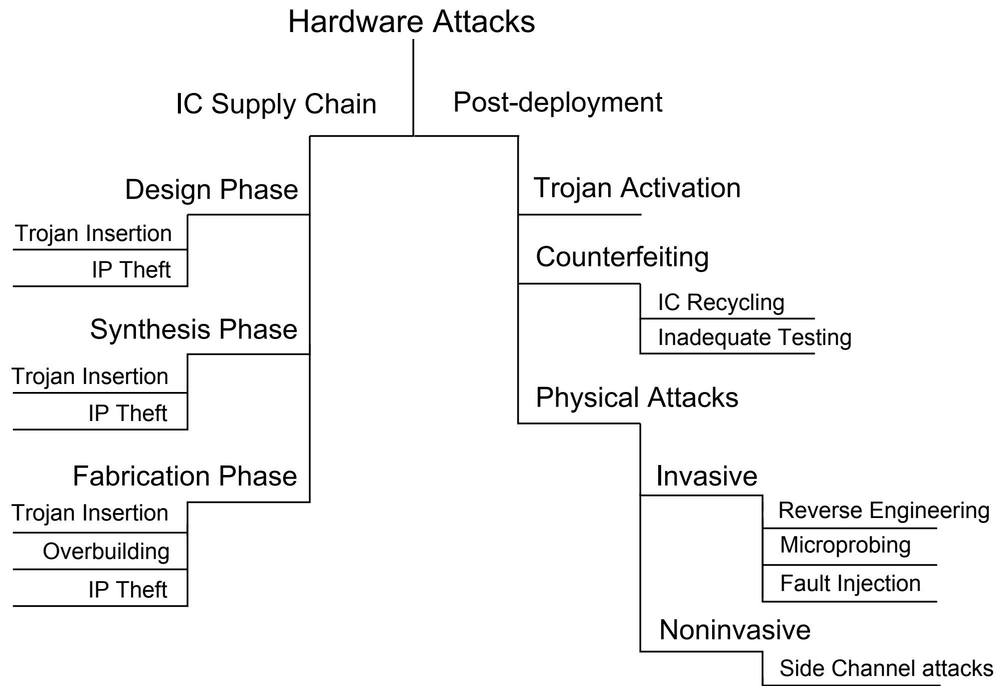


Figure 1.2: Summary of Hardware Attacks. The lefthand side is organized based on location of attack in the IC supply chain. The righthand side is organized based on the type of post-deployment attack.

attacks.

These challenges have motivated us to come up with better solutions in this dissertation. To deal with the complex interdependencies between the phases of the IC supply chain/process and the IC's post-deployment lifetime, we emphasize a comprehensive strategy. Essentially, our strategy borrows from solutions that have been used to improve yield, reliability, etc. in the past and extends them towards security. Our strategy is shown in Figure 1.3 and consists of three steps:

1. *Bootstrap:* Since ICs are basically unmodifiable after fabrication, designers have begun to rely on embedded infrastructures that improve IC performance, yield, and reliability. For example, it is now common practice to deploy error

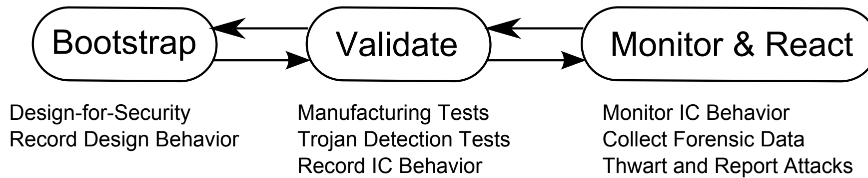


Figure 1.3: Comprehensive Strategy for Self-sustaining Security

correcting code (ECC) in memory for robustness against soft and hard errors. Our 'Bootstrap' step is essentially an extension of this concept towards security.

In the past, security has been only considered as an afterthought during design and fabrication. However, it is becoming apparent that security constraints are just as important as performance, power, etc. constraints. The main function of our 'Bootstap' step is to provide better infrastructure, circuits, and fabrication methods that support IC validation and attack forensics once chips are fabricated and deployed. For example, by fabricating sensors that monitor IC side-channel activity, we may be able to detect Trojan presence at run-time. Information leakage-resistant circuit designs, IC watermarking, and tamper-resistant structures (such as PUFs [10]), should also enable better post-deployment security. The key challenge of implementing these 'Bootstap' features is to do so with minimal overheads and impact on area, power, performance, etc.

2. *Validate:* Conventionally, post-manufacturing tests are used to verify that the gates of an IC operate as expected. The need to do this arises from the defects that might occur during chip fabrication, such as shorted and floating

nodes. The 'Validate' step in our approach uses similar procedures to test for security. For example, one can exploit recent work in the literature which detects hardware Trojans by testing for unexpected changes in logical and side-channel behavior from the original design (more details in Section 2.1.2.2). In this step, we can also collect data from (verified) Trojan-free and non-counterfeit devices to establish expected IC behavior and validate ICs at run-time. For example, one can record the challenge-response behavior of a PUF embedded in the design (more details in Section 2.2.2) to verify IC authenticity.

3. *Monitor and React:* While the pre-deployment testing performed in the 'Validate' step is extremely important, it cannot be relied on to find all IC instances containing Trojans [11, 6]. Furthermore, it cannot protect against other post-deployment attacks, such as IC recycling. The 'Monitor and React' step in our strategy aims to overcome this flaw. In this step, one can use the infrastructures and sensors (added by 'Bootstrap') to 'Monitor' the IC behavior at run-time. Data gathered from design simulations and from 'Validate' can be compared against run-time behavior, and any anomalous behavior detected might indicate a potential Trojan or counterfeit attack. Once an attack is suspected, built-in defenses could then be triggered to 'React' or thwart it before it does any damage. For example, one might react to a Trojan attack by quarantining the Trojan-infected portion of the IC before it can leak sensitive information. In the case of counterfeits, the best mode of defense might be to warn others by reporting the attack to a central database.

The above three-phase strategy is *both comprehensive and self-sustaining* in nature. It is comprehensive in that it effectively combines several techniques for attack prevention, detection, and recovery. Each step builds on infrastructure and results from the previous step to overcome security vulnerabilities in prior steps. Furthermore, the above strategy is also self-sustaining because the information obtained from latter steps can be exploited to improve future device instances. For example, forensic data on hardware Trojans obtained from the 'Monitor and React' step could be used to make future designs and tests more robust to the hardware Trojans detected in past deployments.

1.5 Summary and Thesis Organization

ICs are the basic building blocks of today's computing systems. Advancements that have created higher-performance, lower-cost ICs have also resulted in new opportunities for hardware-based attacks during IC design, manufacturing, and lifetime. Hardware-based attacks, such as Trojan insertion and counterfeit devices, are a critical danger since they have the ability to subvert many forms of software- and hardware-based security. The ever growing complexity of ICs and reliance on outsourcing have made ensuring IC security more difficult than ever. In parallel, modern computing hardware, tools, and imaging technology, are also making it easier to attack ICs after they are deployed.

In this dissertation, we investigate new solutions to overcome attacks during IC design, manufacturing, and post-deployment lifetime. Specifically, we focus on

two areas:

- *Hardware Trojans:* We study hardware Trojans because they are among the most challenging attacks to prevent, detect, and counter. First, they can be inserted at *any untrusted step* of the IC supply chain/process. Second, they are extremely challenging to detect because of their *small size and stealthy nature.* Third, the sheer *number* of attacks against both hardware *and* software that can be executed or aided by a hardware Trojan is immense. Furthermore, the persistence of hardware Trojan attacks (i.e. the threat is present as long as the infected IC is in use) makes detection of their presence extremely important. In this dissertation, we present a novel Trojan detection approach [12] which is a basic instance of the comprehensive strategy shown in Figure 1.3. Specifically, we combine new and old infrastructures, a new side channel metric (temperature), and advanced statistical techniques to estimate an IC's internal state at run-time and detect hardware Trojan attacks as they occur.

- *Physically Unclonable Functions (PUFs):* We also investigate PUFs [5], which are embedded components in ICs that extract IC manufacturing variations to generate *unique, random, and unclonable signatures.* At a relatively small overhead, the PUF signatures can be used to deal with the IP theft, overbuilding, counterfeiting, and physical attacks [13] shown in Figure 1.2. Our work with respect to PUFs is an instance of the 'Bootstrap' step in the comprehensive strategy (see Figure 1.3). Specifically, we leverage existing models

(which are largely ignored in current PUF research) to summarize the sources of IC variation as well as their impacts on PUFs. Automated approaches for physical layout and mask generation are then re-investigated in the context of these variations in order to improve PUF quality and make PUFs more effective against post-deployment attacks.

Organization. The rest of the dissertation is organized as follows. In Chapter 2, we discuss the background of hardware Trojans and PUFs in greater detail, including the current state-of-the-art research and associated shortcomings. Chapter 3 discusses the motivation and details of our proposed Trojan detection approach, which is a basic instance of the comprehensive strategy shown in Figure 1.3. In Chapters 4 and 5, we discuss our innovative techniques for enhancing PUF quality over the current state-of-the-art methods. We summarize the conclusions of our work and discuss future work in Chapter 6.

Chapter 2

Background

In this chapter, we discuss the background and key challenges of research related to Hardware Trojans and Physically Unclonable Functions (PUFs).

2.1 Hardware Trojan Attacks and Mitigation

The emerging trend of outsourcing integrated circuit (IC) design and fabrication has created new opportunities called hardware Trojan attacks that can seriously jeopardize the integrity of any electronic system. As discussed in the previous chapter, anyone with access to the IC manufacturing process, which includes design, synthesis, and fabrication, can make malicious alterations to the original or intended circuity which expose system hardware and software to various attacks. In this section, we discuss the background of hardware Trojans and current research directions. The section is organized as follows:

- Hardware Trojans have various physical, activation, and attack characteristics which shall determine their effects on a computing system. Our ability to defend against Trojans relies heavily on our understanding of these characteristics. In Section 2.1.1, we summarize Trojan anatomy, taxonomy, and real-life examples to frame our discussion of Trojan research in latter sections and chapters.

- Due to the serious threat that hardware Trojans pose to all systems and sectors dependent on ICs, detection of hardware Trojans has garnered significant interest not only in academia, but also in governmental agencies and industry. In Section 2.1.2, we discuss the current state-of-the-art methods for Trojan prevention and detection as well as their associated challenges and shortcomings.

- Finally, we conclude with a summary of the section.

2.1.1 Trojan Anatomy and Taxonomy

2.1.1.1 Anatomy of a Trojan

An example of Trojan circuitry is shown in Figure 2.1. Trojans typically consist of two components [14]:

- *Trojan Trigger:* The trigger waits for a special event and then activates the Trojan's attack. Common events include rare external input patterns and internal logic states. Before the Trojan is triggered, the IC containing the Trojan functions mainly as intended (excluding the trigger's activity). Figure 2.1 is an illustration of Trojan circuitry with a NAND gate acting as the trigger. As long as at least one of the n input nets is a logic 0 (LO), the Trojan will remain inactive and the net of the original circuit will remain unchanged.

- *Trojan Payload:* After the Trojan is triggered, the IC's functionality is changed by the Trojan Payload. In Figure 2.1, the payload is an AND gate. When

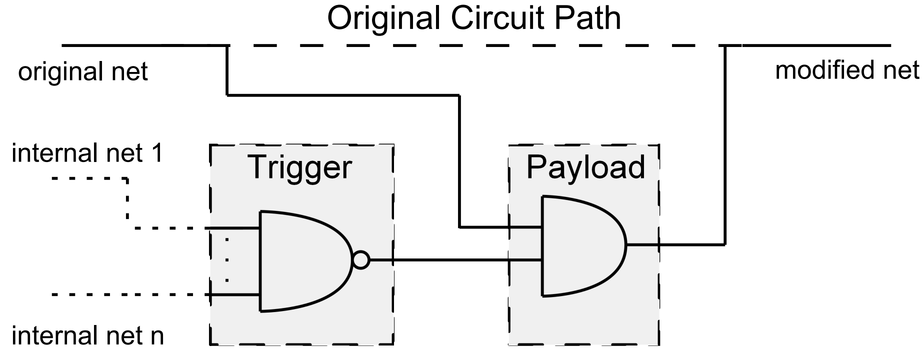


Figure 2.1: Trojan circuity: Trigger and Payload

the Trojan is triggered and the original net signal is logic 1 (HI), the payload changes the net to a LO signal.

In this dissertation, we refer to ICs with Trojans as **Trojan-inserted** or **Trojan-infected**. We refer to ICs without Trojans as **Trojan-free**. Trojan-inserted ICs whose payloads have been triggered and not triggered are referred to as **Trojan-active** and **Trojan-inactive** respectively.

2.1.1.2 Trojan Taxonomy

In order to facilitate development of Trojan detection and mitigation schemes, there have been various hardware Trojan taxonomies proposed in the literature. Our taxonomy (inspired from [15, 14]) is shown in Figure 2.2 and classifies Trojans according to physical, activation, and action characteristics:

1. *Physical Characteristics* describe the hardware manifestations of Trojans and are subdivided into four categories: Distribution, Structure, Size, and Type. *Distribution* denotes how the Trojan appears in the layout (e.g. spread out or isolated). *Structure* refers to the changes to the original layout caused by the

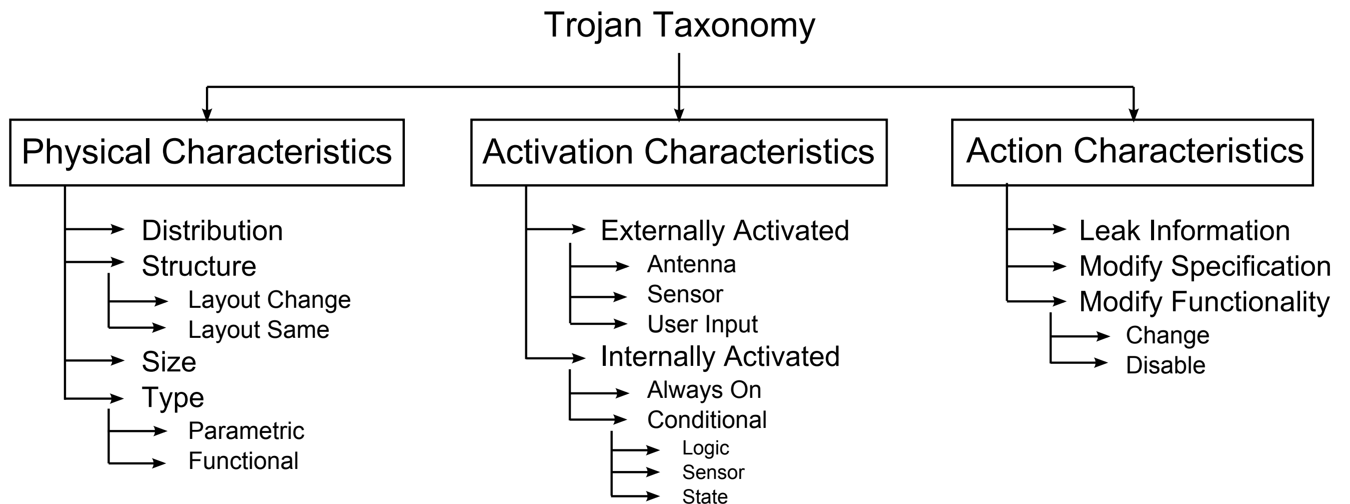


Figure 2.2: Trojan Taxonomy

Trojan insertion. The *Size* category accounts for the number of components in the chip that have been added, deleted, or compromised. *Type* partitions the Trojans into functional and parametric classes. The functional class refers to a Trojan that adds gates to or removes gates from the original design. The parametric class denotes Trojans that are realized by modifying transistor and interconnect parameters (to reduce performance and/or reliability). The nature of the changes to the physical characteristics will heavily impact our ability to detect the Trojan's presence. For example, a small Trojan which does not change the IC structure will be more challenging to detect compared to a larger Trojan that does affect the structure.

2. *Activation Characteristics* refer to the criteria that trigger Trojan attacks. First, Trojans can be triggered *externally*. For example, a rare sequence of keystrokes on a keyboard can activate a Trojan. Second, Trojans can be activated *internally*. Internal activation is further subdivided into 'always on' and

'conditional' cases. 'Always on' refers to the class of Trojans that are always active and do not require a trigger (e.g. parametric type Trojans discussed above). 'Conditionally' triggered Trojans refer to Trojans that are activated when a certain logical state is reached by the IC (as shown in Figure 2.1). Typically, an attacker will choose the activation conditions wisely so that the Trojan is rarely active, thereby preventing accidental detection during simulations and post-fabrication testing.

3. *Action Characteristics* identify the types of disruptive changes caused by the Trojan. Trojans can *leak information* such as a secret key through unused output ports or side channels. The *modify specification* class of attacks denote Trojans that modify the performance of the IC. For example, by modifying existing wire and transistor geometries in an IC's critical paths, a Trojan can degrade the IC's performance. *Modify functionality* refers to Trojans that change or bypass the original logic of an IC. For example, a Trojan can cause a Denial-of-Service (DoS) in a processor by driving its internal clock signal to a permanent logic 0 (LO) state.

Although it is possible for Trojans to be hybrids of the above classification (e.g. having multiple activation characteristics), this taxonomy captures the characteristics of Trojans and is useful for evaluating the capabilities of various detection strategies [15]. For example, the above taxonomy and others like it have been critical in developing Trojan attack models, prototypes, and benchmarks for the research community.

2.1.1.3 Real-life Prototypes and Benchmarks

While many of the Trojan actions and characteristics seem like something out of science fiction, there have been various real-life prototypes and benchmarks discussed in the literature that show just how easy it is to insert Trojans into an unsuspecting system and bypass detection schemes. For example, the Illinois Malicious Processor [16] implemented two attacks (a memory attack and a hidden shadow mode) on a general purpose processor with *only 1341 gates*. During the 2008 CSAW Embedded Systems Challenge [17], participating groups successfully implemented several Trojan-based *information leakage attacks* which evaded manual detection: leakage through RS232 protocol, thermal state, AM transmission, LED transmission, etc. [18]. Malicious Off-chip Leakage Enabled by Side channels (MOLES) [19] was a Trojan implemented in an AES cryptographic circuit *with less than 50 gates* that could leak information through power side channels. In [20], authors showed how it was possible to intentionally modify circuit parameters without detection for gradual performance degradation and early IC wear-out. Finally, the Trust-Hub team has set up a website [21] for the community to upload new Trojan benchmarks for better evaluation and comparison of different methodologies. As of this writing, there are currently 88 benchmarks available at [21].

2.1.2 Trojan Mitigation Techniques and Associated Challenges

Due to the variety of Trojans available to an attacker, detecting hardware Trojans is a very challenging problem. Deterministic validation via exhaustive tests

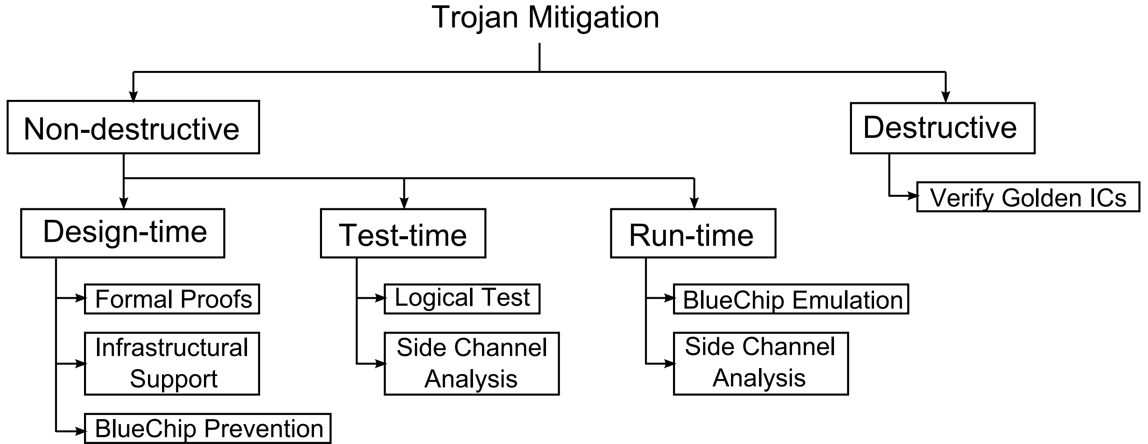


Figure 2.3: Trojan Mitigation Techniques

is simply infeasible due to the large size of modern designs. Furthermore, since Trojans are often triggered by rare events, conventional post-manufacturing tests which only target common and repeatable faults cannot be relied upon [22]. Thus, researchers have had to develop new schemes to outwit attackers. The main detection and mitigation approaches are shown in Figure 2.3 and discussed in the sections below.

2.1.2.1 Destructive Methods

Destructive techniques are reverse-engineering based approaches [23] that use Chemical Mechanical Polishing (CMP) for de-metalization and Scanning Electron Microscopes (SEMs) to extract layer-by-layer images of an IC. Image analysis follows to identify transistors, gates, and interconnects and reconstruct the manufactured IC. The reconstructed IC can be compared to the original IC design to determine if a Trojan exists. The advantage of this approach is that provided the reconstruction process is accurate, it is a foolproof method for detecting whether a Trojan exists

in the IC or not.

Unfortunately, there are several disadvantages of reverse-engineering based detection. First and foremost, they are extremely *expensive and time-consuming*, potentially taking several months to reconstruct a single IC [24]. Second, the reverse-engineering process ends up *destroying* the IC under test (ICUT), and hence cannot be applied to every IC. Finally, the results of one IC *cannot be extrapolated* to the entire manufactured lot, because an attacker might only insert a Trojan into a small subset of ICs.

Due to these disadvantages, the only use for destructive methods in the current literature is to validate a small set of "golden chips" that can be used for process calibration and comparison of side-channels with other ICUTs (see below).

2.1.2.2 Test-time Verification Methods

These approaches consist of additional tests that take place after conventional post-manufacturing testing. They operate under the assumption that the Trojan circuitry will cause unexpected changes in logical and side-channel behavior from the original design. For example, when the Trojan shown in Figure 2.1 is activated, the payload changes a signal propagating in the circuit. The trigger and payload also represent additional loads on the circuit that do not exist in the original design. The additional gates have to be driven by existing gates in the design and powered by the voltage supply, thereby causing larger delay and power consumption than in the original design.

There have been two types of approaches that exploit these properties:

- *Logic-based* approaches [25, 26, 27] develop directed test patterns that activate Trojan payloads in order to detect errors in the output. While such approaches have been shown to be effective for very small Trojans and be robust in the face of noise [22], they also have several disadvantages. First, simulation-based and functional testing both suffer from *state-space explosion*. Second, the complexity of modern designs makes it *difficult to control and observe* all internal node activity. Finally, such approaches *cannot detect parametric type Trojans* (see Section 2.1.1.2) since IC parameters are not explicitly tested. Thus, the scope and effectiveness of these approaches are limited.

- *Side Channel-based* approaches [28, 29, 30, 31] measure physical parameters of ICUTs, such as power consumption and path delay, and compares them with expected parameters of a "golden model" or "golden IC". "Golden models" can be determined through simulation tools or from verified Trojan-free ICs (see destructive methods above). Then, if a side channel of the ICUT falls beyond a threshold determined by empirical observations of the golden model/IC, the ICUT is categorized as a Trojan-infected IC.

  A major advantage over logic-based methods is that the Trojan payload *need not necessarily be activated* in order to detect the Trojan because the trigger alone will impact IC delay, power, etc. The drawback of side-channel approaches is their *vulnerability to noise* (measurement and process), which can make it challenging to detect very small Trojans [22]. Another problem with

side-channel analysis is that the *golden model/IC may not always exist.* For example, one cannot apply side-channel analysis on third party IP since the original/intended design and its behavior are unknown.

Due to the above disadvantages and the fact that there is only a limited amount of time to perform post-manufacturing tests, some Trojans may be missed by test-time methods. Thus, run-time monitoring has also been explored as an additional line of defense against the well-hidden Trojans that circumvent test-time verification.

2.1.2.3 Run-time Monitoring Methods

Run-time monitoring approaches [32, 33, 34] exploit the same properties as test-time methods (i.e. differences in logical and side-channel behavior caused by Trojans). However, the testing is performed after the IC has been deployed, which has several unique advantages. First, rare Trojan activation can be overcome by effective run-time monitoring. If run-time monitoring never stops, then the Trojan activation event itself and its (potentially) large impacts on logic and side-channel behavior can be more easily detected. Second, run-time monitors offers the flexibility to tolerate Trojan-inserted ICs. In short, if the Trojan remains inactive (i.e. not triggered) for the IC's entire lifetime, the IC will always perform its intended functions. Hence, a valid option would be to deploy the Trojan-inserted IC. If the Trojan ever does become active, the run-time monitor can respond accordingly with a defense mechanism. For example, it can disable the IC entirely to prevent the

attack or bypass the Trojan logic to maintain correct operation (e.g. [35, 36]).

The main disadvantage of run-time monitoring has been the high *resource overheads* [22] required to monitor ICs and defend against Trojan attacks. For example, the DEFENSE platform proposed in [35] has not been prototyped and would be difficult to implement in practice [6]. The path delay characterization approach proposed in [32] suffers from considerable area overhead for modern designs with millions of paths [14]. The approach in [33] adds sensors to IC power bumps for high-resolution localized current measurements, which should come with significant area and power overheads as well.

2.1.2.4 Design-time Methods

Design-time methods have been used in three ways:

1. *Trojan Prevention/Removal:* In [36], a hybrid compile-time/run-time Trojan countermeasure called *BlueChip* was developed to prevent Trojan insertion. In *BlueChip*, an Untrusted Circuit Identification (UCI) algorithm and tool-set automatically identify and remove potentially malicious circuits. Any removed hardware is replaced by logic that will trigger an exception if the removed hardware is ever activated at run-time. Low-level trusted software will then try to emulate what the missing hardware was trying to achieve. While this approach is promising, it has been shown [37] that there are malicious circuits that can *still evade* UCI detection. Furthermore, the proposed approach *can only be applied in processors* where software can emulate removed hardware.

2. *Formal Verification/Proofs:* In [38], the authors propose a new protocol where the IP consumer provides both a hardware specification and a list of "security-related properties" to the IP vendor. The IP vendor's task is two-fold: (i) to write the HDL that implements the design; (ii) to produce a formal proof that the specified HDL fulfills all the required properties. The IP consumer can then use a theorem prover to verify the properties when the IP is delivered. While this is a novel approach, it also has some shortcomings. First, it relies on a *trustworthy IP vendor* that will not add Trojans to the proof. Second, specifying the security-related properties that need to be addressed in the hardware is a *nontrivial task* for both IP consumers and vendors.

3. *Test-time Support:* There have also been several design-time strategies that aid test-time approaches. Examples include [39] and [40] which use scan flip-flops to increase the probability of Trojan activation and enhance side channel analysis. The former approach increases the probability of rarely-activated nets (i.e. places where Trojans may be triggered) during manufacturing tests. The latter approach increases circuit activity in specific regions of the IC while minimizing circuit activity in all other regions to provide better resolution for side-channel analysis.

2.1.3 Summary

Hardware Trojans have various physical, activation, and attack characteristics which can determine their effects on a computing system as well as our ability to

detect them. Due to the serious threat that hardware Trojans pose to all systems and sectors dependent on ICs, prevention and detection of hardware Trojans has garnered significant interest not only in academia, but also in governmental agencies and industry.

There have been four basic approaches to mitigate Trojans: Destructive, Design-time, Test-time, and Run-time. While the Destructive approach may be the most effective way to check the integrity and genuineness of an individual IC, the complexity, amount of time, high costs, and destructive nature limit its scope. Test-time approaches can miss out on Trojans because of the lack of observability and controllability of modern ICs, the limited amount of time available for testing, and the presence of measurement/process noise. Design-time approaches have been mostly used to aid in run-time and test-time detection of Trojans. Although run-time monitoring is flexible and can significantly improve Trojan detection/mitigation, there has yet to be a high-quality approach with low resource overhead proposed in the literature.

In Chapter 3, we propose a new comprehensive strategy for Trojan detection, which attempts to overcome many of the above issues in existing research. Our main contribution is a run-time approach with low sensing overhead that makes use of thermal sensors. Our approach is complementary to test-time approaches and monitors for Trojan activation at all times when the IC is in use. We also exploit fundamental theory to deal with measurement/process noise and detect Trojans more accurately.

2.2 Physically Unclonable Functions (PUFs)

In response to the counterfeiting and tampering attacks discussed in Chapter 1, there have been a variety of anti-cloning and anti-tampering solutions proposed in academia as well as industry. One promising solution that covers many of these attacks is the Physically Unclonable Function (PUF). PUFs are essentially an extension of biometrics towards physical objects. In the field on biometrics, random physical features such as fingerprints, which are unique to each individual and difficult to remove/duplicate, have a long history of use in human identification. Similarly, PUFs can be used to distinguish physical objects by extracting and comparing their associated random characteristics.

Silicon PUFs were first proposed by researchers at MIT in [5] as a way to identify ICs. Due to variations occurring in the manufacturing process, each IC instance of a design has slightly different physical features and performance characteristics. A silicon PUF is a special circuit embedded in an IC that extracts the IC's random characteristics to generate a unique signature, identifier, or key [41, 42, 13]. Silicon PUFs have properties that make them exceptional candidates to thwart counterfeiting and physical tampering attacks [13]. First, since many of the fabrication variations are random, the unique signature generated by the PUF cannot be cloned or replicated even by the manufacturer. Thus, in order to obtain the PUF's signature, one must have or have previously been in physical possession of the IC containing the PUF. Second, the PUF technology is tamper resistant because any attempt to physically tamper with the IC may harm the IC's physical features and modify its

associated performance characteristics. For example, if an attacker attempted to steal the PUF key through microprobing, the de-metalization and delayering steps would destroy or modify the key, thereby leaving the attacker empty-handed.

In this section, we discuss PUFs, PUF applications, and existing PUF research. The rest of the section is organized as follows:

- Section 2.2.1 discusses some PUF terminology and several of most common structures proposed as silicon PUFs in the literature: Arbiter PUF, Ring Oscillator PUF (RO-PUF), and SRAM PUF.
- As discussed above, the features of PUF signatures make them promising for many applications in hardware-assisted security. In Section 2.2.2, we highlight the applications which have been envisioned for PUFs.
- For success in hardware security applications, the PUFs and PUF-generated signatures should have three major properties: uniqueness, reliability, and unpredictability. We discuss these properties as well as the metrics which have been used in the literature to measure PUF quality in Section 2.2.3.
- The PUF properties and quality critically depend on the manufacturing variations and temporal variations experienced by the PUF/IC. Manufacturing variations are the source of PUF quality, but are often suppressed in general-purpose ICs. Temporal variations such as voltage supply noise, changes in IC temperature, and aging lead to PUF reliability issues. In Sections 2.2.4.1 and 2.2.4.2, we give a high-level overview of these variations and how they impact PUF quality.

- In Section 2.2.5, we discuss the various architectural and circuit-based approaches that have been proposed in related work to improve PUF quality. In general, these approaches occur during pre-fabrication or post-fabrication steps of the IC/PUF and do not involve modeling of the actual fabrication process or its sources of variation.
- Finally, we conclude with a summary of the section.

2.2.1 PUF Structures

In general, there are two types of silicon PUF discussed in the literature [13]:

1. *Delay-based PUFs* use race conditions to extract variations of wire and gate delays to generate PUF signatures. Examples include the Arbiter and Ring Oscillator (RO) PUFs [41] which are discussed below.
2. *Memory-based PUFs* exploit the random settling behavior of volatile memory elements to generate PUF signatures. An example is the SRAM PUF [42] which is also discussed below.

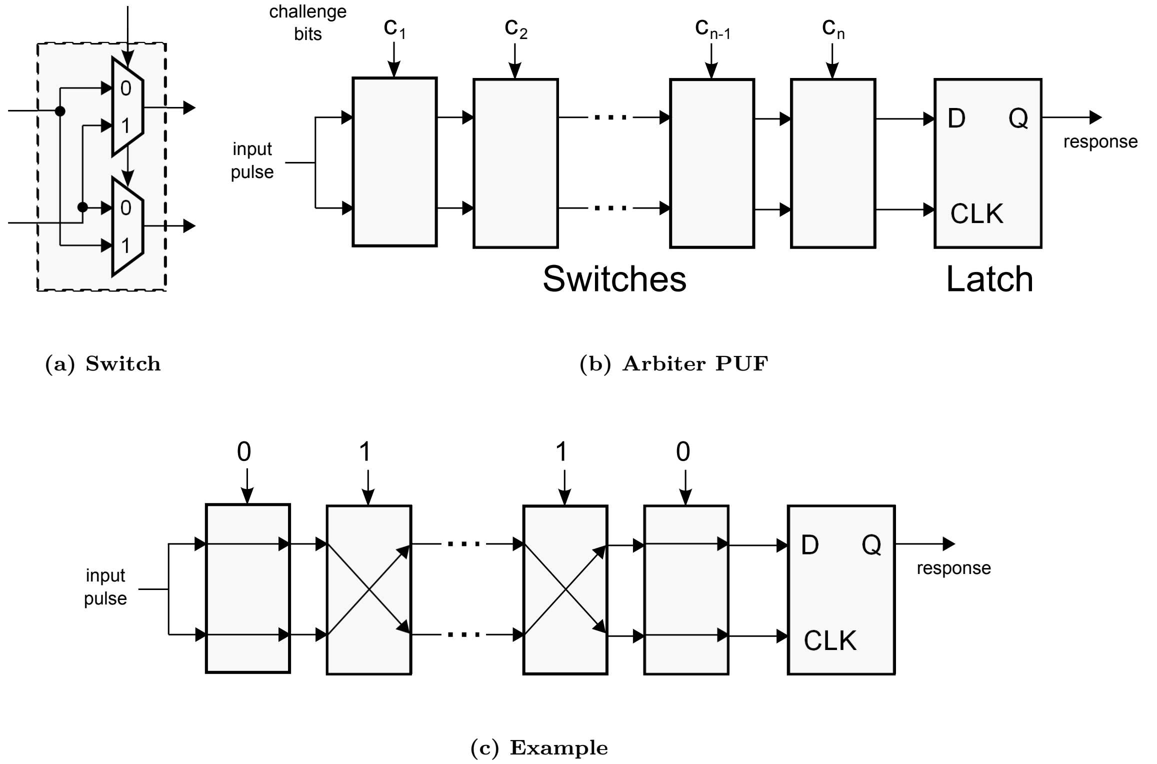
Before we discuss the operation of basic PUF structures, please make note of the following terminology. Inputs and outputs of PUF circuits are typically referred to as **challenges** and **responses** respectively. An applied challenge and its measured response is referred to as a **challenge-response pair (CRP)**. In this dissertation, we refer to all the PUF response bits as the PUF **signature**.

2.2.1.1 Arbiter PUF

The Arbiter PUF [5] was the first silicon PUF realized in an IC. The Arbiter PUF sets up two paths (designed symmetrically for same intended path delay) and uses a race condition to generates a 1-bit output (response) as follows. The two paths are simultaneously asserted with an input pulse. At the end of the paths, an “Arbiter” determines which asserted path won the race. If the pulse reaches the output of the first path faster, the Arbiter outputs a logic 1 (HI). Otherwise, it outputs a logic 0 (LO). The output/response depends on the delay present in both paths and is a function of the variations experienced during IC fabrication.

The Arbiter PUF structure is shown in Figure 2.4(b). Each path consists of a set of stages with each stage containing a switch circuit. The switch circuit is composed of two MUXES (see Figure 2.4(a)) which are controlled by a challenge bit. The challenge bit determines which paths the input signals shall take within each switch. For example, with a challenge bit set as LO, the input signals will continue to the output along their current paths. When the challenge bit is set HI, the signals will switch paths. To illustrate, the paths to a particular challenge are shown in Figure 2.4(c). Due to the variations occurring in the manufacturing process, the delays of each path within the switches will vary among ICs. Hence, the propagation time through both of the selected paths is random. The Arbiter at the end of the paths is typically implemented with a D-latch.

While the Arbiter PUF was the first PUF proposed in the literature, a robust Arbiter PUF is tough to achieve in practice. First, to generate a correct response,



(a) Switch

(b) Arbiter PUF

(c) Example

Figure 2.4: (a) Switch block constructed with MUXES controlled by challenge bit; (b) Arbiter PUF; (c) Effects of challenge bits on paths to the arbiter.

the timing difference between the two paths has to satisfy the setup time and hold time of the D-latch. Second, the routing of both paths must be perfectly symmetric which can be difficult to obtain in practice [43], especially in FPGAs. Without symmetric routing, the PUF response bits are biased towards one value (LO or HI). Finally, it is been shown that after observing a number of CRPs, simple machine-learning techniques can be used to predict PUF responses to unseen challenges with relatively high accuracy [13]. This flaw could allow attackers to determine a PUF response to a new challenge without being in possession of the IC.

2.2.1.2 Ring Oscillator PUF (RO-PUF)

The Ring Oscillator PUF (RO-PUF) is a delay-based PUF structure that is easier to implement than the Arbiter PUF. In this dissertation, we use the RO-PUF in most of our examples and experimental results.

A ring oscillator (RO) circuit consists of an odd number of inverters as shown in Figure 2.5. The oscillation frequency of an RO is determined by the total delay of its inverters. Due to process variations, the precise frequency is random and IC dependent. An RO-PUF generates signature bits by comparing oscillation frequencies of two or more ROs. A common RO-PUF architecture is shown in Figure 2.5 [41] and functions as follows. The RO-PUF contains a fixed number of ROs, which are each expected to have slightly different delay/frequency due to process variation. A challenge (input) to the RO-PUF selects two of the ROs. The frequencies of the selected ROs are compared and the response is one bit: a logic 0 (logic 1) if the upper (lower) RO has higher frequency than the lower (upper) RO.

The frequencies of the selected ROs can be obtained quite easily using standard digital components. An edge detector detects the rising edges in output oscillations and a digital counter counts the number of edges over a period of time. A comparator can be used to compare the total number of edges ($\propto$ frequencies) of the two ROs.

2.2.1.3 SRAM PUF

An SRAM cell is a circuit that stores one bit of information. A typical SRAM cell consists of cross-coupled inverters (M1,M2 and M3,M4) and access transistors

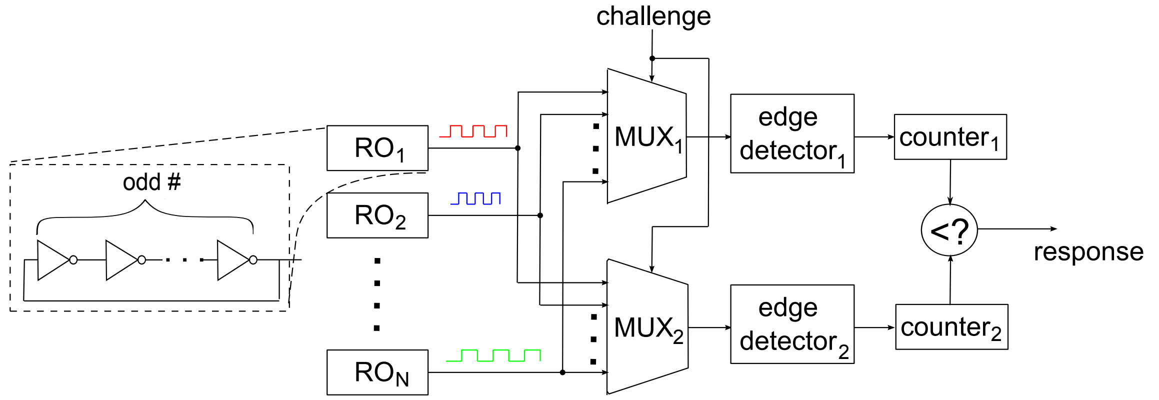


Figure 2.5: Ring Oscillator (RO) and Ring Oscillator PUF (RO-PUF)

(M5 and M6) as shown in Figure 2.6. During typical operation, the inverters drive the output nodes (labeled A and A' in Figure 2.6) to opposing logic values. The SRAM cell stores a LO when $A, A' = 0, 1$V and a HI when $A, A' = 1, 0$V. The access transistors are used to either overwrite or read the bit contained in the SRAM cell.

An SRAM cell exhibits random behavior when reset: (i) when the cell's power supply is off ($V_{\mathrm{dd}} = V_{\mathrm{gnd}}$), it enters into an unstable state where $A = A' = 0$V; (ii) when power is re-applied to the cell, it transitions from the unstable state into one of the two stable states (LO or HI). The transition to a stable state depends on the parameters (channel length, channel width, threshold voltage, etc.) of each transistor in the cell. Due to manufacturing variations, all these parameters are random and result in a tendency towards one of the stable states after power is reset. An SRAM PUF exploits the random settling behavior of a group of SRAM cells. The challenge (input) to the PUF selects a subset of the SRAM cells to power off. Response bits are the resulting logic values of the selected cells when power is re-applied.

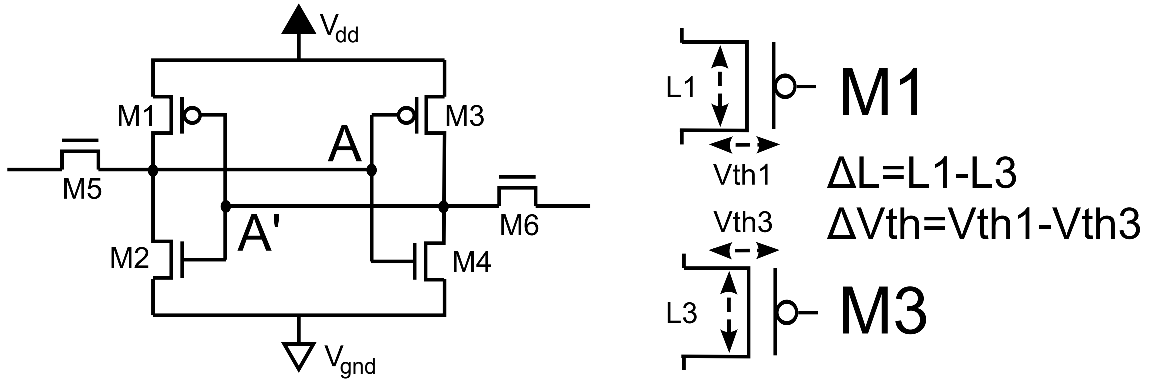


Figure 2.6: SRAM cell and parameter mismatch between M1 and M3 (ΔL, ΔV_{th})

2.2.2 PUF Applications in Hardware Security

Silicon PUFs and their associated signatures are convenient for many applications in IC security:

1. *IC Identification/Authentication:* After manufacturing a device, the vendor can record the challenge-response pairs (CRPs) of its PUF in an enrollment phase. After deployment, a device's identity can be verified at any time by the vendor by applying any challenge from the enrollment phase to the PUF. Since each PUF provides a unique response and the response can only be measured if one has the physical device, the identity of the device is verified as authentic when the response returned is the same as the response recorded during the enrollment phase. To avoid replay (eavesdropping) attacks, the selected challenge should only be used once to identify the device [13].

   The above enrollment/verification procedure can be used by IC vendors to prevent counterfeiting and overbuilding attacks (see Section 1.3)

2. *Safe Encryption Key Generation/Storage:* The safety of cryptographic algo-

rithms critically depends on the secrecy of encryption keys. Traditionally, keys are permanently stored in non-volatile memory of a device where they are susceptible to invasive attacks on memory. However, if a PUF's response to a unique challenge (or some derivative of its response) is used as an encryption key [41] , then the key is physically embedded in the device rather than stored in memory and is therefore protected against such attacks.

3. *Tamper Resistance/Evidence:* Many PUFs have a property that if their physical device is modified, their CRPs also change [13]. This property can keep the key safe (i.e. self-destruction) while also determining if a device has been tampered with in the field. By exploiting similar principles, PUF structures have also been used [44] to identify old ICs in IC recycling attacks (see Section 1.3.2).

Use of silicon PUFs in the above applications is not only been restricted to research in academia. Several companies, such as Verayo and Intrinsic-ID, are also using them [45].

2.2.3 PUF Quality and Metrics

For success in the above applications, there are three properties that are very important for PUFs [10]:

1. *Uniqueness:* In order for a PUF signature to be used as a form of identity, then for any particular challenge the difference in responses of any two PUF instances (in separate devices) should be large. A typical measure for unique-

ness is mean *inter-distance* [46]

$$d_{inter}(C) = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{\text{HD}(r_i, r_j)}{m} \times 100\% \tag{2.1}$$

where $\text{HD}(r_i, r_j)$ is the hamming distance between any two responses r_i and r_j from *different PUFs to the same challenge* C; k is the number of chips/devices in the population under test; and m is the number of bits per response. The optimal $d_{inter}(C)$ is 50%.

2. *Reliability:* The response of a particular PUF instance for the same challenge may vary due to temporal variations (see Section 2.2.4.2). However, one desires relatively stable responses so that the PUF can re-generate its key/identifier. A common measure for reliability is mean *intra-distance.* This is calculated by collecting s samples of a response at different operating conditions (supply voltage, temperature, etc.) and computing [46]

$$d_{intra}(C) = \frac{1}{s} \sum_{j=1}^{s} \frac{\text{HD}(r_i, r'_{i,j})}{m} \times 100\% \tag{2.2}$$

where r_i is the nominal response of a challenge C to a PUF; $r'_{i,j}$ is the jth sample of r_i for that *same challenge and same PUF instance*; and m is the number of bits per response. Ideally, $d_{intra}(C) = 0$ which corresponds to no changes in response for challenge C (i.e. perfect reliability).

3. *Unpredictability:* Since PUFs can be used to store secrets and cryptographic keys, PUF responses should be unpredictable/random in order to ensure that the secret remains safe from machine-learning attacks. Several measures of unpredictability have been utilized in the literature. One ad-hoc approach is

to determine how well machine learning attacks can be used to model PUF CRPs [10]. More formal metrics such as min-entropy [47] and bit-aliasing [46] measure randomness in the signatures.

2.2.4 IC Variations and Impact on PUFs

The above PUF properties are heavily influenced by the nature of the variations experienced during IC manufacturing and over the lifetime of the IC.

2.2.4.1 Manufacturing Process, Variations, and DFM

Variation in the IC fabrication process has a large impact on the yield and performance of ICs as well as PUFs. IC fabrication on a silicon wafer involves three basic steps [48]:

1. *Patterning:* Optical lithography and etching are used to create patterns on the silicon wafer. In short, light is shined through a mask (which contains the desired patterns) onto a wafer covered by a photoresist film. The portions of the resist that receive enough light are removed by development and etching processes.
2. *Semiconductor Doping:* Various regions of the silicon wafer (exposed by the above patterning step) are selectively doped with impurities. This allows the conductivity of the silicon to be changed when voltage is applied.
3. *Film Deposition:* Films of conductors and insulators are used to connect and isolate electronic devices respectively. A chemical mechanical polishing (CMP)

step is used to planarize the films and photoresist.

The above manufacturing steps have become more complex and difficult to maintain with continued semiconductor scaling. As a result, there are several sources of variability during fabrication which include CMP, optical proximity effects, lens imperfections in the lithography system, etching precision (line edge roughness or LER), and the number and position of dopants in the channel [48, 49]. These variations lead to physical differences in device structures (channel length/width, oxide thickness, etc.), differences in device electrical parameters (threshold voltage, drain-to-source current, etc.), and also variation in device performance specifications (timing, power).

Nature of Variations. Process variations have been modeled in the literature as consisting of two components [49]:

1. *Systematic.* This component is a function of the optical lithography system, the IC layout, chemical mechanical polishing (CMP), etc. and can be (mostly) predicted upfront [49]. During patterning, light passing through the mask diffracts resulting in constructive and destructive interference of light on the wafer. This is known as the proximity effect and heavily depends on the mask patterns [48]. Chemical mechanical polishing (CMP) is a function of layout/mask density and results in non-planarity in the photoresist film and light reaching the wafer out of focus (lower patterning resolution). Since the sources of systematic variation are similar for each wafer, systematic variations result in all instances of a chip having the same deterministic deviation from

design specifications [46].

2. *Random.* The major sources of random variation include doping, etching (LER), variation in oxide thickness, and material granularities [48, 49]. Since these sources are orthogonal to design implementation, they result in truly uncertain physical-parameter variations and are often modeled as random variables.

Impact of Variation on PUFs. With respect to PUF quality, recent research has shown that the systematic type of the variations are detrimental to PUF quality while the random variations are beneficial to PUFs [46, 50]. Intuitively, the systematic variations result in PUF circuity that behaves more predictably and similar among different device instances. This has the effect of making the signatures generated by the PUF more similar (less unique) and less random. Random variations, on the other hand, can enhance the signatures produced by the PUFs.

Design-for-Manufacturability. Since manufacturing variations increase the probability of yield loss, current chip fabrication methods (commonly referred to as Design-for-Manufacturability or DFM [51]) attempt to generate ICs which are immune to both systematic and random variations. *Current DFM practices are actually counterintuitive for PUFs because decreasing random variations will result in lower PUF quality.*

2.2.4.2 Temporal Variations

While ICs are designed to operate at some nominal system conditions, the conditions in which they do operate will actually change over time. Operating conditions can be due to environmental variations (voltage supply noise and thermal variations) as well as aging effects. Such changes can alter the underlying behavior of the ICs [52] and therefore affect the reliability of PUF signatures.

- *Voltage Supply Variation.* Voltage may vary due to many reasons including tolerances of the voltage regulator, IP drops along supply rails, and $\frac{dI}{dt}$ noise [52]. Since circuit speed is roughly proportional to voltage supply, variations will cause the IC to speed up and slow down. In the case of delay-based PUFs which critically depend on delay, such variations will cause PUF responses and signatures to change over time. This is undesired in many PUF applications and results in lower reliability. For example, in the RO-PUF (see Section 2.2.1.2), voltage noise can cause the oscillation frequencies of ROs being compared to swap their original sorted order, thereby changing the associated response bit in the PUF signature.

- *Thermal Variation.* Temperature across the IC is a function of the environment's temperature (where the IC is operating) and the activity levels, power consumption, etc. across the IC. Since delay, current, power consumption, etc. are strong functions of temperature, thermal variations can also causes changes in PUF signatures.

- *Variations due to Aging.* Aging effects are caused by phenomena such as neg-

ative bias temperature instability (NBTI), temperature-dependent dielectric breakdown (TDDB), hot carrier injection (HCI) and electromigration (EM), which are becoming more prominent with the continuous shrinking of ICs [53]. While voltage supply and thermal variations are transient in nature, aging causes irreversible changes (oxide wear-out and interconnect failure) in circuit components and leads to permanent shifts in IC parameters and behavior. Experimental results in [54] show that PUF reliability decreases with aging.

2.2.5 Existing PUF Research

As discussed above, PUF quality critically depends on the PUF's ability to generate signatures that are simultaneously: (i) different between IC instances (*unique* and *unpredictable/random*) and (ii) the same for a given IC instance under different operating conditions (*reliable*). In an effort to improve the uniqueness, unpredictability, and reliability of PUF signatures, researchers have investigated new architectural designs, circuit designs, and post-fabrication processing elements.

2.2.5.1 Architectures

New PUF structures and architectures have been proposed to improve PUF quality. For example, [55] and [41] proposed feed-forward and XOR-based Arbiter PUF architectures that increase randomness and nonlinearity of responses for better resistance to machine-learning attacks. Mecca PUF [56] utilizes SRAM write failures to produce PUF signatures that are more reliable and random. [46] explored

proximity-based methods for selecting ROs in order to reduce the impact of systematic variations in the RO-PUF. [57] generated ternary numbers using unreliable bits in memory-based PUFs to improve PUF signature entropy (unpredictability).

Others have proposed PUF architectures that have lower implementation overheads and better features. For instance, [58] is a hybrid delay-memory based PUF implemented with a ring of inverters that produced a larger number of CRPs compared to existing PUFs. [59] proposed a fast, low-power PUF that relied on a sense amplifier architecture. Finally, [60] proposed a new memory-based PUF called the Butterfly PUF which, unlike the SRAM PUF, does not require removal of power to generate response bits.

2.2.5.2 Circuits

Aside from architectures, several groups have proposed circuit designs that are less sensitive to systematic variations or more sensitive to random fabrication variations. For example, the authors in [61] improved the Arbiter PUF uniqueness by careful choice of sub-threshold parameters. The current starved inverter was proposed as a new element in delay-based PUFs [62] for its high sensitivity to random fabrication variations.

Other groups have examined new circuit designs that are more robust in the face of environmental variations. For example, in [63], the authors use feedback from thermal sensors to dynamically adjust voltage supplies in the RO-PUF and prevent signature bits from flipping due to thermal variations. [64] looks at two ways to im-

prove PUF reliability across thermal variations. First, they choose an optimal (fixed) supply voltage which can effectively balance out the changes in threshold voltage and mobility cause by thermal variation. Second, they use feedback polysilicon resistances to create temperature insensitive drain currents in delay-based PUFs.

2.2.5.3 Post-fabrication

There has also been a great deal of work devoted towards dealing with temporal variations and systematic fabrication variations once the IC/PUFs are fabricated.

While PUF reliability was dealt with above by reducing circuit noise levels, one can also deal with them after fabrication. In such approaches, one accepts that some level of noise will always exist in the PUF signatures and overcomes it by storing redundant helper data and employing error correction schemes. The key challenge is to make sure that no secret bits of the PUF signature/key are leaked by the helper data. For example, "fuzzy extractors" [65] have utilized helper data and hash functions to regenerate keys without revealing PUF output. "Fuzzy embedders" [66, 67] have been used to embed keys and reliably regenerate them using PUF responses. Soft information that measures the confidence of bits in the key has also been applied to enhance PUF reliability [68, 69, 70]. Finally, schemes that model the changes in delay/frequency of PUFs caused by temperature and then adjust the PUF output to compensate for the current temperature have also been investigated [71].

Recently, groups have begun investigating ways to counteract systematic varia-

tions within fabricated PUFs by adding compensation hardware to the PUF. In [72], the authors proposed a programmable delay line (PDL) that tunes the delay bias in Arbiter PUFs caused by asymmetric routing in FPGAs. The PDL is also used to determine challenges that are more robust against environmental variations for each PUF instance. Their results show that they can achieve close to ideal uniformity (randomness measure) with 14 tuning blocks added to an arbiter PUF. In [50], the authors estimate the systematic variation within an RO-PUF by solving an overdetermined system of equations using the ordinary least squares (OLS) method. The systematic biases are corrected using an entropy distiller for two challenges. Their proposed approach improves the randomness (NIST tests [73]) in PUF responses when 2nd order and 3rd order polynomials are used to estimate the systematic portion of oscillation frequencies.

2.2.6 Summary

While manufacturing variations are undesirable in general-purpose ICs and typically suppressed by Design-for-Manufacturability (DFM), silicon PUFs rely on random variations to be successful against hardware-based attacks. Thus, there has been a great deal of work geared towards improving PUF quality by enhancing uniqueness, unpredictability, and reliability of PUF signatures.

There have been three basic directions for improving PUF quality in existing work: new architectural designs, new circuit designs, and additional post-fabrication processing hardware. The architectures have tried to extract signatures with greater

randomness and less biases. Circuit level approaches have attempted to increase sensitivity to random process variations and/or reduce sensitivity to noise. Post-processing has mostly been used to correct noise in the PUF signatures from temporal variations. While all these approaches have met with success, they require larger overheads (area, power, etc.) than necessary since they have to overcome the suppression of random variations by DFM.

In Chapters 4 and 5, we propose innovative techniques [74, 75] that improve PUF quality by focusing more heavily on the source of IC variations: the manufacturing process. In doing so, our approaches can improve PUF quality at low design cost while also complimenting prior approaches.

Chapter 3

Temperature Tracking for Run-time Detection of Trojans

3.1 Introduction

3.1.1 Motivation

As discussed in Chapters 1 and 2, hardware Trojans pose a very serious threat because they have the ability to subvert both software- and hardware-based security measures. As a result, there has been a strong initiative in academia, industry, etc. to develop ways to mitigate hardware Trojan attacks. Detection approaches have been proposed at all three stages of the IC lifecycle: design-time, test-time, and run-time. The majority of existing schemes occur at test-time, but flaws in test-time approaches can allow Trojan-infected ICs to end up being deployed.

- *Inactive Trojans.* Inputs to the Trojan trigger are often chosen carefully by the attacker to prevent accidental Trojan triggering during test-time. Since there is only a limited amount of time to perform tests, the Trojan may remain inactive/dormant and hence, will not cause any changes to the IC's logical behavior.

- *Small Triggers.* While Trojan triggers have been used in the past to detect Trojan presence, very small triggers (eg. one gate [76]) can be difficult to detect because their impact on side-channels (delay, power, etc.) is negligible

and can easily be masked by process variation and measurement noise.

- *Low Impact on Side Channels.* The impact of the Trojan payload on side-channels can also be masked while the Trojan is inactive. As discussed in [76], the trigger can be used to "power gate" [52] the payload, which effectively minimizes its impact on power consumption and delay.

The above flaws have motivated others to investigate run-time approaches to compliment test-time detection. Simply put, since run-time approaches can be utilized for the *entire lifetime of the IC*, they can overcome many of the shortcomings of test-time approaches. While a Trojan-inserted IC's behavior may be indistinguishable from a Trojan-free IC's while the Trojan is inactive, at some point during run-time the Trojan will have to be activated in order to attack the IC. When a Trojan is activated at run-time, its attack may have *much larger impact* on both logical and side-channel behavior, thereby enabling easier detection.

As a motivating example, we compare power consumption of Trojan-free, Trojan-inactive, and Trojan-active ICs for a publicly available Trojan benchmark: RS232-T900 from trust-HUB [21]. RS232-T900 is a micro-UART core with a Trojan inserted in its transmitter. The Trojan is triggered by a sequence of four transmission messages and its payload prevents any further transmission.

We determined power and layout information for each of the RS232-T900 design instances by using state-of-the-art Cadence software. First, switching activity was recorded for the Trojan-free, Trojan-inactive, and Trojan-active instances by simulating each in Cadence SimVision. Next, Cadence RTL Compiler was used

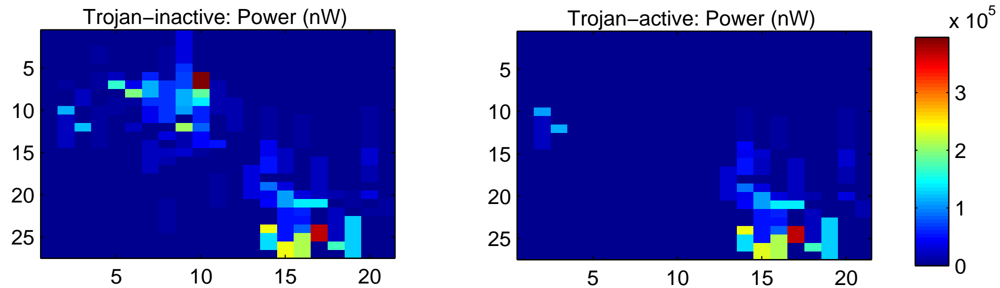


Figure 3.1: Avg. power consumption (nW) in a 250μs time window across Trojan-inactive and Trojan-active ICs for the RS232-T900 benchmark.

to convert the Trojan-free and Trojan-inserted designs to netlists. Then, with the switching activity, we obtained estimates of power consumption for the netlists of all three instances. Finally, we determined the cell layout and spatial distribution of power for each instance by performing place-and-route in Cadence Encounter.

Between the Trojan-free and Trojan-inactive cases, we found little difference in the spatial distribution of power and only a 3% difference in total power consumption. The difference in power is mainly due to the switching activity in the Trojan trigger (which does not exist in the Trojan-free design). A 3% difference is challenging to detect at test-time, especially in the presence of measurement noise and process variation. In contrast, the spatial distribution of power consumption of Trojan-inactive and Trojan-active ICs is shown in Figure 3.1. Comparing these two cases, one can see that the power differs in the upper left quadrant where the transmitter hardware is located. Once the Trojan is triggered (Trojan-active case), the Trojan payload blocks transmission. The Trojan-inactive instance, on the other hand, transmits and receives data at all times. Overall, there is a 30% decrease in total power consumption after the Trojan is triggered. Such a change should be

significantly easier to observe, thereby enabling better Trojan detection.

3.1.2 Main Contributions

The above results demonstrate that run-time monitoring of power can be very effective for detecting Trojans. Unfortunately, as previously discussed in Section 2.1.2.3, such approaches have very large resource overheads. Thus, their use in practical applications has been limited.

In this chapter, we propose an innovative low-overhead solution for Trojan detection at run-time. Our approach explores a new side channel for Trojan detection: temperature[1]. Simply put, Trojans can cause significant changes in power consumption after activation which will *also be reflected in the IC's thermal profile*. In contrast to prior run-time approaches, temperature-based Trojan detection has much lower sensing overhead because many electronic systems are already equipped with thermal sensors (e.g. AMD Operton processor has 38 sensors). Thermal sensors are heavily utilized for dynamic thermal management (DTM) to prevent IC reliability issues and excessive power consumption. By exploiting these existing sensors, we can amortize the cost of detecting Trojans at run-time.

Our main contributions are summarized as follows:

- We propose a comprehensive framework for temperature-based Trojan detection which consists of design-time, test-time, and run-time phases. In the

[1]As of this writing, there is only one other paper [77] that uses temperature to detect Trojans. Their work was developed in parallel with ours and, in contrast to ours, is a test-time Trojan detection approach.

design phase, we statistically characterize an IC's power/thermal dynamics and optimally place thermal sensors. The test-time phase is used to calibrate each IC to account for fabrication variation. The run-time phase integrates the information from the previous phases with thermal sensor measurements to detect Trojan activation.

- We propose two mechanisms to detect Trojan activation during run-time. The first is a local sensor-based approach that uses information from thermal sensors, statistical information provided by the test phase, and hypothesis testing. The second is a global approach that exploits correlation between sensors and maintains track of the IC's thermal profile using a Kalman filter (KF). Deviations from the "expected" thermal profile are used to detect the presence of Trojans.

- We test our detection mechanisms on five publicly available Trojan benchmarks (from [21]) and use state-of-the-art Cadence simulation tools to compute power/thermal profiles. In all but one benchmark, the proposed approaches are capable of detecting Trojan activation quickly (on the order of milliseconds) and with very few false-positives.

The remainder of the chapter is organized as follows. In Section 3.2, we discuss how thermal sensors and predictive models have been used in prior work for dynamic thermal management and to track temperature. Our specific problem and its challenges are clearly defined in Section 3.3. The proposed framework and detection mechanisms are discussed in Section 3.4. Experimental results are discussed

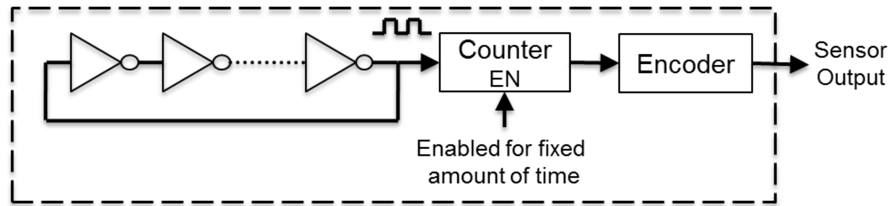


Figure 3.2: Basic thermal sensor circuit

in Section 3.5. We summarize the merits of techniques proposed in this chapter in the last section.

3.2 Preliminary

3.2.1 Thermal Sensors

Thermal sensors are heavily utilized for dynamic thermal management (DTM) to prevent IC reliability issues [78] and excessive power consumption [79, 80]. For example, the AMD Opteron multicore processor is equipped with 38 thermal sensors [81]. A basic thermal sensor, consisting of a ring oscillator (RO), counter, and encoder, is shown in Figure 3.2 [82] and functions as follows. The oscillation frequency of the RO is a function of temperature. The counter measures the oscillation frequency by counting the output pulses from the RO over a fixed period of time. The encoder then converts the count to a temperature value. Since it consists of basic digital components, this thermal sensor can easily be added to processor, ASIC, and FPGA designs.

Basic DTM approaches are reactive in nature. If the temperature of a sensor exceeds a predefined threshold (typically around 80°C), DTM will trigger one or

more mechanisms to cool the IC at the sensor's location. While reactive approaches can be effective, they also have shortcomings:

- *Limited # of Sensors.* Since thermal sensors take up space in the design and consume power, they cannot be placed everywhere leaving some locations less observable. Hence, thermal hotspots at locations without sensors can be missed.

- *Measurement Noise.* Since the thermal sensor is a circuit, it is susceptible to noise in the voltage supply (see Section 2.2.4.2). This noise will have an impact on the RO's oscillation frequency and will cause the sensor's temperature measurement to be noisy. With the thermal sensor being used to trigger cooling, the noise may cause certain hotspots to be missed or cause the DTM to overreact.

To overcome the above issues, recent DTM research [83, 84] has tried to combine the sensor measurements with predictive thermal models such as the RC thermal model which is discussed below.

3.2.2 RC Thermal Model

Temperature is a strong function of power consumption. Hence, if one knows the power profile of chip, the IC's thermal profile can be estimated. One popular method for predicting temperature in academia is the RC thermal model [80].

In the RC model, the IC is divided into grids and the temperature and power consumption of each grid at time t are represented as constants (see Figure 3.3). A

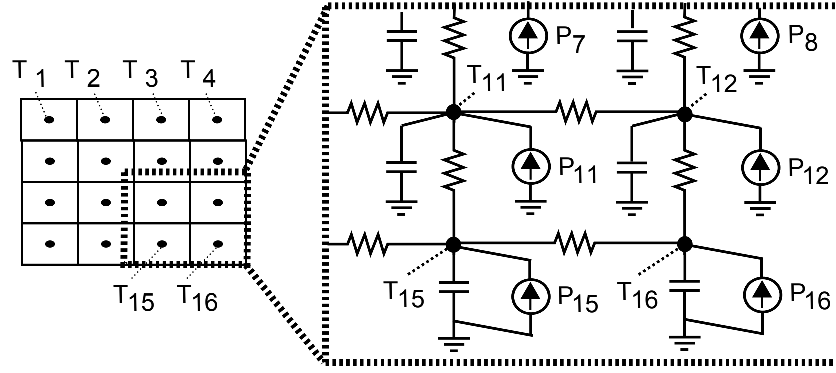


Figure 3.3: IC broken into grids and RC thermal model within dotted region

circuit is used to estimate the IC's thermal profile where node voltage and circuit current are analogous to temperature and power/heat flow in each grid. Voltage ground is analogous to the environment's ambient temperature. Thermal capacitance and thermal resistance between neighboring nodes determine how heat flows between nodes/grids of the IC. Temperature for the entire IC can be determined by solving the following set of ODEs:

$$\sum_{\forall j \in N_i} \tfrac{1}{R_{ij}}(T_i(t) - T_j(t)) + C_i \tfrac{dT_i(t)}{dt} - P_i(t) = 0 \quad \forall i \tag{3.1}$$

where $T_i(t)$ and $P_i(t)$ are the temperature and power dissipated at node i and time t; C_i denotes thermal capacitance at node i; R_{ij} denotes thermal resistance between nodes i and j; and N_i is the set of all neighbors for node i. The above equations are often written in a discrete matrix form [85]:

$$\vec{T}[k] = \mathbf{A}\vec{T}[k-1] + \mathbf{B}\vec{P}[k-1] \tag{3.2}$$

where $\vec{T}[k]$ and $\vec{P}[k]$ are temperature and power vectors (each element corresponds to one node/grid) at discrete timestep k; $\mathbf{A}$ and $\mathbf{B}$ are coefficient matrices that depend upon the RC circuit and timestep duration.

By combining the predictive RC thermal model with sensor measurements, the issues above associated with reactive DTM (limited number of sensors, measurement noise) can be overcome. For instance, recent DTM approaches [83, 84] have used a Kalman Filter (KF) to merge prior knowledge of power consumption and statistical noise characteristics with run-time sensor measurements to obtain *optimal thermal profile estimates of the entire IC over time.*

In this chapter, we exploit the key features of thermal sensors, the RC thermal model, and the Kalman Filter (KF) to detect unexpected changes in IC power/temperature caused by active Trojans. Temperature-based Trojan detection is a relatively unexplored avenue which should have lower overheads compared to previous run-time approaches.

3.3 Problem Definition and Challenges

Our problem is inspired by the example discussed in Section 3.1.1 (RS232-T900). We assume that there are three possible states that an electronic system or IC can be in: *Trojan-free*, *Trojan-inactive*, and *Trojan-active*. Each state is defined by a set of statistical characteristics $\mathbf{S}_f$, $\mathbf{S}_i$, $\mathbf{S}_a$ respectively. The Trojan-free and Trojan-inactive characteristics, while not necessarily identical, are close enough such that the Trojan-inserted IC can evade test-time Trojan detection methods (i.e. $\mathbf{S}_f \approx \mathbf{S}_i$). The Trojan-active characteristics $\mathbf{S}_a$ on the other hand differ significantly from the other two. Our goal is a run-time temperature-based approach that can detect changes from $\mathbf{S}_f$ and $\mathbf{S}_i$ to $\mathbf{S}_a$ after the Trojan is activated. Note, we are not

concerned with Trojan-inactive ICs since, prior to Trojan activation, they essentially provide the same functionality as Trojan-free ICs. Stated formally, our problem is as follows:

Given two hypotheses of the system's state:

$$\begin{cases} \mathcal{H}_0 & \textit{The state is Trojan-free or Trojan-inactive} \\ \mathcal{H}_1 & \textit{The state is Trojan-active} \end{cases}$$

Use thermal sensor observations to determine if the IC's state (characteristics) correspond to $\mathcal{H}_0$ *(*$\mathbf{S}_f$*,* $\mathbf{S}_i$*) or* $\mathcal{H}_1$ *(*$\mathbf{S}_a$*).*

The above problem has various challenges to overcome some of which are specific to temperature tracking and some of which are common to Trojan detection:

- *Golden IC/model.* Most Trojan detection approaches rely on the existence of a "golden model" to distinguish Trojan-free and Trojan-inserted ICs. In our case, we assume that the Trojan-free design is given and from it we can compute $\mathbf{S}_f$ characteristics to function as our golden model.

- *Autonomous detection.* Since the $\mathbf{S}_f$ characteristics are known and $\mathbf{S}_f \approx \mathbf{S}_i$, we should be able to easily track temperature for Trojan-free and Trojan-inactive designs as in prior work. The challenge is detecting active Trojan ICs because the $\mathbf{S}_a$ characteristics are unknown. We propose two mechanisms for detecting Trojan activation at run-time.

- *Sensor Infrastructure and Noise.* Prior work has shown that sensor placement, number of sensors, sensor noise, etc. have a profound impact on temperature

tracking [82]. In this chapter, we vary the number of sensors to see the impact on temperature-based Trojan detection. One of our approaches uses the Kalman Filter which explicitly accounts for measurement noise.

- *Fabrication Variation (FV).* FV makes it more challenging to track temperature as well as detect Trojans. For tracking, FV results in larger uncertainty in the estimated thermal profile [84]. For Trojan detection, FV makes it difficult to distinguish between deviations in power/temperature due to manufacturing and Trojan presence [31]. In our approach, calibration is performed for each IC to ensure robustness in the face of FV.

3.4 Temperature-Based Detection

In this section, we discuss the overall framework and algorithms for our temperature-based Trojan detection. While detection itself occurs at run-time, the approach itself is comprehensive in nature with each phase of the IC supply chain/process playing a critical role. Namely, offline profiling steps at design-time and test-time are used to deal with many of the challenges discussed in the previous section. An overview of entire approach with design, test, and run-time phases is shown in Figure 3.4. The details of each phase are discussed below.

Assumptions. Before we begin, please make note of the following assumptions. As discussed above, side channel-based Trojan detection often relies on a “golden” model or IC. Our approach assumes that we have access to the Trojan-free design or prototype from which we can obtain statistical characteristics ($\mathbf{S}_f$) of switching

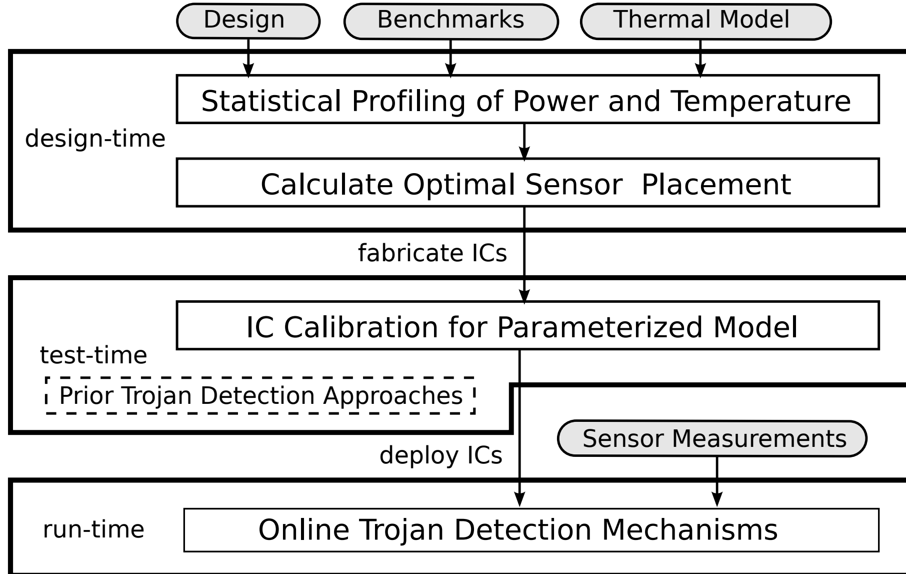


Figure 3.4: Phases of the Proposed Approach

activity, power consumption, thermal dynamics, etc. For simplicity, we shall also assume that the statistical characteristics are Gaussian. In general, this a valid assumption due to the central limit theorem (CLT) which states that the statistical characteristics of a sufficiently large number of independent random variables will be approximately normally distributed. *Note, however, that our detection approaches are very general and can be extended to deal with other statistical characteristics as well.*

3.4.1 Design Phase

Design Profiling. We profile the Trojan-free designs using the RC thermal model (see Section 3.2.2), benchmarks, and either state-of-the-art simulation tools or prototype ICs. The RC thermal model divides the power consumed by the design into grids and uses a vector $\vec{P}$ to represent power consumed in the grids. The

statistical approaches for temperature tracking and Trojan detection used in this chapter require probability distribution functions (PDFs) to summarize the design's expected power and/or temperature. Benchmarks that are representative of the design's expected workload along with simulation tools are used to estimate the PDFs. Alternatively, IC prototypes that are verified as Trojan-free can be used. We approximate the PDFs as Gaussian with mean vectors $\vec{\mu}_p$ and $\vec{\mu}_T$ and covariance matrices $\mathbf{Q}_p$ and $\mathbf{Q}_T$ (for power and temperature respectively).

Sensor Placement. We adopt the sensor placement approach from [82] which uses the temperature covariance matrix $\mathbf{Q}_T$. Specifically, sensors are placed in a greedy fashion to minimize the following cost function

$$cost = \sum_{\forall grids:i} \max\left(0, 1 - \sum_{\forall sensors:j} q_{i,j}\right) \tag{3.3}$$

where $q_{i,j}$ denotes the element at location i and j of matrix $\mathbf{Q}_T$ (i.e. correlation in temperature between IC grids i and j). The cost captures how much information is provided by sensors in selected locations and encourages sensor placement at locations that have high correlation with other locations and are not yet covered by a sensor. It has been shown that using this approach improves thermal estimation accuracy [82].

3.4.2 Test Phase

When the design phase is complete, we fabricate the ICs. At this point, we assume that some Trojan-inserted ICs are fabricated and inserted into the supply chain.

IC Parameter Calibration. Fabrication variation (FV) results in ICs that have different physical, electrical, and performance parameters from the nominal design. As discussed above, FV makes it more challenging to accurately detect Trojans and track temperature. To ensure robustness, we must have accurate power/thermal statistics ($\vec{\mu}_p$, $\vec{\mu}_T$, $\mathbf{Q}_p$, $\mathbf{Q}_T$) for each IC under test (ICUT). One can accomplish this by applying test vectors to the ICUT, measuring power consumption, and estimating the PDFs after fabrication. For example, gate-level characterization has been applied in prior work [31] to successfully profile IC gate parameters. Temperature-based approaches which utilize infrared cameras [77] and Expectation Maximization (EM) [86] are also applicable.

Test-time Detection. One can argue that our temperature-based approach may not be able to detect all types of Trojans. For example, Trojans that are only active for a few clock cycles may not have a large impact on power and/or temperature. This is why we emphasize an integrated framework as shown in Figure 3.4. Prior test-time detection schemes are used during this phase to remove Trojan-inserted ICs (that might be missed by our approach) before they are deployed. The proposed run-time approach would then complement test-time approaches by detecting the Trojan-inserted ICs (that are missed by test-time schemes) as they are activated in the field.

3.4.3 Run-time Phase

As discussed in Section 3.3, our main problem is to decide the correct hypothesis (state of the system): $\mathcal{H}_0$ or $\mathcal{H}_1$. In other words, is the IC Trojan-free/Trojan-inactive or Trojan-active? In this section, we propose two mechanisms to solve the problem. The first is a local sensor-based approach that uses an hypothesis testing (HT) framework and Bayesian decision theory. The second is a global approach that exploits correlation between sensors and maintains track of the IC's thermal profile with a Kalman filter (KF).

3.4.3.1 Local Sensor Approach

For simplicity, let us suppose we have one sensor measurement at timestep k denoted by $S[k]$ from which we shall decide the state (we'll consider more sensors later). In an hypothesis testing (HT) framework, one assumes that $S[k]$ can only come from one of two PDFs: S_0 or S_1 which correspond to null and alternative hypotheses respectively.

In our case, the null and alternative hypotheses correspond to the IC thermal state in Trojan-free/Trojan-inactive ($\mathcal{H}_0$) and Trojan-active ICs ($\mathcal{H}_1$) respectively. We shall choose the correct state as the one with the highest probability of occurrence given $S[k]$ (i.e. $\operatorname{argmax} P(H_x \big| S[k]), x \in \{0,1\}$). By applying Bayesian decision theory, it can be shown the optimal decision is [87]:

Choose $\mathcal{H}_1$ (Trojan-active state) when:

$$\frac{p(S[k] \big| \mathcal{H}_1)}{p(S[k] \big| \mathcal{H}_0)} > \frac{P(\mathcal{H}_0)}{P(\mathcal{H}_1)} \tag{3.4}$$

Otherwise, choose $\mathcal{H}_0$ (Trojan-inactive state).

where $P(x)$ and $p(x|y)$ denote the prior probability of x and probability of x given y respectively.

While the above decision rule is theoretically sound, it is difficult to directly apply to our problem for two reasons.

- One cannot assume that all measurements come from single stationary PDF (i.e. one that is time invariant) since the IC temperature varies with time.
- Even if the PDFs were stationary, we do not have access to the Trojan-active design and therefore cannot accurately estimate $p(S[k]|\mathcal{H}_1)$ or $P(\mathcal{H}_1)$.

We get around these issues by making the following simplifying assumptions.

Stable State Temperature. The first issue no longer presents a problem when the IC's temperature has reached a stable state. Put simply, if an IC's power consumption is similar for a long period of time, the IC's temperature will end up converging to a "stable state" [88] where measurements of its thermal state are actually samples from a stationary PDF. In our approach, we assume that we have a verified Trojan-free design/IC (see Section 2.1.2.1) and benchmarks that can characterize the IC's activity. We can then run the benchmarks on the verified Trojan-free design/IC until reaching the stable state. From there, we take measurements and approximate $p(S[k]|\mathcal{H}_0)$, as Gaussian with mean μ_0 and variance σ_0^2.

Trojan PDF Estimate: To overcome the second issue, we exploit the fact that S_0 and S_1 must be slightly different and assume a simple Trojan attack model. Specifically, we assume that the mean of $p(S[k]|\mathcal{H}_1)$ (μ_1) differs from $p(S[k]|\mathcal{H}_0)$'s

known mean (μ_0) by some fixed percentage difference $S_{\%}$ and both possess the same variance ($\sigma_0 = \sigma_1$.) If $S_{\%} < 0$ ($S_{\%} > 0$), Trojan activation causes the IC to lose (gain) some functionality. While this assumption is imperfect, it's simply the best we can do to apply the theory since we have little if any knowledge of the actual Trojan-active design/attack. Moreover, it does allow us to make optimal guarantees for Trojan detection. If the assumed statistics hold, then we can optimally detect the active Trojan. Furthermore, if the $\mathcal{H}_1$ statistics differ from the $\mathcal{H}_0$ statistics by more than $S_{\%}$, we can also probably detect the Trojan.

Single sensor Decision Rule. With the above assumptions, we now have enough information to apply the optimal decision rule in Eqn. (3.4). For simplicity, we assume that $P(\mathcal{H}_0) = P(\mathcal{H}_1) = .5$ and express the conditional probabilities with

$$p(S[k]\big|\mathcal{H}_x) = \frac{1}{\sigma_x\sqrt{2\pi}} exp\left(-\frac{(S[k]-\mu_x)^2}{2\sigma_x^2}\right), \quad x \in \{0,1\} \tag{3.5}$$

With the above, one can easily solve Eqn. (3.4) w.r.t. thermal measurement $S[k]$ to determine the test statistic [87] (i.e. a hypothesis test specified in terms of temperature)

Choose $\mathcal{H}_1$ (Trojan-active state) when:

$$S[k] \in R_1 \tag{3.6}$$

Otherwise, choose $\mathcal{H}_0$ (Trojan-inactive state).

In the above equation, R_1 defines a set of temperature values belonging to the Trojan-active state $\mathcal{H}_1$. If the sensor measurement $S[k]$ is from R_1, then the IC's state is most likely $\mathcal{H}_1$. Otherwise, the state is most likely $\mathcal{H}_0$. Note that for the

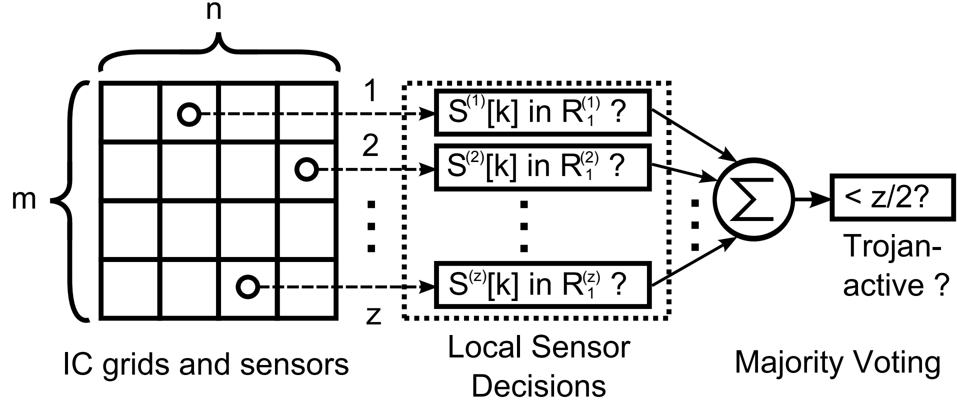


Figure 3.5: Local Hypothesis Testing (HT) Approach with z sensors and $m \times n$ grid

single mode Gaussian PDFs assumed in this chapter, Eqn. (3.6) can be simplified to either $S[k] < r_1$ or $S[k] > r_1$ (depending on the sign of $S_{\%}$) where r_1 is a threshold value. *Nevertheless, the theory and associated decision rule are general enough to handle other PDFs as well.*

Multi-sensor Decision Rule. For multiple sensors, we can easily extend the above rule by collecting the sensor measurements as a vector $\vec{S}[k]$ and using multivariate Gaussian PDFs. For simplicity, we take a simple ad-hoc approach instead. We evaluate Eqn. (3.6) for each sensor and come to decision ($\mathcal{H}_0$ or $\mathcal{H}_1$) by majority voting.

Overheads. A high-level overview of the local HT approach and its overheads is shown in Figure 3.5. HT's offline overhead includes computing the optimal test-statistics (R_1) for each sensor. At run-time, a simple circuit determines if the sensor measurement is in the sensor's corresponding R_1 and outputs a 1-bit vote. *Note that for the single mode Gaussian PDFs assumed in this paper, the circuit one needs to implement is a simple comparator.* Majority voting is used to combine the decisions

of z sensors and obtain a final decision. All these operations are simple and the overall complexity *depends only on the number of sensors z.*

3.4.3.2 Global (Kalman Filter-based) Approach

Our second approach is a global approach that exploits correlation between sensors and uses a Kalman Filter (KF) to dynamically track the system's thermal profile at run-time. An autocorrelation based metric then decides between hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$ (Trojan-free/Trojan-inactive and Trojan-active).

Temperature Tracking Via Kalman Filter. We track IC temperature at run-time using the standard Kalman filtering (KF) approach developed in prior work [83, 84]. For simplicity, we only consider dynamic power, but leakage power can be handled as well [89]. The KF relies on a state-space equation to model the random dynamics of the state being estimated and on a measurement equation to relate measurements with the state being estimated. The state-space equation for temperature tracking is the discrete form RC thermal model equation discussed in Section 3.2.2 (copied below for convenience)

$$\vec{T}[k] = \mathbf{A}\vec{T}[k-1] + \mathbf{B}\vec{P}[k-1] \tag{3.7}$$

The above equation assumes that the current thermal state $\vec{T}[k]$ depends on the previous thermal state $\vec{T}[k-1]$ (Markovian assumption) and also local power dissipation $\vec{P}[k-1]$. Due to variations in the voltage supply noise, system workload, etc., the power $\vec{P}$ is random at each timestep and $\vec{T}[k]$ cannot be precisely computed with the state-space model alone. To improve the estimate, the KF uses measurements

collected by thermal sensors and the following measurement model

$$\vec{S}[k] = \mathbf{H}\vec{T}[k] + \vec{v}[k] \tag{3.8}$$

where $\vec{S}[k]$ is a vector of sensor measurements at timestep k; $\mathbf{H}$ is a transformation matrix based on the sensor placement; and $\vec{v}[k]$ is a Gaussian random vector with zero mean and known covariance $\mathbf{R}$ [83] representing measurement noise.

The KF estimates the thermal state of a chip as follows. $\vec{P}[k]$ is modeled as a Gaussian random vector[2] with known mean $\vec{\mu}_p$ and covariance $\mathbf{Q}_p$ (which we determine in the design/test phases). KF estimation is then performed recursively with *predict* and *update* steps. In the *predict* step, the KF uses $\vec{\mu}_p$ and the previous temperature estimate to predict the IC's new thermal state. In the *update* step, the KF corrects this estimate based on new sensor measurements. This predict-update

[2]Note that while Gaussian distributions are assumed for power/temperature by the KF in this paper, prior work [85, 86] has shown that the KF framework can be extended to handle a mixture of Gaussians (MOGs) for more general PDFs. Such approaches can easily account for multiple power profiles, modes of IC operation, etc. For simplicity, we assume a single Gaussian PDF in this paper, but shall evaluate MOGs for Trojan detection in future work.

process is shown in Figure 3.6 and described by the following equations:

$$\textit{predict:} \qquad \vec{T}[k|k-1] = \mathbf{A}\vec{T}[k|k-1] + \mathbf{B}\vec{\mu}_p \tag{3.9}$$

$$\mathbf{C}[k|k-1] = \mathbf{A}\mathbf{C}[k|k-1]\mathbf{A}^T + \mathbf{B}\mathbf{Q}_p\mathbf{B}^T \tag{3.10}$$

$$\textit{update:} \qquad \vec{e}[k] = \vec{S}[k] - \mathbf{H}\vec{T}[k\big|k-1] \tag{3.11}$$

$$\vec{T}[k|k] = \vec{T}[k|k-1] + \mathbf{K}[k]\vec{e}[k] \tag{3.12}$$

$$\mathbf{K}[k] = \mathbf{C}[k|k-1]\mathbf{H}^T(\mathbf{R} + \mathbf{H}\mathbf{C}[k|k-1]\mathbf{H}^T)^{-1} \tag{3.13}$$

$$\mathbf{C}[k|k] = (\mathbf{I} - \mathbf{K}[k]\mathbf{H})\mathbf{C}[k|k-1] \tag{3.14}$$

$\vec{T}[k|k]$ and $\vec{T}[k|k-1]$ are estimates of the temperature at timestep k computed with and without sensor information respectively; $\mathbf{C}[k|k-1]$ and $\mathbf{C}[k|k]$ are the error covariance matrices associated with $\vec{T}[k|k-1]$ and $\vec{T}[k|k]$; $\vec{e}[k]$ is the KF *residual* which reflects the discrepancy between the predicted and actual measurements; $\mathbf{I}$ is the identity matrix; $\mathbf{K}[k]$ represents the Kalman gain at the kth step and is chosen to minimize the error in $\vec{T}[k|k]$. First, Eqn. (3.9) is used to compute a prediction of the temperature ($\vec{T}[k|k-1]$). Then, Eqn. (3.12) updates the prediction ($\vec{T}[k\big|k-1] \rightarrow \vec{T}[k\big|k]$) based on the residual (Eqn. (3.11)). The filter also generates error covariance matrices associated with $\vec{T}[k|k-1]$ and $\vec{T}[k|k]$ which keep track of the error in the thermal estimates and are computed based on the power and sensor noise covariance matrices ($\mathbf{Q}_p$ and $\mathbf{R}$).

Steady State Kalman Filter. When the statistical characteristics of $\vec{P}$ and measurement noise are fixed (or do not change for a relatively long time), the KF stabilizes which means $\mathbf{C}[k|k-1], \mathbf{C}[k|k]$, and $\mathbf{K}[k]$ converge to static values. This is referred to as the KF *steady state* During the steady state, even though the

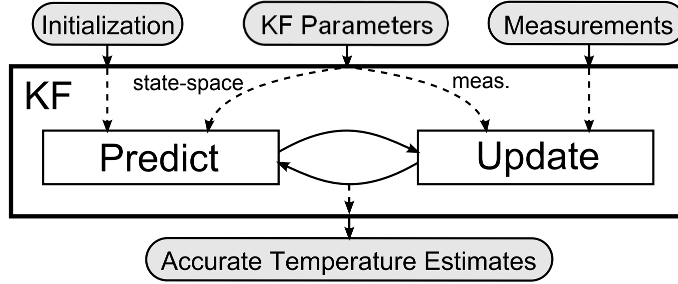


Figure 3.6: KF-based Temperature Tracking

temperature may change with time, the error associated with the estimates remains the same. The steady state allows one to create low overhead implementations of the KF [84] by replacing $\mathbf{C}[k|k-1], \mathbf{C}[k|k]$, and $\mathbf{K}[k]$ in Eqns. (3.9) to (3.14) by constants.

Autocorrelation-based Detection Rule. While the KF can be used to accurately track temperature, we also need a rule to decide on the correct state ($\mathcal{H}_0$ or $\mathcal{H}_1$). Our decision rule is based on the KF residual and uses the autocorrelation function of the residual process. In the KF, residual $\vec{e}[k]$ represents the discrepancy between the predicated temperature and thermal sensor measurements. If $\vec{e}[k]$ is small (large), the two agree (disagree) on the thermal state. Assuming the state-space model/parameters and the sensor noise covariance are reasonably accurate, the autocorrelation of the residual should be close to zero on average [90]. When a Trojan gets activated, the state-space model (*which does not account for the power of an active Trojan*) becomes less accurate and should cause the autocorrelation to diverge from zero.

We use the following method to detect a Trojan at timestep x. We record the residual in the N previous timesteps of the KF ($\vec{e}[x-N], \vec{e}[x-N+1], \ldots, \vec{e}[x]$) and

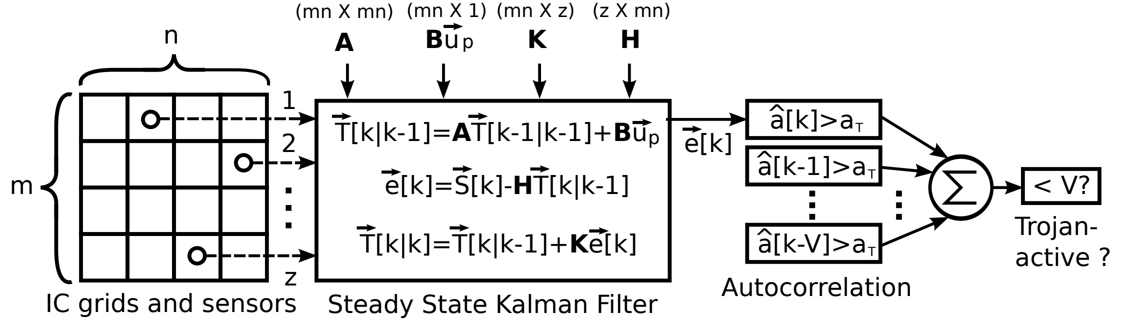


Figure 3.7: Global Kalman Filter (KF) Approach with z sensors and $m \times n$ grid

then compute the following cost function

$$\hat{a}[x] = \frac{1}{N} \sum_{i=x-N+1}^{x} \left(\vec{e}[i] \cdot \vec{e}[i-1]^T \right) \tag{3.15}$$

This cost is the average autocorrelation of the residual process in the last N timesteps. To decide if a Trojan is activated, we define thresholds a_T and V. If $(|\hat{a}[x]| > a_T)$ for more than V consecutive timesteps, we assume a Trojan has been activated (i.e. state $\mathcal{H}_1$). a_T and V are parameters that tune the aggressiveness of the decision rule.

Overheads. A high-level overview of the KF approach and its overheads is shown in Figure 3.7. The main offline overheads are estimating the power statistics ($\vec{\mu}_p$ and $\mathbf{Q}_p$) and computing the steady state Kalman gain matrix $\mathbf{K}$. During run-time, the KF performs some matrix-vector multiplications and vector additions/subtractions (Eqns. (3.9), (3.11), (3.12)) at each timestep k. The size of the matrices/vectors and complexity of these operations *depend on the number of grids in the RC thermal model (mn) and the number of sensors (z).* Running averages of the autocorrelation $\hat{a}$ for the last V timesteps must be stored and the final state is chosen based on $V + 1$ thresholding operations.

3.4.3.3 Qualitative Comparison

The salient differences between the above two detection mechanisms are as follows:

- *Stable State Assumption:* The local hypothesis testing (HT) approach requires the system under test to be in a stable thermal state so that the sensor measurements can be compared with stationary PDFs. The global Kalman Filter (KF) approach works with the system in any thermal state since it tracks the system's thermal profile at all timesteps.

- *Sensor Correlation:* The local HT approach compares each sensor measurement with its corresponding stable state PDF in an independent fashion. Correlation between the sensors is not exploited and the final decision is made based on a majority vote. The global KF approach exploits the correlation between sensors to accurately track temperature and detect Trojans.

- *Run-time Overheads:* The KF approach clearly has larger run-time overheads than the HT approach (see Figs. 3.5 and 3.7). While the HT approach primarily works with scalar values and computes 1-bit decisions, the KF approach requires matrix-vector storage and computations which are more expensive ($O(mn)$).

3.5 Experiments and Discussion

3.5.1 Setup

Benchmarks. We tested our Trojan detection schemes on five publicly available Trojan benchmarks (from trust-HUB [21]):

1. *RS232-T900* was discussed in Section 3.1.1.

2. *s38417-T300* contains a Trojan trigger with activation probability of 1.7e-44. Once activated, the payload leaks the value of a specific net through a 29 stage ring oscillator.

3. *BasicRSA-T200* is an RSA encoder with a Trojan triggered by a specific plaintext input. The payload permanently disables encoding of the plaintext.

4. *MC8051-T300* is an implementation of the 8051 microprocessor with a Trojan. The Trojan is triggered when a specific string is sent through the UART and the payload blocks new messages from the UART.

5. *MC8051-T600* is another implementation of the 8051 microprocessor with a Trojan. The Trojan is activated by an external interrupt and disables 8051 instructions containing jumps.

We determined power and layout information in the above benchmarks by simulating, synthesizing, and placing each design with Cadence SimVision, RTL Compiler, and Encounter tools for two different testbench instances: one which activates the Trojan (i.e. Trojan-active) and one which does not (i.e. Trojan-inactive). The

Benchmark	RS232-T900	s38417-T300	BasicRSA-T200	MC8051-T300	MC8051-T600
% difference	-39.97%	54.33%	-28.40%	-1.5%	-72.16%

Table 3.1: % difference in total power consumption between Trojan-inactive and Trojan-active in 250ms experiment

difference in power consumption between the two is shown in Table 3.1 for all the benchmarks. In all cases but one (MC8051-T300) there is a % difference larger than 25%.

Temperature-based Trojan Detection. We divided the IC into 20 by 16 grids (320 distinct regions). In the design phase, we computed power and temperature statistics using 250ms of data generated by Cadence (from the Trojan-free designs) and the RC thermal model [80]. With the resulting statistics, we placed sensors as discussed in Section 3.4.1. The number of sensors we tested were 4, 16, and 32. For simplicity, we ignored fabrication variation and therefore did not implement the test phase. For the run-time phase, we computed "real" dynamic thermal profiles using the RC thermal model. A steady state Kalman filter (KF) implementation was used to estimate the thermal profile for Trojan-active and Trojan-inactive cases. Sensor measurements were made by overlaying noise onto the "real" thermal profile. We assumed sensor noise variance of 0.1 which seems like a worst-case for state-of-the-art thermal sensors [91]. For KF-based Trojan detection, we stored $N = 50$ residuals and chose autocorrelation thresholds $V = 10$ and $a_T = .18, .34, .40$ for 4, 16, and 32 sensors respectively (based on data from the Trojan-free designs). For the hypothesis testing (HT) approach, we used mean difference $S_{\%} = \pm 2.5\%$ to

		RS232-T900			s38417-T300			BasicRSA-T200			MC8051-T300			MC8051-T600		
	# sensors	t_+	f_+	t_{dec}	t_+	f_+	t_{dec}	t_+	f_+	t_{dec}	t_+	f_+	t_{dec}	t_+	f_+	t_{dec}
HT	4	100%	79%	4.9E-2	100%	45%	1.9E-3	100%	66%	1.7E-2	0%	0%	-	100%	66%	5.2E-2
	16	100%	0%	1.7E-1	100%	0%	4.3E-3	100%	0%	3.9E-2	0%	0%	-	100%	0%	1.6E-1
	32	100%	0%	2.1E-1	100%	0%	5.1E-3	100%	0%	4.8E-2	0%	0%	-	100%	0%	2.0E-1
KF	4	100%	0%	1.0E-3	100%	0%	3.6E-3	100%	0%	8.3E-3	1%	1%	1.2E-2	100%	0%	3.7E-2
	16	100%	0%	8.3E-4	100%	0%	2.7E-3	100%	0%	6.3E-3	3%	3%	1.1E-2	100%	0%	3.2E-2
	32	100%	0%	5.9E-4	100%	0%	2.2E-3	100%	0%	5.2E-3	7%	7%	1.0E-2	100%	0%	2.4E-2

Table 3.2: Average true positive rate t_+, false positive rate f_+, and detection time t_{dec} (seconds) for 100 trials. Note detection time is given in seconds and only includes true positives. '-' indicates no true positives.

estimate the Trojan-active stable state PDF S_1. Except where specified, one can assume the experiments were conducted while the ICs were in stable thermal states.

3.5.2 Results

We conducted 100 trials with random sensor noise on both the Trojan-inactive and Trojan-active ICs. We recorded the following data: average true positive rate t_+, average false positive rate f_+, and average time t_{dec} to obtain a true positive. The results are shown for all 5 benchmarks and both detection mechanisms in Table 3.2. "HT" and "KF" denote the local hypothesis testing and global Kalman Filter based approaches respectively.

Local Hypothesis Testing (HT). HT was able to detect the active Trojans (true positives) in all the benchmarks with 100% accuracy except for MC8051-T300. MC8051-T300 had the smallest difference in power consumption between Trojan-active and Trojan-inactive cases (see Table 3.1) and thus there was little deviation

in the thermal profile. False positives on the inactive Trojans were only an issue in the 4 sensor case and went to 0% with additional sensors. Increasing the number of sensors also resulted in slower Trojan detection. Put simply, it took a longer time for the majority of sensors to agree on a true positive when there were more sensors. On average, 4 sensors could detect Trojans 70% faster than 32 (ignoring MC8051-T300), but with higher false positive rate.

Global Kalman Filtering (KF). The KF was also very successful with true positives in every benchmark but one. Once again, the deviation in power/temperature was too small to detect for MC8051-T300. The KF had a false positive rate of zero in all instances but MC8051-T300. In contrast to HT, increasing the number of sensors from 4 to 32 improved detection time by 38% on average (ignoring MC8051-T300). Basically, the more measurements the better the resolution of thermal profile and autocorrelation $\hat{a}$. To illustrate, Figure 3.8 shows $\hat{a}$ of Trojan-inactive (blue) and Trojan-active (red) ICs with 4 and 32 sensors for s38417-T300. The Trojan-inactive autocorrelation stays below the threshold a_T (black line) while the Trojan-active autocorrelation diverges from zero and exceeds the threshold. The Trojan-active case crosses the threshold more quickly in the 32 sensor case.

Note that the results in Table 3.2 correspond to the case where the ICs were in stable thermal states, but we also ran trials at room temperature. For the latter, the KF yielded similar results.

Comparing Local HT and Global KF. While both approaches worked well, the KF achieved better results. The KF found active Trojans 60% faster on average than HT and was effective even with only 4 sensors. Also, while the HT approach

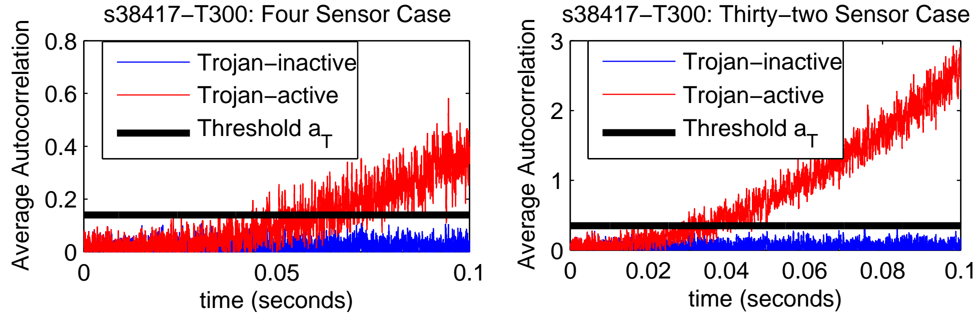


Figure 3.8: Average autocorrelation ($\hat{a}$) over time with 4 and 32 sensors for s38417-T300

could only operate with the ICs in a stable thermal state, the KF worked in all scenarios. The advantage of HT is its lower overheads.

3.6 Summary

In this chapter, we proposed innovative temperature-based approaches for on-line Trojan detection. The merits and key features of our approaches are summarized as follows:

- *Exploit Thermal Sensors For Low Sensing Overhead.* Existing run-time approaches [32, 33] have suffered from high overheads and lack of scalability. Our approach exploits the fact that Trojan activation can cause significant changes in power consumption, which will be reflected in the IC's thermal profile. Our temperature-based Trojan detection has lower sensing overhead compared to prior run-time approaches because many electronic systems are already equipped with thermal sensors for dynamic thermal management.

- *Comprehensive in Nature.* Our novel temperature-based Trojan detection is a basic instance of the comprehensive strategy discussed in Section 1.4. Each

phase of the IC supply chain/process plays a critical role in our method. In the design phase, we statistically characterize an IC's power/thermal dynamics to create "golden models" and place optimally thermal sensors based on these statistics. At the test-time phase, we weed out Trojan-infected ICs with prior detection approaches. We also gather information from ICs that pass logic-based approaches and side-channel analysis in order to calibrate each IC for fabrication variation. The run-time phase integrates the information from the previous phases with thermal sensor measurements to detect Trojan activation. Any Trojans that are detected at run-time can be reported to warn other IC consumers as well as to improve future design instances and test-time verification approaches.

- *Strong Theoretical Foundations.* Existing methods for Trojan detection tend to be ad-hoc in nature. They fail to make optimal decisions regarding Trojan presence and do not directly account for measurement/process noise. In contrast, the temperature-based detection schemes that we propose are more rigorous and rely on fundamental theories from signal estimation and detection theory. Specifically, we use well-known Kalman Filter (KF) theory to combines prior knowledge of state transitions and statistical noise characteristics with sensor measurements in order to optimally estimate IC thermal profiles in time. We also exploit Bayesian decision theory and autocorrelation metrics to create optimal thresholds in our detection schemes.

- *Highly Generalizable.* Our approaches are highly generalizable in various re-

spects. First, the thermal sensing circuits (eg. ring oscillators) are easy to manufacture on processor, ASIC, and FPGA platforms. Second, the KF approach itself can be extended to utilize multiple and heterogeneous sensing modalities for better IC temperature or state tracking. Our proposed framework can also be extended to utilize nonlinear and non-Gaussian filtering approaches such as the Extended Kalman Filter (EKF) and Particle Filter (PF). Finally, while we have only applied our approach to hardware Trojans, it should also be able to handle software Trojan attacks which may have a similar impact on power/temperature profiles.

Chapter 4

Mask Generation for Physically Unclonable Functions

4.1 Introduction

As discussed in Chapter 2, silicon PUFs rely on the existence of variations occurring during IC fabrication to produce unique, unclonable, and tamper-resistant signatures. At a relatively small overhead, the PUF signatures can be used to deal with many hardware-related attacks, such as overbuilding, counterfeiting, and microprobing. While improving PUF quality has been an important area of research, we have recognized two significant shortcomings in prior work:

- *Limited Focus on Fabrication.* Figure 4.1 summarizes the techniques used to improve PUF quality and is organized according to each technique's goal and with respect to its place in the IC design/fabrication process. One can see that existing work (with lone exception in [92]) has only attempted to improve PUF sensitivity to systematic variations (SV), random variations (RV), and environmental variations (EV) at architectural level, at circuit level, and with additional post-fabrication processing. While such techniques successfully exploit the existence of variations, they do not explicitly focus on the fabrication process itself, which is the source of PUF quality in the first place. Rather, they treat the manufacturing process and underlying variations as a black box.

	SV Compensation	RV Enhancement	Resistance to EV
Architectural/Circuit Design	[61, 46, 57]	[41, 55, 62, 56]	[63, 64, 56]
Synthesis/Physical Layout*		[92]	
Mask Generation*			
Fabrication Process *			
Post-Fabrication	[72, 50]		[65, 66, 67, 68, 69, 70, 71]

Figure 4.1: Summary of existing research geared towards improving PUF quality. Columns represent the three main areas of research: Systematic Variation (SV) Compensation, Random Variation (RV) Enhancement, and Environmental Variation (EV) Resistance. Rows represent the steps in the IC design/fabrication process. Starred rows denote steps where research is lacking. New approaches at mask and layout levels are the focus of Chapters 4 and 5 in this dissertation.

- *Current DFM Flow Counterintuitive for PUFs.* With each generation of VLSI technology, the miniaturization and complexity of state-of-the-art products is making it more challenging to achieve high-yielding designs. In short, modern feature sizes make the fabrication process and resulting ICs more susceptible to process variations. In response, there has been an initiative called Design-for-Manufacturability (DFM) in the fabrication community. DFM research models the sources of fabrication variations and develops tools for suppressing all of them. While DFM has been critical to continuous scaling of ICs, PUFs designed and manufactured by such approaches will be more immune to random variations and, therefore, shall generate signatures of lower quality. Furthermore, since the existing techniques for enhancing PUFs (shown in Figure 4.1) do not address the fabrication process or DFM directly, they

must incur larger overheads than necessary to counter DFM's suppression of random variations.

In this chapter and the next chapter of this dissertation, we propose innovative techniques which overcome the above shortcomings. In an attempt to bridge the gap between DFM and PUF research, we take a fundamentally different approach that places greater emphasis on physical layout, mask generation, and the fabrication process to improve PUF quality. Specifically, in Chapter 4, we propose two new mask generation techniques that improve fabrication variation caused in PUFs during IC patterning/lithography. In Chapter 5, we propose new standard cell layouts that reduce the impact of systematic variation on PUFs and show how they can be combined with existing circuit design approaches.

The key features of our approaches and main contributions are summarized as follows:

- *Role-reversal for DFM.* Our approaches exploit the existing DFM models and tools that are ignored in current PUF research. We also develop new algorithms that reverse the role of standard DFM tools and extend them towards improving PUF quality. Rather than trying to suppress all fabrication variations, our techniques are designed to increase sensitivity to random variations and thereby improve the uniqueness, unpredictability, and reliability of PUF signatures. Furthermore, we also develop techniques that focus on suppressing only systematic variations within PUFs. All this is done such that the non-PUF circuits remain unaffected (more below).

- *Effectively Balance DFM and PUF Quality.* Although PUFs rely heavily on random variations, non-PUF portions of an IC (CPU, etc.) desire less variability during fabrication. Our approaches can balance these competing needs. For mask generation, we need only apply our techniques to PUF portions of the IC. The mask for remaining parts of the IC can still be obtained using standard DFM approaches. Since spatial correlations imposed by mask generation algorithms decay very quickly with distance [93], we can easily avoid any interference between the two mask patterns. In the case of standard cells, the cells we develop for PUFs need only be used in PUF portions of the IC.

- *Complimentary to Existing PUF Research.* As shown in Figure 4.1, physical layout and mask generation occur in between circuit design and post-fabrication steps of the standard design/fabrication flow. Since our approaches occur after design, they can be applied to any PUF architecture/circuit design (delay-based, memory-based, etc.) to generate better PUF signatures. Since they enhance PUF quality during fabrication, they can also be complimented by post-fabrication approaches if necessary. For example, the underlying PUF circuits can also utilize existing error correction schemes to obtain further improvements.

- *Lower Overhead.* Since our approaches are fabrication-based, they shall generally improve PUF quality with little architectural or circuit overheads (area, power, etc.). Furthermore, by improving PUF quality, they can also reduce the need for the architectural and circuit blocks required to overcome DFM.

- *Highly Generalizable.* The models, objective functions, framework, and analysis used by our approaches are very general and can be used as a guide in future mask generation, physical layout, and IC fabrications techniques for improving PUF quality.

For the remainder of this chapter, we examine the variations occurring during optical lithography and develop new mask generation techniques to improve PUF quality at fabrication. In Chapter 5, we look at a layout-based approach to improve PUF quality.

Outline. The rest of this chapter is organized as follows.

- As discussed in Chapter 2, variations during manufacturing heavily influence PUF quality. In this chapter, we look at the lithography/patterning step of manufacturing to improve variations in PUFs. Our main goal is to generate lithography masks that result in better PUF signatures. In Section 4.2, we discuss background for optical lithography, Optical Proximity Correction (OPC, often used to suppress fabrication variations, and state-of-the-art OPC-based mask generation.

- In order to understand the impact of variations, we must have effective models for fabrication variation. Section 4.3 discusses variability models from the literature. We also qualitatively illustrates the impact of systematic and random variations on the RO-PUF.

- In Section 4.4, we discuss alternative OPC formulations and cost functions called P-OPC and SVC-OPC that generate mask which improve PUF suscep-

tibility to random and systematic variations respectively. We also highlight how our approaches can be applied without affecting non-PUF portions of the ICs.

- In Section 4.5, we discuss our experimental setup and results. We compare the PUF quality resulting from conventional OPC and our proposed OPC approaches for several different PUF architectures.
- The advantages of our mask-based approaches are summarized in the final section of the chapter.

4.2 Preliminary

Notation. We denote matrices by bold letters and use subscripts and superscripts to denote matrix elements. For example, an element at row j and column i of a matrix $\mathbf{A}$ is given by A_j^i.

4.2.1 Optical Lithography

Optical lithography is the process by which a photoresist covering a silicon wafer is exposed to optical wavelengths and then developed to form desired patterns/structures on the wafer. An optical lithography system consists of two modules [94] (shown in Figure 4.2) which are described below:

1. *Illumination Module.* A mask that contains the desired silicon patterns is illuminated by a light source through an illumination lens. The mask is typically binary, meaning it consists of only transparent and opaque structures

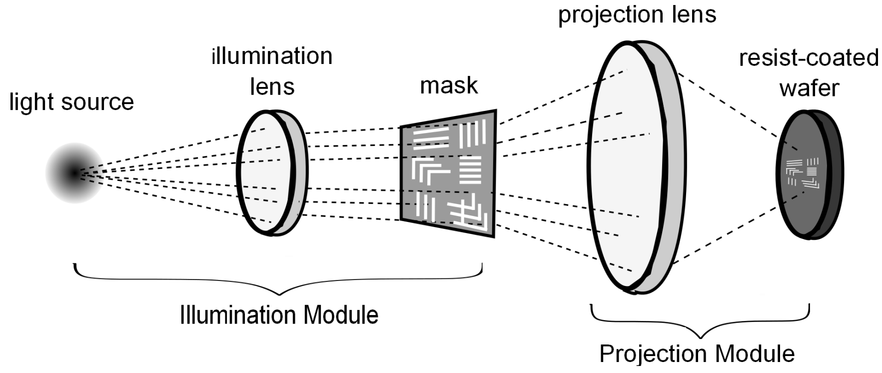


Figure 4.2: Optical Lithography System

which allow light to reach and prevent light from reaching the photoresist respectively.

2. *Projection Module.* Light diffracts as it passes through transparent parts of the mask. A projection lens picks up a portion of this light and projects an "image" (pattern) onto the photoresist. Depending on the type of photoresist, it either hardens or remains soft in presence of light. The portions of the resist that receive enough light are removed by a chemical etching process. The parts of the wafer still covered with resist are protected from doping, deposition, etc.

Lithography Models. The above lithography process and variations have been modeled in the literature (eg. [48]) and there are many simulation tools that estimate the structures in chips resulting from lithography (eg. Calibre Workbench). The lithography process (illustrated in Figure 4.2) is modeled with two basic steps:

1. Light passing through the mask is projected onto the photoresist creating an "aerial" image $\mathbf{I_a}$. Let $M_x^y \in \{0, 1\}$ denote the pixels at the location x, y for the mask $\mathbf{M}$ where zero (one) refers to opaque (transparent) pixels. For a

coherent imaging system, the aerial image is given by [95, 96]

$$\mathbf{I_a} = |\mathbf{M} * \mathbf{h}|^2 \tag{4.1}$$

where $*$ denotes the convolution operator. $\mathbf{h}$ denotes the point spread function (PSF) of the projection lens and is modeled as

$$h_x^y = \frac{J_1\left(2\pi NA\sqrt{x^2+y^2}/\lambda\right)}{2\pi NA\sqrt{x^2+y^2}/\lambda} \tag{4.2}$$

where J_1 is the Bessel function of the first kind, order one [95], NA is the numerical aperture of the projection lens, and λ is the source light wavelength.

2. Once the optical image falls onto the resist-coated wafer, the photoresist is developed and etched based on image intensity at the corresponding wafer location. If the photoresist material is positive (negative) and the image intensity at a certain location is greater (lesser) than a specific threshold, the resist gets etched out. The resulting image is called the resist or pattern image $\mathbf{I}$ and is often calculated as follows [95]

$$I_x^y = \begin{cases} 1, & \text{if } I_{a_x^y} \geq I_{\text{th}} \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where I_{th} is a constant intensity threshold.

Lithography Process Variations. Fundamentally, the optical light pattern falling on the wafer decides what physical structures are fabricated. There are many sources of variation in the optical lithography process which include imperfections in the mask, light source, and lenses as well as variation in the distance between the projection lens and wafer. These sources of variation result in differences between

the pattern appearing on the photoresist and the desired pattern, which in turn result in variation in structures and features as well as performance deviation from design specifications.

Process variations during lithography are typically modeled as a combination of focal and dose variations [48]:

1. *Focal Variations (Defocus):* Focal errors (defocus) are essentially small changes in the distance between the light source and resist/wafer from the ideal setting. Focus variations are modeled as a change to the PSF of the projection lens [96]

$$h(F)_x^y = \mathscr{F}^{-1}\{(\tilde{h}(f_x, f_y)e^{-j\pi F(f_x^2+f_y^2)}\} \tag{4.4}$$

   In Eqn. (4.4), $\mathscr{F}^{-1}$ denotes the inverse Fourier transform, $\tilde{h} = \mathscr{F}(h)$, f_x and f_y are spatial frequencies, and F denotes the defocus. Equation (4.4) is used in place of h_x^y in Eqn. (4.1) to produce an aerial image $\mathbf{I_a}(F)$ for a given defocus F at the wafer plane. This impacts the pattern image $\mathbf{I}$ and therefore the final manufactured transistor/wire parameters. Defocus has both random and systematic components, with the systematic components generally regarded as having six times the standard deviation of the random ones [48].

2. *Exposure Dose Variations:* Exposure dose variations result from differences in light source intensity, exposure duration, etc. Dose variations are often modeled by replacing the constant I_{th} in Eqn. (4.3) with a random variable [94].

Models for the systematic and random variation of defocus, dose, etc. are discussed in Section 4.3.1.

4.2.2 Optical Proximity Correction (OPC)

As IC features scale downward, it is more difficult to print high resolution patterns on the wafer [94]. This is because the source light wavelengths are much larger than modern feature sizes which increases their susceptibility to process variations. Thus, there has been a great deal of work devoted towards developing techniques that make the fabrication process more robust. For example, Optical Proximity Correction (OPC) [94, 96] is a widely used technique that modifies the lithography mask in order to improve the chance of obtaining desired patterns on the wafer. There have been two types OPC algorithms discussed in the literature:

1. *Polygon-based OPC* treats the transparent parts of the mask as a set of polygons. Then the polygons are broken down into edge segments which are iteratively moved with the goal of improving a cost function [93].

2. *Pixel-based OPC* represents each pixel of the mask as a 0 or 1 decision and models the printed pattern as a continuous function [95]. Then an optimization problem is solved by using a gradient descent-like algorithm. These algorithms are also referred to as inverse lithography (ILT).

Basic OPC Algorithm Instance. Before moving on, we briefly discuss a basic OPC algorithm and cost function which we feel summarize state-of-the-art OPC. The algorithm follows a polygon-based approach, but also includes some pixel-based elements. In latter sections, our OPC approaches will closely follow this basic frame-

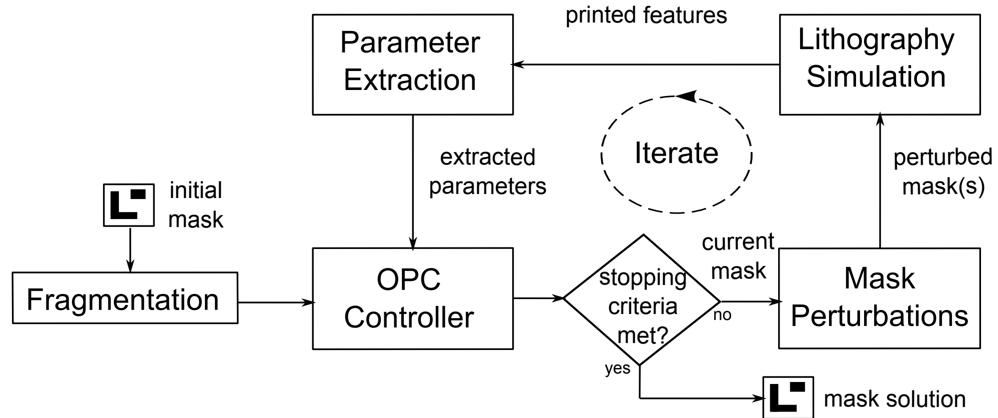


Figure 4.3: Basic OPC Algorithm

work[1]. In a nutshell, the basic algorithm [93] iteratively moves "mask edges" to improve an optimization cost function (see below) until a stopping criterion is met. The main steps are shown in Figure 4.3 and described as follows [93, 97]:

1. *Inputs:* An initial mask pattern is used as input.
2. *Fragmentation:* Polygons within the initial mask are fragmented into edges (see Figure 4.4(a-b)). These fragmented edges are essentially optimization variables. When shorter edges are used, there are more degrees of freedom during the OPC algorithm [93]. The current mask **M** is set to the initial mask.
3. *Mask Perturbation:* One or more edges in current mask **M** are chosen for perturbation (i.e. offset from their current position). For example, a vertical edge can be perturbed by moving one grid unit to the left or right from the current position. Similarly, a horizontal edge can be perturbed by moving it

[1]Note that we illustrate our approaches using this basic framework, but the main concepts should easily extend to other frameworks as well.

up or down by one grid unit. Edges can be chosen for perturbation in a fixed or random order. Two perturbed masks are shown in Figure 4.4(c-d).

4. *Lithography Simulation:* Perturbed masks result in different features in the resist/wafer. Patterns resulting from the perturbed masks are simulated by a calibrated lithography tool.

5. *Parameter Extraction:* The patterns themselves or characteristics of the patterns (eg. channel length, active regions, etc.) are examined. IC performance may also be computed from these patterns [97].

6. *OPC controller:* An objective cost is computed based on the extracted patterns and/or performance. A perturbed mask is accepted if it improves the cost.

7. *Iterate:* If the algorithm's stopping criteria has not been met, the algorithm returns to step 3 and continues. Otherwise, the current mask is output as the solution.

Conventional OPC (C-OPC) Objective. As discussed above, the typical goal of OPC is a mask that obtains desired patterns on the wafer with high probability. Most OPC objective functions will therefore use some difference between desired and estimated patterns/parameters in the algorithm's cost function. Several difference metrics have been used in the literature including Edge Placement Error (EPE) [94] and pixel-based Mean Square Error (MSE) [96].

As an example, we illustrate an objective cost function for state-of-the-art OPC approaches. By state-of-the-art, we refer to OPC that accounts for the variations in lithography. While in general not every OPC cost function models variations,

approaches which do have been shown to produce more accurate IC features (eg. [94, 96]). For ease of exposition, we discuss an MSE-based cost function. MSE between the desired and printed resist patterns can be calculated as follows (dose variation ignored for simplicity)

$$MSE(\mathbf{M}; F) = \sum_{x,y} (\hat{I}_x^y - I_x^y(\mathbf{M}; F))^2 \tag{4.5}$$

where x and y denote the horizontal and vertical coordinates of the pattern; $\mathbf{M}$ is a matrix denoting the lithography mask (each element in $\mathbf{M} \in \{0, 1\}$ with 0 and 1 representing opaque and transparent pixels respectively); $\hat{I}_x^y$ represents the desired pattern at location x,y; $I_x^y(\mathbf{M}; F)$ denotes the estimated pattern at location x,y as a function of mask $\mathbf{M}$ and defocus F. Note that $MSE(\mathbf{M}; F)$ is a random variable since it depends upon defocus F which is random. A mask that generates an IC close to the target design can be determined by *minimizing the expected MSE* [96]

$$\mathbf{M}^* = \operatorname{argmin}\,(\mu_{MSE}) \tag{4.6}$$

$$\mu_{MSE} = \int MSE(\mathbf{M}; F) p(F) dF \tag{4.7}$$

where μ_{MSE} is the expected MSE and $p(F)$ represents the probability distribution function of defocus. For this C-OPC objective, the basic OPC algorithm modifies the mask patterns on each iteration so that the resulting IC patterns more closely resemble the desired patterns across process variations. This cost function would typically be applied to all portions of an IC (including PUFs) and reduce all variation (systematic and random) in the printed patterns.

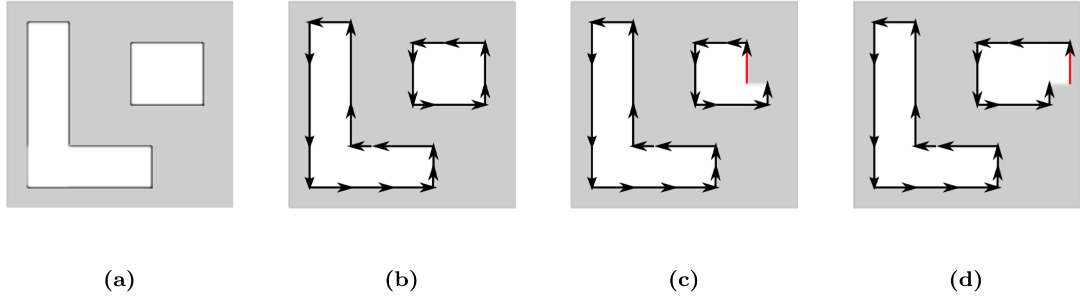


Figure 4.4: (a) Mask: White (gray) areas correspond to transparent (opaque) pixels; (b) Fragmentation; (c) Mask perturbation 1; (d) Mask perturbation 2

4.3 Contributions and Discussion

In this dissertation, we focus on a delay-based PUF called the Ring Oscillator PUF (RO-PUF) which was discussed earlier in Section 2.2.1.2. *Note that this is merely for the sake of illustration. The models, analysis, and algorithms proposed in this dissertation are general and may be applied to other PUF instances as well.*

4.3.1 Wafer Variability Model

PUF quality critically depends on the systematic and random variation present in the PUFs. In this section, we discuss a wafer variability model which frames our discussion on variation in chips and PUFs in future sections. In general, wafer variation consists of the following components: (i) wafer-to-wafer; (ii) across wafer; (iii) field-to-field; (iv) across field; and (v) device-to-device. In this chapter, we adopt the hierarchical model discussed in [98] to analyze and model variation across the wafer. In this model, any parameter p can be modeled as:

$$p = p_0 + p_{\text{sys}} + p_{\text{ran}} \tag{4.8}$$

where

$$p_{\text{sys}} = p_{\text{aw}} + p_{\text{af}} \tag{4.9}$$

$$p_{\text{ran}} = p_{\text{w2w}} + p_{\text{f2f}} + p_{\text{d2d}} \tag{4.10}$$

p_0 is the nominal parameter value and p is the actual value which includes systematic and random deviations (p_{sys} and p_{ran} respectively) from the nominal value. p_{sys} includes an across wafer component (p_{aw}) and an across field component (p_{af}). p_{ran} includes a wafer-to-wafer component (p_{w2w}), a field-to-field component (p_{f2f}), and a device-to-device component (p_{d2d}). p_{w2w}, p_{f2f}, and p_{d2d} are assumed to be zero mean Gaussian random variables. The systematic components p_{aw} and p_{af} are modeled as 2D parabolic functions [98]:

$$p_{\text{aw}}(x_w, y_w) = a_w x_w^2 + b_w x_w + c_w y_w^2 + d_w y_w + e_w x_w y_w + f_w \tag{4.11}$$

$$p_{\text{af}}(x_f, y_f) = a_f x_f^2 + b_f x_f + c_f y_f^2 + d_f y_f + e_f x_f y_f + f_f \tag{4.12}$$

where x_w and y_w denote horizontal and vertical locations on the wafer, x_f and y_f denote horizontal and vertical locations in a field, and the remaining terms (a_w, b_w, etc.) are modeling coefficients. The modeling coefficients for the systematic variation and standard deviations for the random variations can be determined by sampling from wafers as discussed in [98].

The above model is very general and has been used to characterize device channel length, leakage power, and delay. In this chapter, we shall use it to characterize RO oscillation frequency and the defocus experienced by chips/PUFs in a wafer (as well as their corresponding probability distribution functions). For example, Figure 4.5 illustrates the systematic components of defocus using the above

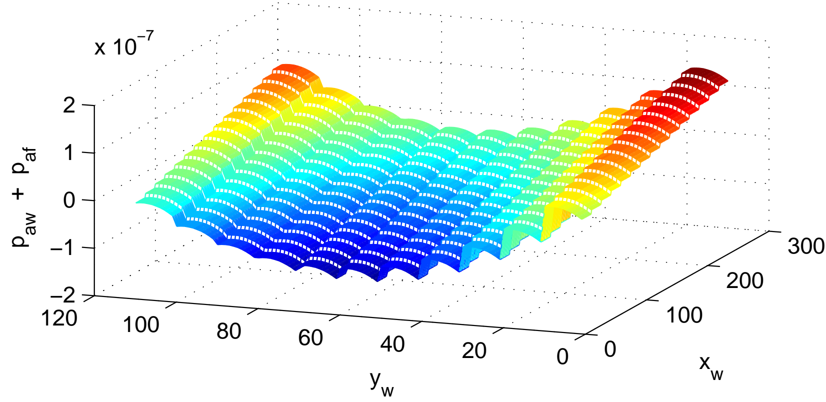


Figure 4.5: Systematic portion of defocus across the wafer (i.e. $p_{\mathrm{aw}}(x_w, y_w) + p_{\mathrm{af}}(x'_f, y'_f)$ where x'_f, y'_f represent the field location in wafer coordinates)

models (Eqns. (4.11) and (4.12)) with coefficients from [98][2]. By including the random components (p_{w2w}, p_{f2f}, and p_{d2d}), one could obtain the defocus distribution for any given chip. In this dissertation, we shall assume that the PDFs associated with defocus are known.

4.3.2 Impact of Variations on PUFs

Below, we discuss how the variations in the lithography step affect the behavior in wafers and PUFs. *Note that this discussion is given in the context of RO-PUFs merely for the sake of illustration. In general, effects of variation on other PUFs can be modeled similarly.*

Assume we have a wafer with n chips, each containing an RO-PUF with m

[2]Note that [98] contains modeling coefficients for effective channel length distributions (calculated from RO frequencies). In this dissertation, we assume that RO frequency variations are caused only by defocus errors and ignore dose variations for simplicity. We obtain defocus distributions by scaling the distributions from [98] to a suitable defocus range (determined from [99]).

ROs. We denote the jth ring oscillator on chip i as RO_j^i and its oscillation frequency by

$$f_j^i = f_0 + f_{\mathrm{sys},j}^i + f_{\mathrm{ran},j}^i + f_{\mathrm{env},j}^i \tag{4.13}$$

where f_0 is the nominal frequency (intended by design), $f_{\mathrm{sys},j}^i$ is the systematic component of the oscillation frequency, $f_{\mathrm{ran},j}^i$ is the random component of the oscillation frequency, and $f_{\mathrm{env},j}^i$ denotes the noise due to environmental variations. $f_{\mathrm{ran},j}^i$ is a random variable that differs for every RO on all the wafers. $f_{\mathrm{ran},j}^i$ comes from lithography as well as semiconductor doping (RDF) and etching (LER). $f_{\mathrm{sys},j}^i$ is mainly from the lithography step and can be modeled as a function of the mask, defocus, and exposure dose. In this dissertation, we assume that the systematic components are constants for all wafers (wafer-to-wafer variation is purely random).

For chip i's RO-PUF, the PUF response bit $r_{g,h}^i$ can be determined by the difference in frequency $\Delta f_{g,h}^i$ between RO_g^i and RO_h^i

$$\begin{aligned}\Delta f_{g,h}^i &= f_g^i - f_h^i \\ &= (f_{\mathrm{sys},g}^i + f_{\mathrm{ran},g}^i + f_{\mathrm{env},g}^i) - (f_{\mathrm{sys},h}^i + f_{\mathrm{ran},h}^i + f_{\mathrm{env},h}^i) \\ &= (f_{\mathrm{sys},g}^i - f_{\mathrm{sys},h}^i) + (f_{\mathrm{ran},g}^i - f_{\mathrm{ran},h}^i) + (f_{\mathrm{env},g}^i - f_{\mathrm{env},h}^i) \\ &= \Delta f_{\mathrm{sys},g,h}^i + \Delta f_{\mathrm{ran},g,h}^i + \Delta f_{\mathrm{env},g,h}^i \end{aligned} \tag{4.14}$$

$$r_{g,h}^i = \begin{cases} 1 & \text{if } \Delta f_{g,h}^i < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

In most applications, nominal PUF responses are recorded in an enrollment/provisioning step (see Section 2.2.2) where $\Delta f_{\mathrm{env},g,h}^i = 0$. After deployment, the responses are re-generated (subject to noise) for authentication or cryptography. There are several

major points about PUF responses that can be drawn from Eqns. (4.13)-(4.15):

- During provisioning, a PUF's nominal response is determined by the random and systematic components only. In instances where $|\Delta f^i_{\text{sys},g,h}| >> |\Delta f^i_{\text{ran},g,h}|$, response $r^i_{g,h}$ will be heavily biased towards one bit value for all the wafers because $\Delta f^i_{\text{sys},g,h}$ is a similar for all wafers (wafer-to-wafer variation is purely random). As a result, the PUF responses shall possess lower uniqueness and lower unpredictability. One should be able to improve this by decreasing $|\Delta f^i_{\text{sys},g,h}|$, increasing $|\Delta f^i_{\text{ran},g,h}|$, or both.
- After deployment, responses may not re-generate exactly due to noise. For $|\Delta f^i_{\text{sys},g,h} + \Delta f^i_{\text{ran},g,h}| < |\Delta f^i_{\text{env},g,h}|$, the response will differ from the provisioned response when the sign of $\Delta f^i_{g,h}$ changes. To improve the reliability, one could increase $\Delta f^i_{\text{sys},g,h}$, increase $\Delta f^i_{\text{ran},g,h}$, or both. However since systematic variations reduce uniqueness, increasing random variations would be the best approach. One could also try to decrease $\Delta f^i_{\text{env},g,h}$ such as in the approaches from Section 2.2.5.2).

4.4 Optical Proximity Correction for PUFs

In this section we propose two new OPC cost functions and corresponding approaches for improving PUF quality. We begin by discussing our assumptions. Then we describe the proposed algorithms and their objective cost functions (random variation enhancement and systematic variation suppression respectively).

Assumptions. As discussed in Section 4.2.1, the wafer patterns resulting from

lithography depend on the mask patterns, defocus, and dose in optical lithography. In this chapter, we shall assume that the dose variations are zero[3]. In that case, the underlying circuits (in our case ROs) depend strictly on the mask patterns and defocus. We shall assume that the defocus in the wafer follows the model given by Eqn. (4.8) and the corresponding probability distribution is known. We shall ignore lot-to-lot variations for simplicity.

4.4.1 PUF-aware OPC (P-OPC)

Our first algorithm is called PUF-aware OPC (P-OPC) and it tries to do the opposite of conventional OPC in PUF portions of the IC. Rather than suppress random variations in the PUF patterns, P-OPC tries to enhance them in order to obtain larger random variation in PUF parameters. Note that the remaining (non-PUF) portions of the IC can still use the conventional approach provided that there is a small "keep out" zone around the PUF that does not contain any non-PUF portions of the IC (see Section 4.4.1.3 for details).

4.4.1.1 P-OPC Objective Cost

To illustrate P-OPC, we discuss an objective cost for RO-PUFs. As discussed in Section 2.2, randomness in the RO frequency improves PUF quality. Our objective function leverages this fact and finds an optimal mask pattern $\mathbf{M}^*$ that increases

[3]This assumption is merely made for simplicity and ease of exposition. Our approaches can easily be extended to handle both dose and defocus.

randomness as follows

$$\mathbf{M}^* = \text{argmax}\left(\frac{\sigma_f}{\mu_{MSE}}\right) \tag{4.16}$$

σ_f represents the variance in RO frequency for a given mask and defocus distribution. μ_{MSE} denotes the average MSE between the intended and printed patterns. Note that σ_f and μ_{MSE} can be computed using lithography simulations and distributions for the defocus (details forthcoming). Our objective function balances two competing goals: (i) we increase randomness in the RO frequency by maximizing the variance in RO frequency (σ_f) across the chips in the numerator; (ii) since increasing frequency variation in the ROs may result in patterns that do not resemble the size and structure of the intended design and may not function correctly, we also include a corrective term in the denominator. The corrective term (μ_{MSE}) captures the fact that we also want to keep the printed RO patterns similar to the desired patterns on average.

4.4.1.2 P-OPC Optimization Algorithm

In this chapter, we use an instance of the basic OPC algorithm shown in Figure 4.3 to solve our optimization problem. For brevity, we only highlight the salient features of the P-OPC version:

- *Inputs:* An initial RO mask pattern is used as input. A discrete probability distribution $p(F)$ for the defocus F for the chips in the wafer is also needed. This could be generated from the model discussed in Section 4.3.1. It can be assumed that an initial cost value is computed for the initial mask.

- *Mask Perturbation:* In each iteration, we perturb every edge of the current mask twice (i.e. move each vertical edge left and right, each horizontal edge up and down). This results in a set of perturbed masks which differ from the current mask by one (different) perturbed edge.
- *Lithography Simulation:* For each perturbed mask, we compute the RO polysilicon patterns $\mathbf{I}(\mathbf{M}, F)$ by a simulation model that employs Eqns. (4.1)-(4.4) for different random values of defocus F (drawn from defocus distribution $p(F)$).
- *Parameter Extraction:* The polysilicon patterns from the above simulations represent physical parameters for the transistors in the ROs (channel length, active region, etc.) which influence the RO frequencies. From the polysilicon layers (which are non-rectangular), we extract effective channel lengths for the transistors in the ROs [100]. A distribution for each RO's oscillation frequency is computed using standard models [101] with the effective channel lengths (from the polysilicon patterns generated by $p(F)$) and nominal electrical parameters (threshold voltage, oxide thickness, etc.).
- *OPC controller:* The OPC controller computes μ_{MSE} and σ_f as follows:

$$\mu_{MSE} = \varepsilon_F\{MSE(\mathbf{M}, F)\} \approx \sum_{\forall F} MSE(\mathbf{M}, F)p(F) \tag{4.17}$$

$$\sigma_f = \sqrt{\varepsilon_F\left\{f(\mathbf{M}, F)^2\right\} - \varepsilon_F\{f(\mathbf{M}, F)\}^2} \tag{4.18}$$

where $MSE(\mathbf{M}, F)$ denotes the mean square error between the intended and printed patterns ($\hat{\mathbf{I}}$ and $\mathbf{I}(\mathbf{M}, F)$) for mask $\mathbf{M}$ and defocus F; $p(F)$ represents the discrete probability distribution function of defocus; $f(\mathbf{M}, F)$ is the ex-

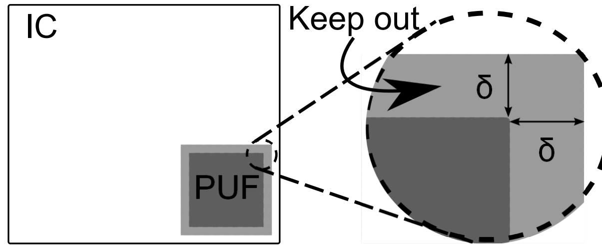


Figure 4.6: Keep Out Zone Illustration: Non-PUF region, PUF region, and keep out zone are shown in white, dark gray, and light gray respectively. δ denotes the length of the keep out zone. Note the figure is not drawn to scale.

tracted frequency for mask $\mathbf{M}$ and defocus F. The cost of each perturbed mask is computed as σ_f/μ_{MSE}. *Note that if a perturbed mask results in a pattern that does not produce functional ROs, then its cost is set to zero and hence it will not be selected as a solution.* The controller chooses the perturbed mask resulting in the greatest improvement to cost as the mask for the next iteration.

- *Stopping Criteria:* (i) We put a limit on the total number of iterations; (ii) If the cost cannot be improved by perturbing any edge before the limit is reached, we terminate.

4.4.1.3 Ensuring Functional Correctness with P-OPC

Since P-OPC increases variation, there is a risk that portions of the IC and the PUF may not function properly. Our approach can overcome reliability issues in the IC and PUF as follows

- *Non-PUF Regions:* We use small "keep out zone" around the PUF which ensures the patterns of light corresponding to the P-OPC algorithm and con-

ventional OPC algorithm do not interfere. An illustration of the IC and PUF with keep out zone is shown in Figure 4.6. The length of the keep out zone, which we denote by δ, depends on parameters of the optical lithography system. From [93], δ is proportional to λ/NA where λ is the wavelength of light and NA is the numerical aperture of the optical system's projection lens. For 90nm process technology, we estimate δ's range as .25 to 1μm from [93].

- *PUF region:* In the parameter extraction step of the algorithm, we extract polysilicon patterns and determine whether the perturbed mask can produce a functional RO with high probability. If not, we guarantee that the perturbed mask is not selected by the OPC controller. In some instances (modeling errors etc.), the P-OPC mask may still result in some PUF failures. One way we propose to overcome this is with self-correcting designs [102]. For example, by fabricating x redundant ROs in the PUF and including some reconfigurable hardware, we could detect failures with built-in self-test and correct up to x failing ROs after fabrication.

4.4.1.4 Dealing with Systematic Variations

As discussed in Section 4.3.2, systematic variation is detrimental to PUF quality while random variation benefits PUF quality. Since the sources of variation in lithography have both random and systematic components [48], the above method will effectively amplify both which is undesired. Thus, the P-OPC approach should ideally be applied when architectural or circuit compensation methods to the PUF

layout/design which counteract systematic variation (eg. [46, 103]) are also applied. This ensures that the variations being amplified are mostly random.

4.4.2 Systematic Variation Compensation OPC (SVC-OPC))

As discussed in Section 4.3.2, in instances where systematic variations are large, the PUF responses can be biased resulting in lower uniqueness and lower unpredictability. One way to improve this is to reduce the systematic variation (and resulting correlation) in the PUFs generated from lithography. In this section, we discuss an algorithm called Systematic Variation Compensation OPC (SVC-OPC) to accomplish this. *Once again, we discuss the algorithm in the context of an RO-PUF, but the same basic framework should also extend to other PUF structures.*

In the SVC-OPC approach, we relax the notion from conventional OPC that the PUF patterns must closely match the design patterns across process variations. Rather, we allow the PUF patterns to deviate from the nominal design patterns as long as it lessens the impact that systematic variation will have on the PUFs. For example, in the RO-PUF, one can allow each RO's silicon patterns to deviate from the nominal patterns as long as the patterns of each RO result in similar systematic frequency components for all ROs in the chip. In that case, when PUF responses are generated (subtracting oscillation frequency of any two ROs), the systematic components will cancel ($\Delta f_{\text{sys}} = 0$). *Note that any reliability and keep out zone related issues with SVC-OPC can be dealt with as previously discussed for P-OPC (see Section 4.4.1.3).*

Notation. Let there be n chips on a wafer and let each chip contain an RO-PUF with m ROs. We denote the jth ring oscillator on chip i as RO_j^i and its oscillation frequency by f_j^i. Let f_j^i be a function of the mask pattern M_j (i.e. the mask pattern used to fabricate RO_j^i $\forall i$) and the defocus experienced by RO_j^i during the lithography step. We assume that the defocus probability distribution functions corresponding to each RO location in the wafers are known. Let $p_j(F)$ denote the defocus distribution (including both systematic and random variation components) for the jth RO across the chips ($1 \leq j \leq m$).

General Objective Goal. Since defocus is similar between wafers (wafer-to-wafer variation assumed to be random), the systematic component of RO oscillation frequency will also be similar for ROs at the same corresponding location on different wafers. Thus, for any given chip, there will be similar (biased) challenge response behavior and signatures (lowering PUF uniqueness) as discussed in Section 4.3.2. In order to remove the systematic bias in any chip's RO-PUF, all the ROs in the RO-PUF should have the same systematic oscillation frequency component (i.e. $f_{\mathrm{sys},g}^i(M_g) = f_{\mathrm{sys},h}^i(M_h)$ $\forall g, h, i$). This effectively means any difference in oscillation frequency between ROs in a chip is purely determined by the random sources (Δf_{rand}) because the systematic components cancel each other out ($\Delta f_{\mathrm{sys}} = 0$). As a result, each PUF's responses (signature) will be as unique and unpredictable as possible. Mathematically, we may express this goal as

$$\mathbf{M}^{*(i)} = \operatorname{argmin} {\sigma_f}^{(i)}, \quad 1 \leq i \leq n \tag{4.19}$$

${\sigma_f}^{(i)}$ represents the variance in the RO oscillation frequencies on chip i. $\mathbf{M}^{*(i)} =$

$\{M_1, M_2, \ldots, M_m\}$ and represents the set of masks for chip i that minimize the variance ${\sigma_f}^{(i)}$. Basically, we want to find mask patterns for all ROs in the RO-PUF such that the systematic variation between all ROs within each chip i is small. In that case, we ensure that each RO-PUF's responses depend mostly on random frequency components. The systematic variation within chips does not matter because the systematic components of RO frequencies will (mostly) cancel when any ROs are compared. This differs from conventional state-of-the-art approaches where the goal is to minimize all variations without regarding the systematic differences between RO frequencies.

4.4.2.1 SVC-OPC Objective Cost

One issue with the above goal (Eqn. (4.19)) is that there may not be one set of masks that minimizes the systematic variation in oscillation frequency within *every chip on the wafer simultaneously* (i.e. $\mathbf{M}^{*(g)} \neq \mathbf{M}^{*(h)}, g \neq h$). However, only one set of masks can be used for all the chips in the wafer. Thus, we use a more relaxed cost function given by

$$\mathbf{M}^* = \operatorname{argmin} \sum_{i=1}^{n} {\sigma_f}^{(i)} \tag{4.20}$$

In Eqn. (4.20), we find the optimal mask patterns that minimize the sum of RO frequency variances ${\sigma_f}^{(i)}$ across the chips. This objective captures the fact that we want small variance in RO frequency within each chip. As discussed above, if the systematic frequency difference between all ROs within each chip i is small, each RO-PUF's responses depend mostly on random frequency components (eg.

semiconductor doping, threshold voltage, etc.). Thus, this should result in responses that are less biased, more unique, and less predictable.

4.4.2.2 SVC-OPC Algorithm

The optimal solution to the SVC-OPC objective function is a set of masks pattern (one mask for each RO in the RO-PUFs, i.e. $M_j, 1 \leq j \leq m$). However, the P-OPC algorithm discussed earlier was only interested in one RO mask for all the ROs in the chip. Thus, we had to modify the P-OPC algorithm with some additional steps to accommodate SVC-OPC. Specifically, we cycle through each RO mask pattern in the chip on every iteration of the algorithm. The SVC-OPC steps are as follows:

1. *Inputs:* Initial mask patterns M_j^0 and the discrete defocus PDFs $p_j(F)$ ($1 \leq j \leq m$) are the algorithm inputs. It can be assumed that an initial cost value is computed for the initial masks.
2. *Fragmentation:* Polygons within M_j^0 are fragmented into edges for each j.
3. *Current Mask Pattern Index:* Let index a denote which mask that we shall perturb and optimize in the current iteration of the algorithm. We begin by optimizing the mask corresponding to the first RO M_1^0. Thus, we set the current mask index to $a = 1$.
4. *Mask Perturbation:* Same as P-OPC.
5. *Lithography Simulation:* Same as P-OPC.
6. *Performance Extraction:* Same as P-OPC.

7. *OPC controller:* Based on the frequency distributions generated in the above steps, the controller computes the cost

$$\sum_{i=1}^{k} \hat{\sigma}_f^{(i)} \tag{4.21}$$

where

$$\hat{\sigma}_f^{(i)} = \sqrt{\varepsilon_M \left\{ f(\mathbf{M}, F_i)^2 \right\} - \varepsilon_M \{ f(\mathbf{M}, F_i) \}^2} \tag{4.22}$$

$$\varepsilon_M \{ f(\mathbf{M}, F_i) \} = \sum_{j=1}^{m} f(M_j, F_i) p_j(F_i) \tag{4.23}$$

$p_j(F)$ denotes the discrete defocus distribution for the jth RO across the chips; k denotes the number of bins in each distribution; $f(\mathbf{M}, F_i)$ represents the frequency at defocus sample F_i; ε_M is an average operation taken over all the RO frequencies at sample F_i. Essentially, $\hat{\sigma}_f^{(i)}$ is the RO frequency variance (includes all m ROs) at sample F_i from the PDFs. The perturbed mask with the greatest improvement to cost is chosen as the ath RO's new mask.

8. *Iterate:* We increment the mask index a and return to step 4. Thus, in the next iteration, we perturb the mask for the next RO in the PUF and calculate the new cost. Note that if $a > m$ (we've cycled through all the RO patterns), we set $a = 1$ (returning back to the first RO mask pattern). The algorithm repeats steps 4-8 until the stopping criteria are met. We use the same stopping criteria as P-OPC.

4.4.3 Qualitative Comparison of P-OPC and SVC-OPC

The P-OPC and SVC-OPC approaches, overheads, advantages, disadvantages, etc. are summarized as follows:

- *Overall Approach.* While both algorithms manipulate the lithography mask to improve PUF quality, they do so with opposite goals in mind. P-OPC tries to increase sensitivity to random fabrication variation. SVC-OPC tries to minimize the impact of systematic variations on the PUFs.

- *Area overheads.* Both approaches require that the PUF regions of the IC be surrounded by a keep out zone where no transistors are fabricated. The keep out zone ensures that non-PUF portions on the IC remain unaffected by the P-OPC and SVC-OPC objective functions. As discussed in Section 4.4.1.3, the length of the keep out zone δ is dependent on the process technology and is generally quite small when compared to the length of the entire IC. Thus, the area overheads in both approaches are small.

- *Algorithm Overheads.* In the RO-PUF instance of the problem, P-OPC solves its objective cost function and obtains one mask for all ROs. Since SVC-OPC tries to eliminate correlation within all the chips on the wafer (a more challenging objective), it obtains a new mask pattern for each RO. Thus, SVC-OPC should have higher runtime overhead. Note that since both algorithms only apply to the small PUF-region of a chip, the overheads should be small relative to the overhead of standard OPC on the whole chip.

- *Designer/Fab Collaboration.* Since both approaches are manufacturer-based, collaboration between the design house and the manufacturer is required. The design house must explicitly reveal the PUF's location to the manufacturer so the manufacturer can apply the proposed algorithms at the PUF region. In

case of large companies which own their own fabs (eg. Intel), collaboration should be no problem whatsoever. In the case of smaller companies, we think that manufacturers could offer this service at a premium to interested customers.

In the next section, we shall simulate the proposed approaches and compare the resulting PUF quality, algorithm overheads, etc. with current state-of-the-art OPC.

4.5 Simulation Experiments

4.5.1 Simulation Setup

4.5.1.1 Simulation Models

We simulated fabrication of RO-PUFs using the models, masks, etc. described below. The RO-PUF architecture is shown in Figure 2.5. Each RO consisted of 3 inverters and ROs were organized in an array on the chips.

Wafer Defocus. Our wafer contains 165 chips. We used the hierarchical model discussed in Section 4.3.1 with coefficients and Gaussian parameters from [98] for defocus. From this model, we also generated discrete probability distributions for defocus which were used in the lithography simulations.

Optical Lithography Parameters. Our feature size (critical dimension) is 90nm. The numerical aperture (NA) is 0.65 and the wavelength of the optical light is 193nm. For simplicity, the illumination source is assumed to be coherent.

Fabrication Simulation. We simulated fabrication of the 165 chips in the wafer. We determined physical and electrical parameters of the ROs in the chips from two sources in the manufacturing process: (i) polysilicon layers generated by the simulations; (ii) random threshold voltages of the RO transistors. Polysilicon layers were computed using the equations discussed in Section 4.2.1, masks generated by the algorithms (see below) and defocus values for the wafer. The defocus values were randomly chosen for each RO in the wafer according to the defocus distribution discussed above. We extracted effective channel lengths [100] for all transistors in the ROs from the resulting polysilicon patterns. Threshold voltages for the transistors in each RO and in each chip were randomly chosen from a Gaussian distribution which varied $\pm 12.5\%$ around a mean threshold voltage. We assumed typical supply voltage, mobility, etc. for the remaining parameters and computed the RO oscillation frequencies as in [101].

4.5.1.2 Mask Generation Algorithms

We generated four PUF populations each containing 165 chips. Masks were generated by the following algorithms and used to simulate fabrication of the ROs in each RO-PUF population:

- *C-OPC:* We generated one mask for all the ROs using the C-OPC cost function and the basic OPC Algorithm discussed in Section 4.2.2. We used the RO polysilicon pattern corresponding to the standard minimum design rules for 90nm devices as the algorithm's initial mask pattern.

- *P-OPC:* We generated one mask for all the ROs using the P-OPC cost function and algorithm discussed in Section 4.4.1. As the initial mask, we used the mask generated by C-OPC.

- *SVC-OPC:* We generated a mask for every RO in the PUF using the SVC-OPC algorithm discussed in Section 5.3.1. As the initial mask for all the ROs, we used the mask generated by C-OPC.

For all the algorithms, we used discrete PDFs with $k = 31$ sampling points/bins.

4.5.1.3 PUF Evaluation

For each PUF, we extracted x response bits at nominal voltage supply in a provisioning step. The x response bits were concatenated together to form an x bit identifier/key for each PUF device. We evaluated the quality of each PUF population's signatures/keys as follows:

- *Uniqueness* was estimated using inter-distance (Eqn. (2.1)).

- *Reliability* was estimated assuming voltage supply variation using intra-distance (Eqn. (2.2)). We randomly varied voltage supply for each inverter in each RO and chip by $\pm 1\%$ and collected $s = 100$ sample responses.

- *Unpredictability (randomness)* was estimated by either min-entropy or the NIST test suite [73]. Min-entropy is computed as

$$H_\infty = -\log_2 \left(\max_{\forall S} (Pr(S)) \right) \tag{4.24}$$

where S represents a PUF signature and $Pr(S)$ denotes the probability of generating signature S. Since it is computationally expensive, we only calculated min-entropy for small RO-PUFs (10 ROs per PUF). We used the NIST test suite for large RO-PUFs (512 ROs per PUF). The NIST test suite takes as input several bitstreams from the same source and determines if the streams can be considered random. We evaluated our PUFs using the NIST test suite in two ways. First, we used the signatures generated by each PUF as the input bitstreams to NIST. We call this the *within-chip* approach since it determines the predictability/randomness among the response bits within each chip's signature. Second, we grouped response bits of all the PUF signatures according to their bit position in the signatures. For x-bit PUF signatures, there are x groups. These x groups were used as the input bitstreams to NIST. We call this the *between-chip* approach because it determines the predictability/randomness of each response bit among the PUFs themselves. The NIST tests we used were the following[4]: Frequency, Block Frequency, Cumulative Sums (2 variants), Runs, Longest Run, Approximate Entropy, and Serial (2 variants).

4.5.1.4 ROs and RO-PUF Response Extraction

We used two architectures in our experiments: one with 10 ROs per PUF and one with 512 ROs per PUF. We used three strategies for generating PUF response

[4]These are the only NIST tests appropriate for the size of our dataset.

bits:

- *Ranking:* ROs are sorted based on oscillation frequency and the sorted order acts as the response/signature [104]. In a PUF with m ROs, there are $m!$ possible ways to sort them which yields $\lceil log_2(m!)\rceil$ response bits.

- *Decoupled Neighbor:* We generated each response bit by comparing two ROs in the PUF. We chose ROs to compare as follows [50]. The ROs were broken into pairs. Each RO was only used in one pair and the ROs in the pair were neighbors in the RO-PUF array. Since nearby ROs experience similar defocus, this setup is naturally resilient to effects of systematic variation. However, assuming m ROs in the PUF, only $m/2$ bits can be generated with this approach.

- *All Possible Pairs:* In this approach, we compared every possible pair of ROs in the PUF rather than just neighbors. Assuming m ROs in the PUF, $m(m-1)/2$ response bits can be generated.

Results for the PUF populations in these three scenarios are discussed below.

4.5.2 Results and Discussion

4.5.2.1 Ranking Approach

We use this approach since it is a strong indicator of how much entropy can be extracted from the RO-PUFs. The RO-PUFs tested in this scenario contained 10 ROs per PUF and the signature/keys were 22 bits long. Table 4.1 shows the

uniqueness, reliability, and min-entropy for each PUF population.

PUF Uniqueness (Inter-distance). We found that the mean uniqueness was closest to the ideal 50% for SVC-OPC. We expected SVC-OPC to have high uniqueness compared to the others since it minimizes systematic variation in the PUFs. P-OPC and C-OPC obtained approximately the same uniqueness.

PUF Reliability (Intra-distance). Table 4.1 shows that the P-OPC approach has mean intra-distance closest to the ideal 0% followed by SVC-OPC. As a result of increasing all variation, PUFs generated by P-OPC are more easily distinguishable and therefore more reliable as the voltage supply varies. C-OPC had the lowest reliability since it essentially minimizes all variation in the ROs.

PUF Unpredictabilty (Min-entropy). The upper bound for min-entropy in this setup is $-\log_2(10!) = 21.7911$. The closest to the upper bound was SVC-OPC with 19.66. This is not surprising because SVC-OPC reduces the systematic variation between the ROs and should make the PUF signatures less biased. P-OPC obtained the second highest min-entropy followed by C-OPC which obtained the lowest. Overall, SVC-OPC and P-OPC masks resulted in 33% and 5.75% improvements respectively over the conventional approach.

4.5.2.2 Decoupled Neighbor Approach

In this approach, systematic variation in the PUFs is reduced by choosing ROs with similar systematic offset. The systematic variations are naturally less since the neighboring RO systematic frequency components will cancel out. The RO-PUFs

		C-OPC	P-OPC	SVC-OPC
Uniqueness	μ	49.5%	49.5%	49.8%
	σ	10.8%	10.7%	10.6%
Reliability	μ	14.9%	8.84%	12.1%
	σ	8.26%	7.62%	8.07%
Unpredictability	H_∞	14.78	15.63	19.66

Table 4.1: Mean (μ) inter-distance and intra-distance, standard deviation (σ) of inter-distance and intra-distance, and Min-entropy (H_∞) for ranking case

tested in this scenario contained 512 ROs per PUF. The signature/keys in this case were 256 bits long. Table 4.2 shows the uniqueness, reliability, and unpredictability for this scenario.

PUF Uniqueness (Inter-distance). Table 4.2 indicates that the PUFs generated by each approach have very similar means and standard deviations for inter-distance. This is not surprising since the decoupled neighbor approach reduces systematic variation in the PUFs.

PUF Reliability (Intra-distance). The results show that the P-OPC approach has the best reliability. Since the P-OPC approach increases random variation between RO frequencies, response bits (which are generated by comparing two ROs) are less likely to flip due to voltage supply noise. The C-OPC and SVC-OPC approaches are more susceptible to noise because they do not increase variation between ROs in the chips. Since C-OPC explicitly reduces all variation, it has the lowest reliability.

PUF Unpredictabilty (NIST tests). Table 4.2 shows how many within-chip and

		C-OPC	P-OPC	SVC-OPC
Uniqueness	μ	50.0%	50.0%	50.0%
	σ	3.10%	3.13%	3.15%
Reliability	μ	15.8%	9.52%	12.2%
	σ	3.88%	4.93%	4.84%
NIST	within-chip	9/9	9/9	9/9
	between-chip	9/9	9/9	9/9

Table 4.2: Mean (μ) inter-distance and intra-distance, standard deviation (σ) of inter-distance and intra-distance, and NIST pass rates for decoupled neighbor case

between-chip NIST tests were passed in each mask generation approach. We found all the approaches did quite well since the decoupled neighbor approach reduces systematic variation in the PUF architecture. The C-OPC, SVC-OPC, and P-OPC approaches passed all the tests which indicates that their signatures are random.

4.5.2.3 All Pairs Approach

This approach should be more susceptible to systematic variation since ROs fabricated with very different defocus values are used to generate responses. However, it can generate longer PUF signatures compared to the decoupled neighbor approach. The RO-PUFs tested in this scenario contained 512 ROs per PUF. The signature/keys were as large as 130816 bits long. Tables 4.3 and 4.4 show the uniqueness, reliability, and unpredictability for this scenario.

PUF Uniqueness (Inter-distance). Table 4.3 shows that the SVC-OPC approach obtained average uniqueness closest to the ideal 50% and with lowest stan-

dard deviation. C-OPC obtained the worst average uniqueness compared to the other approaches but lower standard deviation that P-OPC.

PUF Reliability (Intra-distance). Table 4.3 also shows the reliability. The results were similar to the previous two cases and will not be discussed for brevity.

PUF Unpredictabilty (NIST tests). Using every RO pair possible (130816) results in correlation in the bitstreams input to NIST because the response bits are not independently generated. Thus, none of the approaches passed the NIST tests in this scenario. In order to compare the approaches and still potentially pass the NIST tests, we generated new keys as follows:

- We use the 256 challenges from the decoupled neighbor case as input to the PUF and generate the first 256 bits of every PUF's key.
- To generate a key of x total bits, we arbitrarily choose $(x - 256)$ additional challenges. Each arbitrary challenge utilizes a new RO pair (i.e. different from the original 256) and generates another response bit.
- The same 256 challenges and $(x - 256)$ arbitrary challenges are used as input to all the PUFs in each population to generate their keys.

In practice, we generated arbitrary challenges by randomly choosing new RO pairs. We did this 100 times and ended up with 100 "sample keys". We also repeated the process for several different values of x and obtained keys that were 320, 384, 448, and 512 bits long. We ran the NIST tests for each of the 100 samples and values of x. We recorded the percentage of cases that passed all the NIST tests in Table 4.4. The results show that only the SVC-OPC approach could generate keys longer than

		C-OPC	P-OPC	SVC-OPC
Uniqueness	μ	47.3%	48.5%	50.0%
	σ	3.61%	6.49%	3.19%
Reliability	μ	15.0%	8.71%	11.6%
	σ	3.73%	4.81%	4.29%

Table 4.3: Mean and standard deviation of inter-distance and intra-distance for all pairs case

	key size (bits)			
	320	384	448	512
C-OPC	0%	0%	0%	0%
P-OPC	0%	0%	0%	0%
SVC-OPC	46%	32%	16%	14%

Table 4.4: NIST pass rates for 100 random challenges/keys

	time (hr:min:sec)
C-OPC	00:42:41
P-OPC	01:39:05
SVC-OPC	05:17:12

Table 4.5: Mask Generation Overhead

256 bits that still pass NIST tests. Furthermore, as one would expect, the longer the key the less often the NIST tests were passed. The results support our prior results for the ranking approach which indicated that the keys generated by the SVC-OPC approach have larger min-entropy.

4.5.2.4 Mask Generation Algorithm Overheads

We also recorded the time required by the mask generation algorithms in Table 4.5. Note that since the P-OPC and SVC-OPC algorithms used the C-OPC mask as initial input, we have included the initialization (C-OPC) in their overall runtime. Thus, both cases have larger overhead than C-OPC. Comparing SVC-OPC

and P-OPC overheads, SVC-OPC required much more time because it generates a mask for every RO in the PUF. In any case, since masks only need to be generated once for the PUF region and can be used to fabricate PUFs in countless chips, these overheads should be acceptable.

4.5.3 Summary of Results

The results in this section can be summarized as follows:

- The proposed approaches (SVC-OPC and P-OPC) outperform the conventional approach (C-OPC) in at least one of the PUF quality metrics (uniqueness, reliability, unpredictability) and yield comparable results in others.

- The SVC-OPC approach generates PUFs that have excellent uniqueness and unpredictability. The uniqueness/unpredictability are high in all our test scenarios because the SVC-OPC masks reduce systematic variation. Results also showed that only SVC-OPC could generate longer keys that pass the NIST tests. While the reliability was lower than P-OPC, it still better than the conventional approach (21% improvement). We also feel that this could always be improved by using SVC-OPC on variation resistant circuit designs [63, 64] and shall investigate this in future work.

- The P-OPC approach generates PUFs with higher variation between ROs. The uniqueness/unpredictability end up lower than SVC-OPC since P-OPC doesn't suppress the systematic variation. However, the larger overall variation results in higher reliability than SVC-OPC and C-OPC (40% improvement

over C-OPC).

- The SVC-OPC and P-OPC mask generation algorithms have the larger run-time overheads than C-OPC. However, since masks only need to be generated once for the PUF region and can be used to fabricate countless chips, the overhead should be acceptable for practical applications.

4.6 Summary

In this chapter, we presented a novel framework for improving PUFs which focused on mask generation for lithography. The merits of the approaches from this chapter are as follows:

- *PUF Quality Improvements at Mask Generation Step:* Our approaches are the first to exploit existing DFM models and tools that are ignored in current PUF research. Specifically, we used the hierarchical variability model [98] to express random and systematic variations in wafers. Then, we re-investigated OPC with new PUF-based cost functions to make improvements in PUF quality: one that enhanced random variations and one that suppressed systematic variations. Since both approaches occur after design, they can be applied to any PUF architecture/circuit design (delay-based, memory-based, etc.) to generate better PUF signatures.

- *Lower Overhead:* Our approaches do not have the same hardware overheads as architectural/circuit based approaches. The main overheads occur during mask generation and, since the portion of the mask for the PUF is small

compared to the rest of the IC design, these overheads are small. Our two approaches also require a keep-out zone around the PUF. However, the length of the zone is on the order of .25 to 1μm for 90nm process technology, which is an insignificant cost to chip area.

Chapter 5

Custom Cell Layouts for Physically Unclonable Functions

5.1 Introduction

In the previous chapter, we demonstrated how Optical Proximity Correction (OPC) could be used to manipulate systematic and random variations for better PUF quality. Being that OPC applies to the lithography mask, it is best suited for fabrication companies rather than fab-less IC design houses (who cannot control the mask). Our objective in this chapter is to investigate DFM techniques applied at the physical layout level, which is the last portion of the design in the hands of the designer.

Fundamentally, the physical layout affects the wafer topography, CMP, optical proximity effects, etc.; all of which contribute to PUF variability. Thus, by altering the layout, one should be able to manipulate manufacturing variations and once again improve the uniqueness, reliability, and unpredictability of PUF signatures. In this chapter, we take advantage of a technique called self-compensation [103] and use it to create customized standard cell layouts for PUFs that are less sensitive to systematic variations. We also combine our custom cell layouts with random variation enhancing PUF circuit designs. This combined procedure effectively allows the designer to simultaneously increase sensitivity to random variations while suppressing systematic variations for better PUF quality.

Outline. Background on the self-compensation technique is discussed in Section 5.2.1. Section 5.2.2 discusses the impact of variations on delay of PUFs. Our self-compensation-based optimization procedure for creating custom cell layouts for PUFs is described in Section 5.3.1. Two random variation enhancing circuit designs are discussed in Section 5.3.2. Experimental setup and results are discussed in Section 5.4. We summarize the main results and merits of our approach in the last section.

5.2 Preliminary

5.2.1 Self-compensation

As discussed in the previous chapter, defocus is one of the main sources of variation in the patterning/lithography step of IC fabrication. The impact of defocus depends heavily on the density of features/patterns in the IC's physical layout [48]. Features that are further apart (isolated or iso) result in shorter channel lengths while dense features obtain longer channel lengths under varying defocus [103] (see Figure 5.1).This behavior is important because devices with shorter/longer channel lengths have faster/slower performance. Since defocus is more systematic in nature, performance (delay, power, etc.) will be systematically biased within chips/PUFs and can be predicted with knowledge of chip layout, cell layout, and defocus values.

In [103], the authors exploit the behavior shown in Figure 5.1 to reduce IC timing and leakage power sensitivity to systematic variations. We discuss the basic idea below and then extend it to PUFs in the next section.

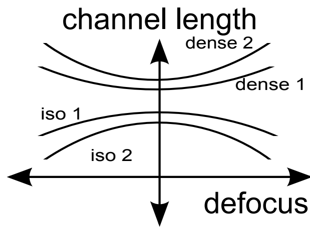
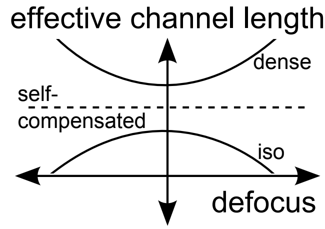
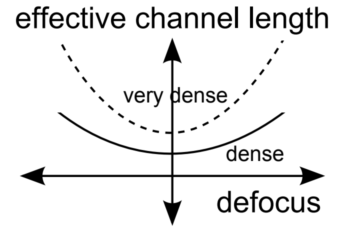


Figure 5.1: Systematic and opposing behavior of dense and iso patterns with defocus.



(a)



(b)

Figure 5.2: Effective channel lengths (dotted) for cells in series. (a) dense and iso cells in series; (b) two dense cells in series

Self-Compensation Technique. Given the opposing behavior of dense and iso patterns during fabrication (see Figure 5.1), [103] shows that one can compensate for systematic variation in two ways:

1. *Intra-cell Self-Compensation.* Within a cell, transistor polysilicon, interconnects, etc. can be divided into iso/dense lines for performance compensation. Assuming the transistors in a given cell have the same defocus value (valid assumption since they are in close proximity), if an "iso" transistor and "dense" transistor are placed in series, then at any defocus their respective delays (iso-fast, dense-slow) will balance out, thereby resulting in an "effective" channel length that is independent of defocus (i.e. self compensated) as shown in Figure 5.2(a). In contrast, if both cells are dense or iso, the overall delay and effective channel length vary more drastically with defocus (see Figure 5.2(b)).

2. *Inter-cell Self-Compensation.* The cells within a circuit can be divided into iso/dense patterned cells for performance compensation as well. For example, if an "iso" cell and "dense" cell are placed in series, then at any shared defocus

value their respective delays (iso-fast, dense-slow) shall balance out. Thus in this case, the delay of the circuit should be similar to an outsider regardless of the defocus experienced during chip fabrication.

Typical circuits are designed with all iso cells (for faster speed) or all dense cells (for lower power leakage and chip area). However, since such cells/circuits vary heavily with defocus, ICs using either approach will suffer from high systematic variation. Self-compensation essentially makes individual cells or interconnected cells (circuits) insensitive to defocus so that every cell or circuit has similar performance regardless of chip location (no systematic variation across chips).

5.2.2 Impact of Variations on PUFs.

As discussed in the previous chapter, the nature of the variations experienced during manufacturing has a large influence on PUF quality. In this section, we qualitatively discuss the impact of systematic and random fabrication variations on delays within PUFs below. *Note that the discussion is given in the context of RO-PUFs merely for the sake of illustration. In general, effects of variation on other PUFs can be modeled similarly.*

Assume we have an RO-PUF with m ROs. We denote the jth ring oscillator as RO_j, the delay through RO_j by d_j, and RO_j's oscillation frequency by f_j. The delay and oscillation frequency of RO_j may be expressed as follows

$$d_j = d_0 + d_{\mathrm{sys},j} + d_{\mathrm{ran},j} \tag{5.1}$$

$$f_j = \tfrac{1}{2d_j} \tag{5.2}$$

where d_0 is the nominal delay (intended by design), $d_{\text{sys},j}$ is the systematic component of the RO's delay, and $d_{\text{ran},j}$ is the random component of the RO's delay. $d_{\text{ran},j}$ is a random variable that differs for every RO and in every PUF. $d_{\text{sys},j}$ is mainly from the lithography step and while it varies within each chip (i.e. $\forall\ j$) it is the same between chips/wafers (see Section 4.3.1). Note that in the above formulation, environmental noise and across wafer variation have been ignored for simplicity.

The RO-PUF response bit generated by comparing RO_g and RO_h, $r_{g,h}$, is determined by the difference in delay $\Delta d_{g,h}$ between the two ROs. Ignoring noise, we can express $\Delta d_{g,h}$ and $r_{g,h}$ by

$$
\begin{aligned}
\Delta d_{g,h} &= d_g - d_h = (d_{\text{sys},g} + d_{\text{ran},g}) - (d_{\text{sys},h} + d_{\text{ran},h}) \\
&= (d_{\text{sys},g} - d_{\text{sys},h}) + (d_{\text{ran},g} - d_{\text{ran},h}) \\
&= \Delta d_{\text{sys},g,h} + \Delta d_{\text{ran},g,h} \qquad (5.3)
\end{aligned}
$$

$$
r_{g,h} = \begin{cases} 1 & \text{if } \Delta d_{g,h} < 0 \\ 0 & \text{otherwise} \end{cases} \qquad (5.4)
$$

$\Delta d_{\text{ran},g,h}$ is random. $\Delta d_{\text{sys},g,h}$ is a function of the ROs' locations on the chip through systematic defocus and the PUF's physical/cell layout. The effects of $\Delta d_{\text{ran},g,h}$ and $\Delta d_{\text{sys},g,h}$ on RO-PUF response are as follows:

- In instances where $|\Delta d_{\text{sys},g,h}| >> |\Delta d_{\text{ran},g,h}|$, the response $r_{g,h}$ is heavily biased in the PUF population, resulting in lower uniqueness and unpredictability. For example, if we compare an RO that experiences zero defocus with one that experiences large defocus (assuming both ROs are composed of standard cells), responses in the PUF population will be biased towards 1 or 0 depending

on the density of features in the RO cells. If the cells have isolated (more dense) features, the second RO will be faster (slower) on average resulting in a response biased towards 1 (0). Lowering $|\Delta d_{\text{sys},g,h}|$ should reduce such biases and result in more unique and unpredictable responses. In Section 5.3.1, we accomplish this by exploiting self-compensation and carefully selecting the density of cells that make up the ROs.

- For large $|\Delta d_{\text{sys},g,h}|$ and/or $|\Delta d_{\text{ran},g,h}|$, the difference in delay between any two ROs ($\Delta d_{g,h}$) is on average greater. The larger gap in behavior of ROs should make it more difficult for environmental variations (voltage supply, temperature, etc.) to alter responses (i.e. change sign of $\Delta d_{g,h}$). This implies greater reliability. Since the self-compensation approach reduces systematic variation, it may actually degrade PUF reliability. One way to compensate for losses in reliability is to increase $\Delta d_{\text{ran},g,h}$. We discuss approaches to do this in Section 5.3.2.

5.3 Proposed Approach

5.3.1 Self-compensated Cell Layouts for PUFs

As discussed above, we want to make $\Delta d_{\text{sys},g,h}$ close to zero $\forall$ h, g in order to improve PUF uniqueness and unpredictablity. The self-compensation technique (discussed in Section 5.2.1) is one way to effectively accomplish this goal. In self-compensation, the density of cells in a layout is chosen carefully to remove systematic

biases in IC timing or leakage power. In our case, we choose the combination $\vec{C}$ of cell densities in the ROs in order to reduce the impact of systematic biases in RO frequencies and PUF responses. The formulation and objective are discussed below.

Let there be N inverter cells per RO. We assume that the delay of any inverter cell is a function of the cell's density p and its defocus value F at manufacturing time. Then, the delay (frequency) of an RO can be expressed as the sum of its N inverters in series

$$d = \sum_{i=1}^{N} d_i(p_i, F_i) \tag{5.5}$$

d_i is the delay of the ith inverter in the RO. p_i and F_i denote the cell density and defocus respectively of the ith inverter in the RO. Note that since the designer has control over the physical layout, the p_i's are variables within our control. On the other hand, the defocus values (F_i's) are variables determined by the manufacturing process and, hence, they are not within our control. While defocus is uncontrollable, its statistical characteristics can be determined by sampling wafers in an early manufacturing process (see Section 4.3.1). Hence, we assume that the probability distribution function (PDF) of defocus is known and denote it by $pdf(F)$.

As shown in Figure 5.1, the effective channel length of the inverter will shrink or grow depending on density p_i and defocus F_i. If the inverter cell is more dense (isolated), the effective channel lengths will grow (shrink) with defocus. By choosing the inverter cell densities carefully, we can reduce the impact of systematic defocus on PUF responses. In our formulation we choose the combination of cell densities

$\vec{C} = [p_1, p_2, \ldots, p_N]^T$ according to the following objective function:

$$\vec{C}^* = \operatorname{argmin} \sigma_d, \tag{5.6}$$

where

$$\sigma_d = \sqrt{\frac{1}{k} \sum_{j=1}^{k} (d_j - \mu_d)^2} \tag{5.7}$$

$$\mu_d = \sum_{j=1}^{k} d(\vec{C}, \vec{F}_j) pdf(\vec{F}_j) \tag{5.8}$$

In the above equations, σ_d represents the sampling variance of delay in the population of ROs in the PUFs; μ_d represents the average RO delay and is computed using the discrete defocus PDF $pdf(\vec{F})$; $\vec{F}_j$ is a vector denoting the defocus experienced by each inverter in the RO for sample j of the defocus PDF; and k denotes the number of bins in the PDF. The goal of the above objective function is to minimize the variance in the RO delays (frequencies) across the chip according to the defocus PDF. This goal is similar to the one discussed in Section 5.3.1 for SVC-OPC. In order to remove the systematic bias in any chip's RO-PUF, all the ROs in the RO-PUF should have the same systematic delay/frequency component (i.e. $d_{\text{sys},g} = d_{\text{sys},h}$ $\forall g, h$). This effectively means any difference in delay/frequency between ROs in a chip is purely determined by the random sources (Δd_{ran}) such as RDD, LER, etc. because the systematic components cancel each other out ($\Delta d_{\text{sys}} = 0$). As a result, each PUF's responses (signature) will be as unique and unpredictable as possible. Note that alternative objective functions are also possible and shall be investigated in future work.

In Section 5.4, we solve the above problem and compare the PUF responses

resulting from our self-compensated cell layouts with standard layout approaches (i.e. fixed cell densities for ROs in the PUFs).

5.3.2 Reliability Enhancement

As discussed in the previous chapter, increasing random variation in PUFs has several benefits to PUF quality: better uniqueness, better unpredictability, and higher reliability. In this section, we discuss circuit-based approaches for increasing sensitivity to random variation in PUFs.

Targeted Source of IC variations. Threshold voltage is an independent random variable with respect to doping [49]. Thus even for transistors that are located near one another within a chip and across chips, there is significant random variation in threshold voltage with little spatial correlation. In this section, we utilize this independence property to amplify random variations by adding extra inverters and transistors to PUF designs. *Note that since inverters are used in many silicon-based PUFs (eg. RO-PUF, SRAM PUF, arbiter PUF), the inverter-based approach is quite general. For the sake of illustration, we describe our approach for an RO-PUF.*

Inverter Cell and RO Behavior: A standard inverter cell contains a single PMOS transistor and single NMOS transistor (see Figure 5.3(a)). Let the threshold voltages for these transistors be random variables represented by v_{tp} and v_{tn} respectively. The delays measured during high-to-low and low-to-high transitions at the inverter output (d_{HL} and d_{LH}) can be described as functions of the threshold

voltages and sizes of the transistors [105]:

$$d_{\text{HL}} = C_{\text{out}} \int_{\frac{V_{\text{dd}}}{2}}^{V_{\text{dd}}} \frac{dV_{\text{out}}}{I_{\text{DS}_n}} \tag{5.9}$$

$$d_{\text{LH}} = C_{\text{out}} \int_{0}^{\frac{V_{\text{dd}}}{2}} \frac{dV_{\text{out}}}{I_{\text{DS}_p}} \tag{5.10}$$

where

$$I_{\text{DS}_n}(v_{tn}) = \begin{cases} g\left(V_{\text{dd}} - v_{tn}\right)^{\alpha} & \text{, Saturation} \\ gV_{\text{out}}\left(V_{\text{dd}} - v_{tn} - \frac{V_{\text{out}}}{2}\right) & \text{, Linear} \end{cases} \tag{5.11}$$

$$I_{\text{DS}_p}(v_{tp}) = \begin{cases} g\left(V_{\text{dd}} - v_{tp}\right)^{\alpha} & \text{, Saturation} \\ g\left(V_{\text{dd}} - V_{\text{out}}\right)\left(\frac{V_{\text{dd}}}{2} + v_{tp} - \frac{V_{\text{out}}}{2}\right) & \text{, Linear} \end{cases} \tag{5.12}$$

$$g = \tfrac{1}{2}\mu C_{\text{ox}} \tfrac{W_{\text{eff}}}{L_{\text{eff}}} \tag{5.13}$$

C_{out} is the capacitance present at the inverter's output. I_{DS_n} and I_{DS_p} denote the drain-to-source currents (NMOS and PMOS respectively) with mobility μ, oxide capacitance C_{ox}, effective channel width W_{eff}, effective channel length L_{eff}, supply voltage V_{dd}, and modeling parameter α. Intuitively, the above expressions model delay as the time it takes to charge and discharge V_{out} at the inverter's output capacitance.

With the above delay models, the delay for N inverters in series (i.e. ring oscillator RO delay d) can be approximated by summing the rise and fall delays of each inverter [105] and the oscillation frequency of the RO can be computed as $f = \frac{1}{2d}$. Note that since threshold voltages v_{tn} and v_{tp} are random variables, d, f, etc. are also random and will be different for ROs within and between chips. By increasing the variance in d, there will be more variability in RO frequencies in each

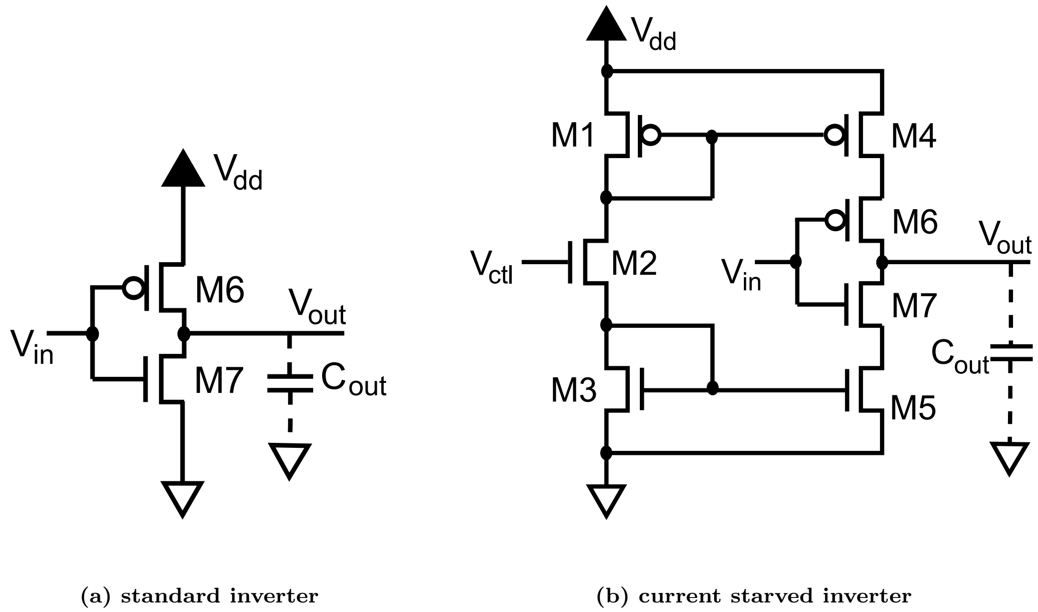


Figure 5.3: Standard and current starved inverters

chip. For a PUF, this should represent an increase in reliability since the ROs being compared will be separated by larger Δd_{rand} (as discussed in Section 5.2.2).

Extra Inverters: One way to increase variance in d is to increase the number of inverters (N) in the RO. This works for two reasons:

1. The transistors in each inverter have independent and random threshold voltages. Therefore, the contributions of each inverter to overall RO delay are also independent and random.

2. The variance of a sum of independent variables is the sum of the variances of the independent variables. Since the inverter delays are independent and the RO delay is computed by summing them, Δd_{rand} will increase as inverters are added to the RO.

Using the above models for current, delay, etc. (Eqns. (5.9)-(5.13)), we de-

	mean		variance	
	μ_d	normalized	σ_d	normalized
N=3	6.64	1.00	37.0	1.00
N=7	15.5	2.33	56.8	1.54
N=13	28.8	4.33	77.1	2.09

Table 5.1: Mean (psec) and standard deviation (fsec) in RO delay as number of inverters (N) increase in the RO

termined the mean and standard deviation of d using Monte Carlo simulations. Table 5.1 shows the mean and standard deviation with number of inverters N=3, 7, and 13 where the means and standard deviations have been normalized by a base case (N=3). As N increases, there is an increase in both mean and variance of d compared to the base case. We will investigate RO-PUF quality using extra inverters in Section 5.4

Current Starved Inverter. An alternative way to increase the random variation in the RO delay is to add transistors to the inverter circuits in such a way that I_{DS}, d, f, etc. are functions of more random variables. This is essentially what occurs in the current starved inverter [62] shown in Figure 5.3(b). The inverter transistors are denoted by M6 and M7 and the remaining transistors M1-M5 make up current mirrors for M6 and M7.

The key concept of the current mirror in the pull-up network of the current starved inverter is summarized as follows. The gate and source of M1 are wired together so that the M1 transistor is always in saturated mode (when on). V_{ctl} is a control voltage that (along with the random threshold voltages, widths, lengths of

M1, M2, and M3) determines the gate voltages of M1 and M4 and the current I_{ref} through M1, M2, and M3. During the the low-to-high transitions at the inverter output, the current I_{out} through M4 and M6 is a function of I_{ref} (and the random threshold voltages, widths, lengths of M4 and M6). Thus, the charging behavior at C_{out} is a function of a greater number of random variables than the standard inverter case. The current mirror for the pull-down network and discharging behavior at C_{out} can be explained similarly.

Since the current starved inverter was shown to improve PUF quality in [62], we shall only focus on our contribution (extra inverters) and will not experiment with the current-starved inverter in the remainder of the chapter.

5.3.3 Combined Approach

We feel that there are three important benefits achieved by combining the approaches from Sections 5.3.1 and 5.3.2:

1. In general, components in PUF circuits should possess high sensitivity to random variations and low sensitivity to systematic variations. By combining the two techniques we should effectively accomplish both of these goals.

2. In inter-cell self-compensation, each cell's density is essentially an optimization variable that can be used to tune and lower systematic variation due to defocus. By adding inverter cells, we are actually adding more tuning variables and can thereby expect more opportunities to reduce systematic variation in the PUFs.

3. Since reducing systematic variation might actually harm PUF reliability (see

Section 5.2.2), adding additional inverters can compensate for this trend by increasing randomness in the PUF.

We shall evaluate the proposed approaches and combined approach in Section 5.4

5.4 Simulation Experiments

5.4.1 Experimental Setup

In this section, we simulate RO-PUFs using several standard and proposed schemes. In each scheme, we simulate a population of 100 PUF instances. Each PUF contains 512 ROs and each RO contains a fixed number of inverters denoted as N. One PUF response bit is computed by comparing the oscillation frequencies of 2 out of the 512 ROs. Each RO is only used in one comparison so there are 256 response bits per PUF. The RO pairs are selected randomly and the same pairs are used for each RO-PUF in the population.

# of Inverters. As discussed in Section 5.3.2, random variation should increase with the number of inverters N in an RO. We vary N from 3 to 13 in our experiments and examine the effects on RO-PUF quality.

Cell Density Types. To analyze self-compensation (Section 5.3.1), we utilize five types of inverter cells:

1. *Very Dense (VD)* contains features which are as close as permitted by the lithography process.

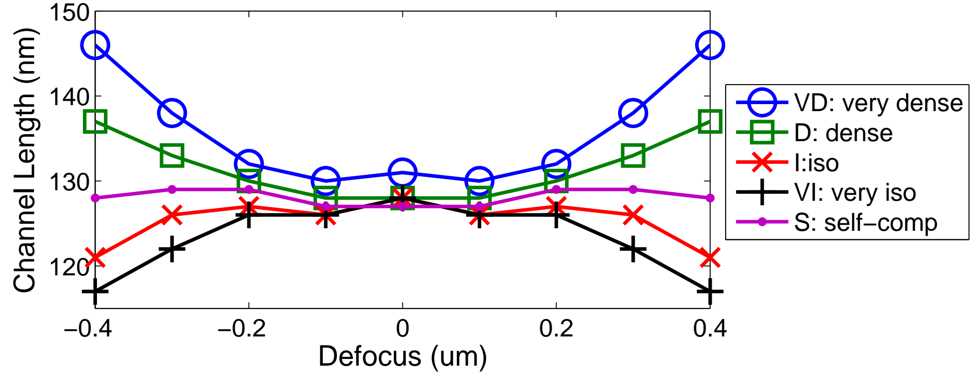


Figure 5.4: Channel length vs. Defocus for cell layout types: (1) Very Dense VD; (2) Dense D; (3) Isolated I; (4) Very Isolated VI; (5) Self-Compensated S

2. *Dense (D)* contains features which are close together, but less so than VD.

3. *Very Isolated (VI)* contains features which are as far as possible.

4. *Isolated (I)* contains features which are spaced far apart, but not as much as the Very Isolated (VI) case.

5. *Self-compensated (S)* utilize intra-cell self-compensation so that features are spaced in such a way that channel length varies little with defocus.

The channel lengths of each cell type versus defocus condition are shown in Figure 5.4 (data from [103]). As one can see, the dense cells "smile" and the isolated cells "frown" while the self-compensated cell is more flat w.r.t. defocus.

Cell Combination Types. We generate several PUF populations with ROs that use different combinations of these cell types. We refer to a combination as "standard" if there is no intra-cell or inter-cell self-compensation in the layout/cells (eg. all three inverter cells in the ROs are VD type). We also examine two "self-compensated" cell combinations which were obtained by solving the objective shown

	Case	RO Inverter # 1	2	3	4	5	6	7	8	9	10	11	12	13
$N = 3$	1	D	VI	S	-	-	-	-	-	-	-	-	-	-
	2	VI	S	S	-	-	-	-	-	-	-	-	-	-
$N = 7$	1	VD	D	VI	VI	VI	S	S	-	-	-	-	-	-
	2	D	D	D	VI	VI	VI	S	-	-	-	-	-	-
$N = 13$	1	VD	D	D	D	D	VI	VI	VI	VI	VI	VI	S	S
	2	VD	VD	D	D	VI	VI	VI	VI	VI	VI	S	S	S

Table 5.2: Inter-cell self-compensation combinations for N = 3, 7, 13 inverter ROs.

in Eqn. (5.6). The two combinations (lowest variation) are shown in Table 5.2 for each N. One can see that each combination contain a similar number of dense and isolated cells with one or more self-compensated cells.

Fabrication Simulations. To simulate fabrication variation of each PUF instance, we choose (1) random defocus values for each RO and (2) random threshold voltages for every transistor (within each inverter).

Random defocus values are chosen using the quad tree model discussed in [49] which models inter-chip and intra-chip correlations. In the quadtree modeling approach [49], the area of a chip is recursively partitioned into four equally sized regions. This is illustrated in Figure 5.5. The first partition is "0-1" and corresponds to the root of the quadtree. "0-1" is divided into four partitions "1-1", "1-2", "1-3", and "1-4" which form the next level of the tree. The four partitions are subdivided into another four partitions and so forth. Each partition in every level of the tree is assigned a random variable (RV) with its own probability distribution. The spatially

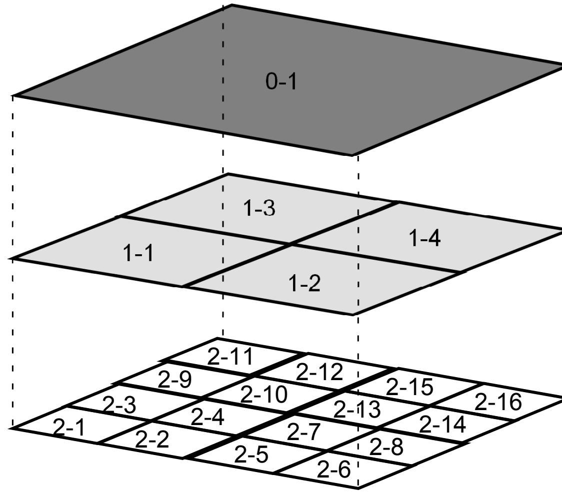


Figure 5.5: Quadtree partitioning for a chip. The depth of the tree shown is 3 levels.

correlated variation associated with a gate in the IC is then defined as the sum of the RV at the lowest partition containing the gate and the RVs of the higher partitions that overlap with the gate's position. Correlation exists between gates on a single chip due to the sharing of RVs at higher levels of the quadtree. Correlation between chips exists because the probability distributions associated with each partition are the same for all chips.

In our case, the random variables (RVs) and associated probability distributions correspond to defocus across the chip. The gates in our case are inverters in the ring oscillators (ROs). We generate RVs for all partitions and levels in each chip/PUF according to some knowns distributions and then compute defocus values as follows. Suppose we want to generate the defocus value F for an RO located in partition "2-13" of chip x (see Figure 5.5). F is computed by summing the RVs associated with partitions "2-13", "1-4", and "0-1" of chip x. In our experiments, we do this for 512 ROs and 100 chips totaling 51200 random samples. The channel

length is determined by the cell type (VD, D, etc.) and the RO's defocus value using the relationships shown in Figure 5.4.

All threshold voltages are selected from a Gaussian distribution with mean .3V and standard deviation of 20mV. With the assigned threshold voltages and channel lengths, we compute RO delay and PUF responses for each chip using Eqns (5.9)-(5.13) and Eqn. (5.4) respectively.

5.4.2 Results and Discussion

For the PUF populations generated above, we estimate the quality (uniqueness, reliability, and unpredictability) of each population as follows. Uniqueness is computed using responses from the PUF population and calculating inter-distance (Eqn. (2.1)). For reliability, we compute PUF responses corresponding to a nominal voltage supply of 1.3 volts. Then, we compute 100 sample responses where a different supply voltage is chosen for each inverter in the RO-PUF from a normal distribution with mean 1.3V and standard deviation 4.4mV. Unpredictabilty is measured using the NIST test suite [73]. We use the same NIST tests as the previous chapter.

Uniqueness (Inter-distance). Mean inter-distance is shown on the left-hand side of Table 5.3. The "standard" rows correspond to the conventional RO-PUF design with non-compensated cell layout and fixed cell type per RO. The "self-compensated" rows are for RO-PUFs using the intra-cell and inter-cell self-compensation approach (cases shown in Table 5.2). The columns correspond to the number of in-

verters N used for all ROs in the PUFs.

For standard layout, the uniqueness measure is much lower than the ideal 50%. The best performing combination among them is "all I" (all iso cells) which in our experiments had the lowest systematic variation in RO frequency (not shown). Self-compensated layout improves on mean inter-distance with case 1 having the largest mean 49.69%. This outcome is not surprising because self-compensation removes systematic variation in chips and bias in PUF responses.

As the number of inverters (N) increases, one would expect that the increase in random variation of the ROs would result in greater uniqueness. However, we found different results for the two approaches. Uniqueness obtained by the standard layout approach worsened with greater N while uniqueness obtained by self-compensated approaches slightly increased. We explain this phenomena as follows. In the standard approach, more "alike" inverters are added into the RO. While this does increase the overall random variation, systematic variation will also increase because the inverters are similarly biased with defocus. This lowers uniqueness. In the self-compensated approach, uniqueness is improved in two ways. First, we increase random variation by adding additional inverters. Second, we have more inverters to compensate with (more tuning variables) and therefore can obtain even lower systematic variation.

Reliability (Intra-distance). The right-hand side of Table 5.3 shows mean intra-distance with random voltage supply variation over the PUF population. The columns and rows are organized as discussed for uniqueness. The ideal reliability occurs when intra-distance is 0% meaning no deviation in PUF responses from the

	Case	Uniqueness N=3	Uniqueness N=7	Uniqueness N=13	Reliability N=3	Reliability N=7	Reliability N=13
standard layout	all VD	43.91	43.86	43.14	4.05	3.31	3.38
	all D	42.25	41.31	39.82	3.98	3.22	3.18
	all VI	42.71	41.42	39.68	3.83	2.82	2.58
	all I	48.07	47.73	47.03	4.64	3.97	3.43
self-compensated	1	49.34	49.55	49.69	5.28	5.01	4.56
	2	49.11	49.36	49.59	5.27	4.99	4.51

Table 5.3: PUF uniqueness and reliability (mean inter- and intra-distance)

response at nominal voltage supply.

For $N = 3$, the standard layout approaches yield better average reliability compared to the self-compensated approaches. We assume this occurs because systematic variation is high for the standard layouts thereby creating a larger gap between RO frequencies/delays that are compared. Intuitively, this means there is a smaller chance of the PUF response bits flipping as a result of environmental variations. The self-compensated layouts have lower systematic variation and are therefore more prone to changes in response.

As N increases, the increase in random variation of the ROs results in better reliability for both standard and self-compensated layouts. On average, from $N = 3$ to 13, there is an average improvement of 25% for the standard layouts and 14% for the self-compensated layouts. The reason that reliability improvements are greater for the standard layouts is as follows. As inverters are added to ROs in the standard layouts, there is an increase in systematic variation (as discussed above for

uniqueness) and with more variation comes greater reliability.

Unpredictability (NIST tests). We found that all test cases passed nearly all the NIST tests for randomness in PUFs. We found that standard layout approaches failed 1 to 3 out of 18 tests while the self-compensated layout approaches only failed 1 out of 18 tests at most. Neither approach seemed to be noticeably affected by extra inverters.

Summary of Results. The results of this section are summarized as follows:

- The proposed self-compensated layout reduces systematic variation and improves PUF uniqueness. Compared to the worst (best) standard layout, self-compensated layouts resulted in 14% (2.6%) better uniqueness.

- Extra inverters have an interesting impact on uniqueness. Adding more "alike" inverters increases systematic variation in the standard layout PUFs (lowering uniqueness). However, the uniqueness improves with additional inverters in the self-compensated PUFs.

- Extra inverters increase random variation in both the standard and self-compensated layout PUFs and improve their reliability by 25% and 14% respectively.

- The NIST tests show that the proposed self-compensated approaches are marginally less predictable than the standard approaches. Unpredictability neither improves nor degrades with extra inverters.

5.5 Summary

In this chapter, we manipulated manufacturing variations in PUFs to simultaneously reduce the impact of systematic variations while enhancing the random variations. The merits of the overall approach in this chapter are as follows:

- *PUF Quality Improvements at Physical Layout Step.* Our self-compensation-based approach is one of the first to improve PUFs at the physical layout level. Specifically, we proposed an optimization framework that creates custom cell layouts for reducing the impact of systematic variations on PUF signatures. Furthermore, since our approach works on the physical layout, it can easily be combined with existing PUF enhancement approaches, such as better circuit designs and post-fabrication processing.

- *Interaction of Layout and Design Approaches.* We showed that by combining our custom cells with naive random variation enhancement solution (extra inverters), it was possible to obtain improvements to PUF uniqueness and reliability. Our work highlights the need to investigate the interaction of PUF circuit layout and design.

Chapter 6

Conclusion and Future Research Directions

In Chapter 1, we highlighted many of the key issues and challenges in hardware security. The IC design/fabrication flow itself is highly susceptible to malicious attacks that threaten to subvert the security and reliability of all systems dependent on ICs. Recent advances in tampering, reverse engineering, etc. also compound the trust issues in computing systems after they are deployed. The main theme of our work has been to address the diversity of such hardware-based attacks through comprehensive strategies that secure hardware platforms during design, fabrication, and post-deployment. Our overall strategy (discussed in Section 1.4) consisted of three phases: *Bootstrap*, *Validate*, *Monitor and React*. In Chapter 2, we discussed two key areas of research in hardware security which could benefit from steps in this strategy: hardware Trojan detection and Physically Unclonable Functions (PUFs).

In Chapter 3, we proposed novel run-time Trojan detection approaches which were basic instances of our comprehensive strategy. Our approaches exploited on-chip thermal sensors which already exist in many modern systems for dynamic thermal management. We statistically characterized IC's power/thermal dynamics to create "golden models" and placed optimally thermal sensors based on these statistics. After fabrication, we gathered information from ICs that pass logic-based and side channel-based detection approaches in order to calibrate each IC for fabrication

variation. The run-time phase integrated the information from the previous phases with thermal sensor measurements to detect Trojan activation. Simulation results using state-of-the-art tools on publicly available Trojan benchmarks verified that our approaches could detect Trojans quickly and with few false positives.

Physically Unclonable Functions (PUFs) rely on IC fabrication variations to generate unique signatures for various hardware security applications. In this dissertation, we highlighted the fact that there has been limited focus on PUF fabrication (source of PUF quality) in prior work and explored 'Bootstrapping' opportunities at mask generation and physical layout levels to overcome this flaw. In Chapter 4, we showed that our SVC-OPC and P-OPC masks outperformed the conventional approach (C-OPC) in at least one of the PUF quality metrics (uniqueness, reliability, unpredictability) without significant loss in others. Being mask-based, neither approach had significant overheads with respect to IC area, power, etc. In Chapter 5, we also discussed a designer-friendly approach for improving PUFs that utilized self-compensated custom cell layouts. This approach resulted in better PUF uniqueness compared to conventional cell layouts and obtained even better results when combined with random variation-enhancing PUF designs.

6.1 Future Research

The approaches presented in this dissertation are very innovative and unconventional compared to existing work. Being among the first to investigate these approaches, there are still opportunities for further improvements as well as some

open issues that exist in defense against hardware Trojan and for improving PUFs.

6.1.1 Defense Against Trojans

The fundamental limits of Trojan detection as well as the overheads of test-time and run-time approaches are still being actively investigated. In this dissertation, we discussed novel run-time approaches that exploited temperature and a comprehensive Trojan detection framework. This research shall act as an important building block for many other directions we plan to pursue in future work:

1. *Further Application of Theoretical Foundations.* The temperature-based technique we proposed relies heavily upon the fundamental foundations provided by state estimation theory and detection theory. There are many open problems in hardware Trojan research that should be able to exploit the same general theory. For instance, existing test-time approaches tend to be ad-hoc and less rigorous in nature. Our Kalman Filter-based approach should naturally extend to various test-time detection schemes that rely on side channels. Furthermore, the KF framework is robust enough to handle multiple side channel modalities for even better results. For example, by combining thermal sensor measurements with current monitors, one can reduce the uncertainty in the power statistics in time, thereby enabling even better thermal state estimation and Trojan detection.

2. *Reaction Mechanisms to Trojan Attacks.* Our work has only focused on the first step of 'Monitor and React.' We showed that it was possible to monitor

temperature to detect Trojans at run-time. However, once a Trojan attack has been detected, one needs to prevent it from doing damage with appropriate defense mechanisms. For instance, one interesting way to respond to attacks is with self-correcting designs that automatically bypass Trojan logic. With the overheads involved with adding such mechanisms and the fact that Trojans may attack the mechanisms themselves, this is a very challenging area of research.

3. *On-chip Tampering of Measurements.* The effectiveness of run-time approaches for Trojan detection depends heavily on the accuracy of on-chip measurements. One way an attacker might circumvent detection is to insert a Trojan in the sensing infrastructure. For example, the Trojan could bypass the real measurements with fake measurements that comply with the expected side-channel behavior. Formal methods that not only detect tampering of sensor measurements, but overcome it are critical for future run-time approaches.

4. *Temperature-based Trojan Detection Prototypes.* We have emphasized that our temperature-based detection approach is effective and low-overhead. However, to our knowledge, the Kalman Filtering framework for temperature tracking has not been physically realized in a prototype as of yet. We feel that a prototype/testbed implementation is an important final step in evaluating our approach. First, since the complicated interaction of hardware, software, and environmental conditions is hard to emulate in simulations, experiments performed with a real-time testbed could provide more accurate results and high-

light impracticalities unforeseen by the applied theory. Second, the implementation would yield a better estimate of the software and hardware overheads involved in our approach.

6.1.2 Opportunities and Challenges in PUF Manufacturing

Our work has been the first to shift the focus of DFM towards improving PUF quality during fabrication. In future, we shall continue to investigate how randomness in PUFs can be improved while still maintaining the spirit of DFM (i.e. manufacturability/yield):

1. *Ways to Manipulate Dose Variation.* While there are two types of variation in IC fabrication (defocus and dose), we have only looked closely at the opportunities for improving sensitivity to defocus variation. However, techniques that manipulate dose variation might provide further enhancements to PUF quality either on their own or combined with our previous techniques. For example, the ASM DoseMapper technology [106] controls exposure dose during IC fabrication and has been successfully used in prior work [107] for better timing and leakage power in the face of process variations. Naturally, it should be possible to combine our objective functions/framework with the DoseMapper technology to improve PUF sensitivity to either systematic or random dose variations.

2. *Transistors With Greater Sensitivity to Random Variations.* Our custom cell layouts reduced the impact of systematic variation on PUFs. Such layout-

based approaches are not only applicable to all different types of PUFs, but they can also be effectively combined with random variation enhancing circuit designs. As an alternative approach, we could go one step further and develop transistor gate, drain, source terminals explicitly for PUFs. New fabrication materials, doping techniques, topologies, etc. that make the underlying transistors more sensitive to independent random variations would be particularly useful since they would effectively compliment our layout and mask-based approaches that reduce systematic variations.

3. *Investigate Manufacturing/Yield Issues.* While the mask-based approaches should not affect the manufacturability of non-PUF portions of ICs, there's still a chance that some of the PUF components will be non-functional. In future work, we would like to investigate the manufacturing/yield issues caused by our approach either by using our masks to fabricate real silicon chips or using more advanced software simulations. If there does happen to be yield issues, we shall investigate ways to overcome them, such as with self-correcting designs [102].

Bibliography

[1] V. Yeo. Hardware-assisted security kills drive to create malware, March 2011. http://www.zdnet.com/hardware-assisted-security-kills-drive-to-create-malware-2062207902/.

[2] S. Skorobogatov. Tamper resistance and physical attacks. *at Summer School on Cryptographic Hardware, Side-Channel and Fault Attacks*, 2006.

[3] ISO/IEC 7816-1:2011 Integrated circuit cards, 2011.

[4] ISO/IEC 11889-1:2009 Trusted Platform Module (TPM), 2009.

[5] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Silicon physical random functions. In *Proc. CCS*, pages 148–160. ACM, 2002.

[6] M. Beaumont, B. Hopkins, and T. Newby. Hardware trojans-prevention, detection, countermeasures (a literature review). Technical report, DTIC Document, 2011.

[7] R. Torrance and D. James. The state-of-the-art in ic reverse engineering. *Proc. CHES*, pages 363–381, 2009.

[8] S. Skorobogatov. Semi-invasive attacks-a new approach to hardware security analysis. *Technical report, University of Cambridge, Computer Laboratory*, 2005.

[9] H. Bar-El. Introduction to side channel attacks. *Discretix Technologies Ltd*, 43, 2003.

[10] I. Verbauwhede and R. Maes. Physically Unclonable Functions: Manufacturing variability as an unclonable device identifier. In *Proc. GLSVLSI*, pages 455–460, 2011.

[11] J. Villasenor. The hacker in your hardware. *Scientific American*, 303(2):82–88, 2010.

[12] D. Forte, C. Bao, and A. Srivastava. Temperature tracking: An innovative run-time approach for hardware trojan detection. 2013.

[13] R. Maes and I. Verbauwhede. Physically Unclonable Functions: A study on the state of the art and future research directions. *Towards Hardware-Intrinsic Security*, pages 3–37, 2010.

[14] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Des. Test*, 27(1):10–25, 2010.

[15] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic. Hardware trojan detection and isolation using current integration and localized current analysis. In *Proc. DFTVS*, pages 87–95. IEEE, 2008.

[16] S.T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou. Designing and implementing malicious hardware. In *Proc. LEET*, pages 1–8. USENIX Association, 2008.

[17] Cyber Security Awareness Week (CSAW) Embedded Systems Challenge. `http://www.poly.edu/csaw2012`.

[18] A. Baumgarten, M. Steffen, M. Clausman, and J. Zambreno. A case study in hardware trojan design and implementation. *Int. J. of Information Security*, 10(1):1–14, 2011.

[19] L. Lin, W. Burleson, and C. Paar. Moles: malicious off-chip leakage enabled by side-channels. In *Proc. ICCAD*, pages 117–122. ACM, 2009.

[20] Y Shiyanovskii, F Wolff, C Papachristou, D Weyer, and W Clay. Exploiting semiconductor properties for hardware trojans. *CoRR*, 2009.

[21] trust HUB.org. `http://trust-hub.org/resources/benchmarks`.

[22] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. Hsiao, J. Plusquellic, and M. Tehranipoor. Protection against hardware trojan attacks: Towards a comprehensive solution. *IEEE Des. Test*, PP(99), 2012.

[23] Inc. Chipworks. `http://www.chipworks.com`.

[24] S. Narasimhan and S. Bhunia. Hardware trojan detection. In Mohammad Tehranipoor and Cliff Wang, editors, *Introduction to Hardware Security and Trust*, pages 339–364. Springer New York, 2012.

[25] Susmit Jha and Sumit Kumar Jha. Randomization based probabilistic approach to detect trojan circuits. In *High Assurance Systems Engineering Symposium, 2008. HASE 2008. 11th IEEE*, pages 117–124. IEEE, 2008.

[26] Rajat Subhra Chakraborty, Somnath Paul, and Swarup Bhunia. On-demand transparency for improving hardware trojan detectability. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 48–50. IEEE, 2008.

[27] Rajat Subhra Chakraborty, Francis Wolff, Somnath Paul, Christos Papachristou, and Swarup Bhunia. Mero: A statistical approach for hardware trojan detection. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 396–410. Springer, 2009.

[28] Dakshi Agrawal, Selcuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, and Berk Sunar. Trojan detection using ic fingerprinting. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 296–310. IEEE, 2007.

[29] Y. Jin and Y. Makris. Hardware trojan detection using path delay fingerprint. In *Proc. HOST*, pages 51–57. IEEE, 2008.

[30] D. Du, S. Narasimhan, R. Chakraborty, and S. Bhunia. Self-referencing: a scalable side-channel approach for hardware trojan detection. *Proc. CHES*, pages 173–187, 2010.

[31] F. Koushanfar and A. Mirhoseini. A unified framework for multimodal submodular integrated circuits trojan detection. *IEEE Trans. Inf. Forensics Security*, 6(1):162–174, 2011.

[32] J. Li and J. Lach. At-speed delay characterization for ic authentication and trojan horse detection. In *Proc. HOST*, pages 8–14. IEEE, 2008.

[33] S. Narasimhan, W. Yueh, X. Wang, S. Mukhopadhyay, and S. Bhunia. Improving ic security against trojan attacks through integration of security monitors. *IEEE Des. Test*, 2012.

[34] T. Bilzor, M.and Huffmire, C. Irvine, and T. Levin. Evaluating security requirements in a general-purpose processor by combining assertion checkers with code coverage. In *Proc. HOST*, pages 49–54. IEEE, 2012.

[35] M. Abramovici and P. Bradley. Integrated circuit security: new threats and solutions. In *Proc. Annual CSIIR Workshop*, page 55. ACM, 2009.

[36] M. Hicks, M. Finnicum, S.T. King, M.M.K. Martin, and J.M. Smith. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *IEEE Symp. Security and Privacy*, pages 159–172, 2010.

[37] C. Sturton, M. Hicks, D. Wagner, and S.T. King. Defeating uci: Building stealthy and malicious hardware. In *Proc. SP*, pages 64–77. IEEE, 2011.

[38] Y. Jin and Y. Makris. Proof carrying-based information flow tracking for data secrecy protection and hardware trust. In *Proc. IEEE VTS*, pages 252–257, 2012.

[39] H. Salmani, M. Tehranipoor, and J. Plusquellic. A novel technique for improving hardware trojan detection and reducing trojan activation time. *IEEE TVLSI*, 20(1):112–125, jan. 2012.

[40] H. Salmani, M. Tehranipoor, and J. Plusquellic. A layout-aware approach for improving localized switching to detect hardware trojans in integrated circuits. In *Proc. WIFS*, pages 1–6, 2010.

[41] G.E. Suh and S. Devadas. Physical Unclonable Functions for device authentication and secret key generation. In *Proc. DAC*, pages 9–14, 2007.

[42] J. Guajardo, S. Kumar, G.J. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *Proc. CHES*, pages 63–80, 2007.

[43] S. Morozov, A. Maiti, and P. Schaumont. An analysis of delay based puf implementations on fpga. *Reconfigurable Computing: Architectures, Tools and Applications*, pages 382–387, 2010.

[44] X. Zhang, N. Tuzzio, and M Tehranipoor. Identification of recovered ics using fingerprints from a light-weight on-chip sensor. In *Proc. DAC*, pages 703–708, June.

[45] B. Moyer. A puf piece: Revealing secrets buried deep within your silicon, January 2011. http://www.techfocusmedia.net/archives/articles/20110124- puf/.

[46] A. Maiti and P. Schaumont. Improved Ring Oscillator PUF: An FPGA-friendly secure primitive. *J. Cryptology*, pages 1–23, 2011.

[47] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.R. Sadeghi, I. Verbauwhede, and C. Wachsmann. PUFs: Myth, fact or busted? A security evaluation of Physically Unclonable Functions (PUFs) cast in silicon. *Proc. CHES*, pages 283–301, 2012.

[48] C.A. Mack. *Fundamental principles of optical lithography: the science of microfabrication*. Wiley-Interscience, 2007.

[49] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. Statistical timing analysis: From basic principles to state of the art. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 27(4):589–607, 2008.

[50] C.E. Yin. A regression-based entropy distiller for RO PUFs. 2011.

[51] M. Orshansky, S.R. Nassif, and D.S. Boning. *Design for manufacturability and statistical design: a constructive approach*. Springer Verlag, 2007.

[52] N. Weste and D. M. Harris. *Principles of CMOS VLSI design: a systems perspective, Fourth Edition*. Addison-Wesley Publishing Company, 2011.

[53] D. Lorenz, G. Georgakos, and U. Schlichtmann. Aging analysis of circuit timing considering nbti and hci. In *Proc. IOLTS*, pages 3–8. IEEE, 2009.

[54] A. Maiti, L. McDougall, and P. Schaumont. The impact of aging on an fpga-based physical unclonable function. In *Proc. FPL*, pages 151–156. IEEE, 2011.

[55] D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 13(10):1200–1205, Oct.

[56] A. Krishna, S. Narasimhan, X. Wang, and S. Bhunia. MECCA: a robust low-overhead PUF using embedded memory array. In *Proc. CHES*, pages 407–420. Springer, 2011.

[57] D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, T. Ochiai, M. Takenaka, and K. Itoh. Uniqueness enhancement of puf responses based on the locations of random outputting rs latches. *Proc. CHES*, pages 390–406, 2011.

[58] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Ruhrmair. The Bistable Ring PUF: A new architecture for strong Physical Unclonable Functions. In *Proc. HOST*, pages 134–141, 2011.

[59] M. Majzoobi, G. Ghiaasi, F. Koushanfar, and S.R. Nassif. Ultra-low power current-based PUF. In *Proc. ISCAS*, pages 2071–2074, 2011.

[60] S.S. Kumar, J. Guajardo, R. Maes, G.J. Schrijen, and P. Tuyls. Extended abstract: The Butterfly PUF protecting IP on every FPGA. In *Proc. HOST*, pages 67–70, 2008.

[61] L. Lin, D. Holcomb, D.K. Krishnappa, P. Shabadi, and W. Burleson. Low-power sub-threshold design of secure Physical Unclonable Functions. In *Proc. ISLPED*, pages 43–48, 2010.

[62] R. Kumar, V.C. Patil, and S. Kundu. Design of unique and reliable Physically Unclonable Functions based on current starved inverter chain. In *Proc. ISVLSI*, pages 224–229, 2011.

[63] V. Vivekraja and L. Nazhandali. Feedback based supply voltage control for temperature variation tolerant PUFs. In *Proc. VLSI Design*, pages 214–219, 2011.

[64] R. Kumar, H.K. Chandrikakutty, and S. Kundu. On improving reliability of delay based Physically Unclonable Functions under temperature variations. In *Proc. HOST*, pages 142 –147, 2011.

[65] C. Bösch, J. Guajardo, A. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient helper data key extractor on fpgas. *Proc. CHES*, pages 181–197, 2008.

[66] M. Yu and S. Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Des. Test Comput.*, 27(1):48–65, 2010.

[67] M. Hiller, D. Merli, F. Stumpf, and G. Sigl. Complementary ibs: Application specific error correction for pufs. In *Proc. HOST*, pages 1–6. IEEE, 2012.

[68] R. Maes, P. Tuyls, and I. Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for sram pufs. *Proc. CHES*, pages 332–347, 2009.

[69] R. Maes, P. Tuyls, and I. Verbauwhede. In *Proc. ISIT, title=A soft decision helper data algorithm for SRAM PUFs, year=2009, pages=2101-2105,.*

[70] M. Bhargava, C. Cakir, and K. Mai. Reliability enhancement of bi-stable pufs in 65nm bulk cmos. In *Proc. HOST*, pages 25–30. IEEE, 2012.

[71] G. Qu and C.E. Yin. Temperature-aware cooperative ring oscillator puf. In *Proc. HOST*, pages 36–42. IEEE, 2009.

[72] M. Majzoobi, F. Koushanfar, and S. Devadas. FPGA PUF using programmable delay lines. In *Proc. WIFS*, pages 1–6, 2010.

[73] A. Rukhin. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, 2001.

[74] D. Forte and A. Srivastava. On improving the uniqueness of silicon-based Physically Unclonable Functions via Optical Proximity Correction. In *Proc. DAC*, pages 96–105, 2012.

[75] D. Forte and A. Srivastava. Manipulating manufacturing variations for better silicon-based physically unclonable functions. In *Proc. ISVLSI)*, pages 171–176. IEEE, 2012.

[76] S. Wei, K. Li, F. Koushanfar, and M. Potkonjak. Hardware trojan horse benchmark via optimal creation and placement of malicious circuitry. In *Proc. DAC*, pages 90–95. ACM, 2012.

[77] K. Hu, A. N. Nowroz, S. Reda, and F. Koushanfar. High-sensitivity hardware trojan detection using multimodal characterization. In *Design, Automation & Test in Europe (DATE)*, Grenoble, France, 03/2013 2013.

[78] J. Srinivasan, S.V. Adve, P. Bose, and J.A. Rivers. The impact of technology scaling on lifetime reliability. In *Proc. DSN*, pages 177–186. IEEE, 2004.

[79] R. Rao, S. Vrudhula, and D.N. Rakhmatov. Battery modeling for energy aware system design. *Computer*, 36(12):77–87, 2003.

[80] K. Skadron, M.R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM TACO*, 1(1):94–125, 2004.

[81] Y. Zhang and A. Srivastava. Accurate temperature estimation using noisy thermal sensors. In *Proc. DAC*, pages 472–477. IEEE, 2009.

[82] Y. Zhang, B. Shi, and A. Srivastava. A statistical framework for designing on-chip thermal sensing infrastructure in nano-scale systems. In *Proc. ISPD*, pages 169–176. ACM, 2010.

[83] S. Sharifi, C. Liu, and T.S. Rosing. Accurate temperature estimation for efficient thermal managemen. *Proc. ISQED*, pages 137–142, March 2008.

[84] Y. Zhang, A. Srivastava, and M. Zahran. On-chip sensor-driven efficient thermal profile estimation algorithms. *ACM Trans. Des. Automat. El.*, 15(3):25, 2010.

[85] Y. Zhang and A. Srivastava. Adaptive and autonomous thermal tracking for high performance computing systems. In *Proc. DAC*, pages 68–73. ACM, 2010.

[86] Y. Zhang and A. Srivastava. Statistical characterization of chip power behavior at post-fabrication stage. In *Proc. IGCC*, pages 1–6. IEEE, 2011.

[87] S.M. Kay. *Fundamentals of Statistical Signal Processing: Detection theory*. Prentice Hall Signal Processing Series. Prentice-Hall PTR, 1998.

[88] D. Forte and A. Srivastava. Energy and thermal-aware video coding via encoder/decoder workload balancing. In *Proc. ISLPED*, pages 207 –212, aug. 2010.

[89] Y. Zhang and A. Srivastava. Leakage-aware kalman filter for accurate temperature tracking. In *Proc. IGCC*, pages 1–7. IEEE, 2011.

[90] J. L. Crassidis and J. L. Junkins. *Optimal Estimation of Dynamic Systems*. CRC Press, 2004.

[91] F. Sebastiano, L.J. Breems, K.A.A. Makinwa, S. Drago, D.M.W. Leenaerts, and B. Nauta. A 1.2 v 10μw npn-based temperature sensor in 65nm cmos with an inaccuracy of $\pm 0.2^o$c from -70°c to 125°c. In *Proc. ISSCC*, pages 312–313. IEEE, 2010.

[92] A. Sreedhar and S. Kundu. Physically Unclonable Functions for embeded security based on lithographic variation. In *Proc. DATE*, pages 1–6, 2011.

[93] N.B. Cobb. *Fast optical and process proximity correction algorithms for integrated circuit manufacturing*. PhD thesis, UC Berkeley, 1998.

[94] P. Yu, S.X. Shi, and D.Z. Pan. True process variation aware Optical Proximity Correction with variational lithography modeling and model calibration. *J. Micro*, 6(3), 2007.

[95] A. Poonawala and P. Milanfar. OPC and PSM design using inverse lithography: A non-linear optimization approach. In *Proc. SPIE*, volume 6154, pages 1159–1172, 2006.

[96] N. Jia and E.Y. Lam. Machine learning for inverse lithography: Using stochastic gradient descent for robust photomask synthesis. *J. Optics*, 12:045601, 2010.

[97] S.H. Teh, C.H. Heng, and A. Tay. Performance-based optical proximity correction methodology. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 29(1):51–64, 2010.

[98] K. Qian and C.J. Spanos. A comprehensive model of process variability for statistical timing optimization. In *Proc. SPIE*, volume 6925, 2008.

[99] A.B. Kahng, S. Muddu, and P. Sharma. Defocus-aware leakage estimation and control. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 27(2):230–240, 2008.

[100] S.X. Shi, P. Yu, and D.Z. Pan. A unified non-rectangular device and circuit simulation model for timing and power. In *Proc. ICCAD*, pages 423–428, 2006.

[101] A.S. Sedra and K.C. Smith. *Microelectronic circuits. 5th Edition*. Oxford University Press, 2004.

[102] L. Jiang, Q. Xu, and B. Eklow. On effective tsv repair for 3d-stacked ics. In *Proc. DATE*, pages 793–798. IEEE, 2012.

[103] P. Gupta, A.B. Kahng, Y. Kim, and D. Sylvester. Self-compensating design for reduction of timing and leakage sensitivity to systematic pattern-dependent variation. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 26(9):1614–1624, 2007.

[104] C.E. Yin. Kendall Syndrome Coding (KSC) for group-based Ring-Oscillator Physical Unclonable Functions. 2011.

[105] M.K. Mandal and B.C. Sarkar. Ring Oscillators: Characteristics and applications. *Indian J. Pure and Applied Physics*, 48:136–145, 2010.

[106] ASML. `http://www.asml.com`.

[107] K. Jeong, A.B. Kahng, C.H. Park, and H. Yao. Dose map and placement co-optimization for improved timing yield and leakage power. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 29(7):1070–1082, 2010.