12-9-2015

# Techniques for Improving Security and Trustworthiness of Integrated Circuits

Kan Xiao
*University of Connecticut - Storrs*, kanxiaocn@gmail.com

# Techniques for Improving Security and Trustworthiness of Integrated Circuits

Kan Xiao, Ph.D.

University of Connecticut, 2015

The integrated circuit (IC) development process is becoming increasingly vulnerable to malicious activities because untrusted parties could be involved in this IC development flow. There are four typical problems that impact the security and trustworthiness of ICs used in military, financial, transportation, or other critical systems: (i) Malicious inclusions and alterations, known as hardware Trojans, can be inserted into a design by modifying the design during GDSII development and fabrication. Hardware Trojans in ICs may cause malfunctions, lower the reliability of ICs, leak confidential information to adversaries or even destroy the system under specifically designed conditions. (ii) The number of circuit-related counterfeiting incidents reported by component manufacturers has increased significantly over the past few years with recycled ICs contributing the largest percentage of the total reported counterfeiting incidents. Since these recycled ICs have been used in the field before, the performance and reliability of such ICs has been degraded by aging effects and harsh recycling process. (iii) Reverse engineering (RE) is process of extracting a circuits gate-level netlist, and/or inferring its functionality. The RE causes threats to the design because attackers can steal and pirate a design (IP piracy), identify the device technology, or facilitate other hardware attacks. (iv) Traditional tools for uniquely identifying devices are vulnerable to non-invasive or invasive physical attacks. Securing the ID/key is of utmost importance since leakage of even a single device ID/key could be exploited by an adversary to hack other devices or produce pirated devices. In this work, we have developed a series of design and test methodologies to deal with these four challenging issues and thus enhance the security, trustworthiness and reliability of ICs. The techniques proposed in this thesis include: a path delay fingerprinting technique for detection of hardware Trojans, recycled ICs, and other types counterfeit ICs including remarked, overproduced, and cloned ICs with their unique identifiers; a Built-In Self-Authentication (BISA) technique to prevent hardware Trojan insertions by untrusted fabrication facilities; an efficient and secure split manufacturing via Obfuscated Built-In Self-Authentication (OBISA) technique to prevent reverse engineering by untrusted fabrication facilities; and a novel bit selection approach for obtaining the most reliable bits for SRAM-based physical unclonable function (PUF) across environmental conditions and silicon aging effects.

# Techniques for Improving Security and Trustworthiness of Integrated Circuits

Kan Xiao

B.E., Beijing University of Posts and Telecommunications, Beijing, China, 2009

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2015

# APPROVAL PAGE

Doctor of Philosophy Dissertation

## Techniques for Improving Security and Trustworthiness of Integrated Circuits

Presented by

Kan Xiao, B.E.

Major Advisor

________________________________________

Mark Tehranipoor

Associate Advisor

________________________________________

John Chandy

Associate Advisor

________________________________________

Domenic Forte

Associate Advisor

________________________________________

Associate Advisor

________________________________________

University of Connecticut

2015

Dedicated to my dearest family.

# ACKNOWLEDGEMENTS

I am very grateful to my advisor, Dr. Mark M. Tehranipoor, for his support and guidance during my Ph.D study at the University of Connecticut. Thanks to his encouragement and inspiring suggestion whenever I feel frustrated. He not only has taught me academic knowledge, analytical thinking techniques, and written and presentation skills, but also has set an example of excellence as a researcher, mentor, and entrepreneur. In addition, I also would like to express my gratitude to Dr. Domenic Forte for his insightful guidance and patient instruction over the past two years. What I learned from them would be very helpful for my career development.

Furthermore, I sincerely thank Dr. John Chandy, Dr. Domenic Forte, Dr. Lei Wang, and Dr. Mehdi Anwar for joining my Advisory Committees and providing constructive feedback.

My deep gratitude and appreciation also go to Mr. Doug Dreibelbis and Mr. Harry Linzer for their help during my internship at IBM Microelectronic Division. In addition, I would like to thank Mei Su and Yu Huang from Synokey LLC for their sponsorship and cooperation in one of my research projects.

Special thanks to all my labmates, including Xuehui Zhang, Wei Zhao, Jifeng Chen, Shuo Wang, Hassan Salmani, Nick Tuzzio, Niranjan Kayam, Qihang Shi, Ujjwal Guin, Kun Yang, Gustavo Contreras, Md Tauhidur Rahman, Adib Nahiyan, Miao He, Mehdi Sadi, Fahim Rahman, and Halit Dogan for their help and cooperation. The discussions and exchanges of knowledge enriched my skills and experience. I am also deeply indebted to my teachers, mentors, classmates and friends in my life. There are so many of them that I cannot list all their names.

I would like to appreciate my family for their love, encouragement, support, understanding, and protection throughout my life. My wonderful family inspired my drive and ability to tackle challenges head on. It is my greatest honor to dedicate this work to them.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

# Introduction

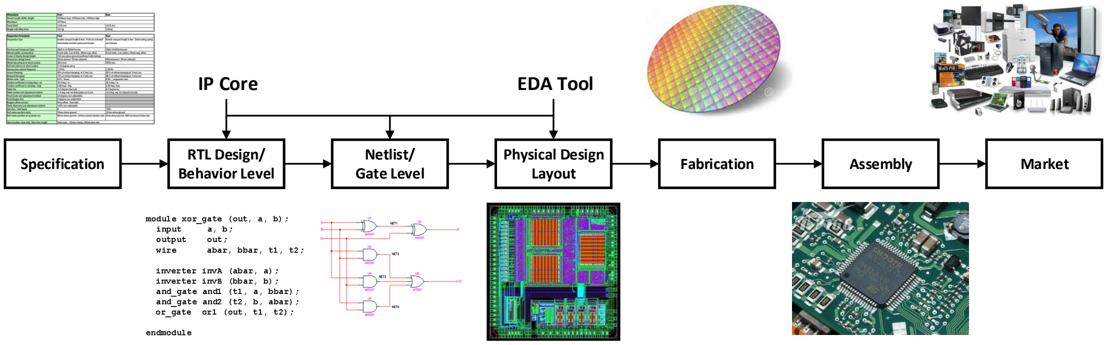
With the emergence of information technology and its critical role in our daily lives, the risk of cyber attacks is larger today than ever before. Many security systems or devices have critical assurance requirement. Their failure may endanger human life and environment (as with military and transportation system), do serious damage to major finanical infrastructure, endanger personal privacy, and undermine the viability of whole business sectors (cable service). Even the perception that a system is more vulnerable than it really is (paying with a credit card over the Internet) can significantly impede economic development. Information security engineering focuses more on the defense against intrusion and unauthorized use of resources with software in the past, such as antivirus, firewall, security information management, virtualization, cryptographic software, and security protocol. While the battle between software developers and hackers has raged since the 1980s, the underlying hardware was generally considered safe, though not perfectly reliable.

However, in the last decade or so, this assumption is increasingly questionable. The battle field extends to hardware domain because more attacks on hardware are discovered and they are shown to be more effective and efficient than traditional software attacks. Additionally, the complexity of the design, fabrication, and distribution of electronics has caused a shift throughout the industry towards a global business model. In such a model, untrusted entities participate either directly or indirectly in all phases in the life of an electronic device or integrated circuit (IC), which provides more potential opportunities for adversaries to perform their attacks. The use of untrusted (and potentially malicious) third parties into the development flow also increases the security concerns as designs and devices pass through deeper supply chains. Therefore, the IC development supply chain is now considered susceptible to various attacks, such as hardware Trojan attacks, IC tampering, reverse-engineering, intellectual property (IP) piracy, counterfeiting, and so forth. The security concerns we focus on in this thesis will be presented in the rest of this chapter.

## 1.1 Security Issues

### 1.1.1 Hardware Trojan

With semiconductor scaling to very deep submicron levels, the complexity and cost of IC design and fabrication increase dramatically. An ASIC/SoC component typically will go through a process as shown in Figure 1.1. This development flow comprises of steps from design and manufacturing all the way to sale in the market. The first step of the process is the translation of the specifications into a behavioral description, typically in a hardware design language (HDL) such as Verilog or VHDL. Next, synthesis is performed to transform the behavioral description into a design implementation in terms of logic gates (i.e., netlist). After implementing the netlist as a layout design, the digital GDSII files are then handed to a foundry for IC fabrication. Once the foundry produces the actual ICs, the testing step ensures their correct operations. Those ICs that pass testing are packaged by assembly and, finally, the electronics components head to the market for sale and eventually deployment on systems. Because of the globalization of semiconductor industry, this current semiconductor supply chain is vulnerable to hardware Trojans. In 2008, [1] reported that a critical failure in Syrian radar might have been intentionally triggered through a "backdoor" hidden within a commercial off-the-shelf microprocessor. According to a U.S. defense contractor who spoke on condition of anonymity, a "European chip maker" recently built such microprocessors with remote kill switches for just such purposes. In another instance, [2] reported detection of a backdoor in a military grade FPGAs. Given the dire consequences associated with such weaknesses, the so called "hardware Trojan" issue has received considerable attention from both academia, industry, and government over the last decade.



```
module xor_gate (out, a, b);
  input     a, b;
  output    out;
  wire      abar, bbar, t1, t2;

  inverter invA (abar, a);
  inverter invB (bbar, b);
  and_gate and1 (t1, a, bbar);
  and_gate and2 (t2, b, abar);
  or_gate  or1 (out, t1, t2);

endmodule
```

**Fig. 1.1:** An ASIC/SoC development flow.

A hardware Trojan is defined as a malicious, undesired, intentional modification of an electronic circuit or design that results in undesired behavior when the circuit is deployed [5]. ICs that are "infected" by a hardware Trojan may experience modifications to their functionality or specification, may leak sensitive information, may experience degraded or unreliable performance, or may be more susceptible to Denial of Service (DoS) and backdoors. Hardware Trojans can be inserted at any phase from circuit design to fabrication. Several articles have proposed detailed hardware Trojan taxonomies. For instance, [4] [5] separates Trojans based on five different attributes: insertion phase, abstraction level, activation mechanism, effects, and location.

The hardware Trojan research domain has seen significant progress over the past decade. Various types of hardware Trojans have been developed and investigated. Regardless of combinational, sequential or other emerging hardware Trojans, in general, a Trojan contains two basic parts: trigger and payload [6]. Figure 1.2 shows a basic Trojan architecture. Trojan trigger is an optional part which monitors various signals and/or a series of events in the circuit. The payload usually taps signals from original (Trojan-free) circuit and the output of the trigger. Once Trojan trigger detects an expected event or condition, the payload is activated to perform malicious behaviors. Typically, the trigger is expected to be activated under extremely rare conditions, so the payload remains inactive most of the time. When the payload is inactive, the IC acts like Trojan-free circuit making it difficult to detect the Trojan. In the recent years, researchers have explored and investigated new Trojan triggers and payloads that could increase difficulty of activation and detection of Trojans. These triggers may utilize don't care states in a design [7] or silicon wear-out mechanisms [8] [9] for Trojan activation, while payloads might generate intentional side-channel signals to leak secret information [10] without impacting primary outputs. Since extra circuitry introduced by Trojan trigger and payload inevitably causes some side-effects, such as additional area, timing, power or radiation, they could be utilized by defenders for Trojan detection. Thus, to make their Trojan more stealthy and avoid being detected, Trojan designs could be further optimized and thus minimize Trojan impact on the original design as much as possible [11] [12].

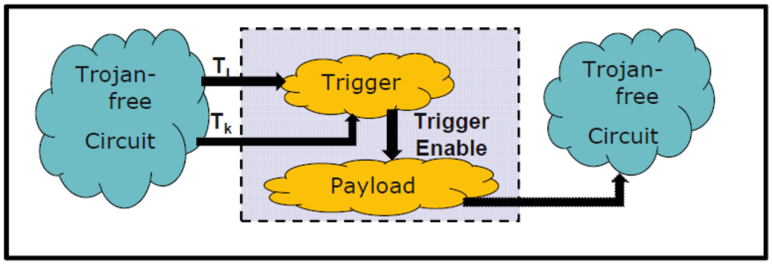


**Fig. 1.2:** Trojan structure.

### 1.1.2 Recycled ICs

The counterfeiting of semiconductor components has been on the rise for many years as a result of several vulnerabilities in the electronics component supply chain. According to one estimate, the United States Department of Defense may have purchased between $15-100M USD worth of counterfeit ICs in 2005 alone [13]. The most recent data provided by Information Handing Service Inc. (IHS) shows that reports of counterfeit ICs have quadrupled since 2009 [14]. If counterfeit ICs were to end up in the supply chain for mission-critical or life-saving applications, the results of the failure of an unreliable or insecure counterfeit part could be catastrophic.

According to the definition of counterfeit electronics in [17], a counterfeit component (*i*) is an unauthorized copy; (*ii*) does not conform to original component manufac-

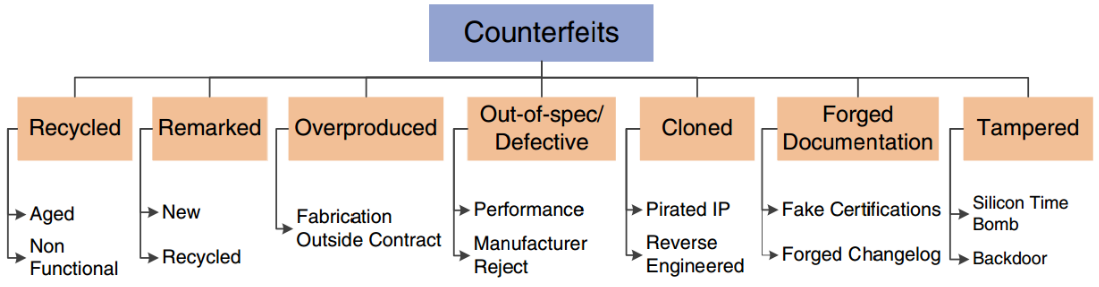


**Fig. 1.3:** A taxonomy of counterfeit component types.

turer (OCM) design, model, and/or performance standards; (*iii*) is not produced by the OCM or is produced by unauthorized contractors; (*iv*) is an off-specification, defective, or used OCM product sold as "new" or working; or (*v*) has incorrect or false markings and/or documentation. The above definition does not include all possible scenarios where an entity in the component supply chain source electronic components that are authentic and certified by the OCMs. Guin et al developed a comprehensive taxonomy of counterfeit types [18–21]. They classified the counterfeit types into seven distinct categories: recycled, remarked, overproduced, out-of-spec/defective, cloned, forged documentation, and tampered ICs, as shown in Figure 1.3. Guin et al also presented all possible vulnerabilities from design to resign of ICs in the supply chain [22,23].

The category that has contributed the most to the rise of counterfeits is the recycled ICs. It is estimated that these recycled ICs account for 80% of all counterfeits being sold worldwide. A report from the Office of Technology Evaluation, part of the U.S. Department of Commerce, also proves that the number of reported incidents of used ICs being sold as new or remarked as higher grade is larger than other types of counterfeits, as shown in Figure 1.4 [16,17]. In addition, electronics consumer and e-

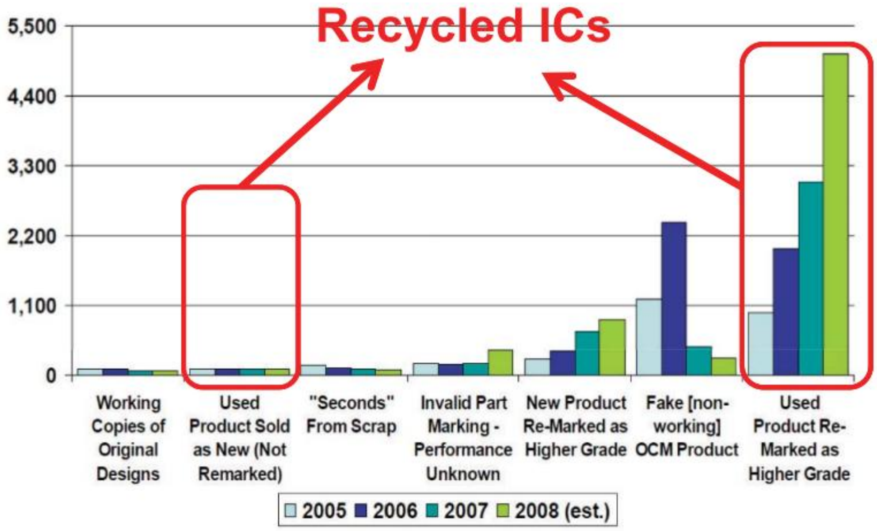


**Fig. 1.4:** Counterfeit incidents by type of problem for microcircuits from 2005 to 2008.

waste trends suggest that this recycling is only going to increase over time as more gadgets are used for shorter periods of time [17]. These used or defective ICs enter the market when electronic “recyclers” divert scrapped circuit boards away from their designated place of disposal for the purposes of removing and reselling the ICs on those boards. After carefully cleaning and remarking, those used ICs look like new and could be re-used in critical applications. However, the used components have been deployed in a system and experienced aging degradations comparing with new components [24,82]. Additionally, the recycling process usually involves a high temperature environment to remove ICs from boards, so recycled ICs could fail sooner and be less reliable than new chips. It is vital to identify and prevent recycled ICs from entering critical infrastructure, aerospace, medical, and defense supply chains.

### 1.1.3 Reverse Engineering

Reverse engineering (RE) is a process that attempts to gain a full understanding of a circuit's construction and/or functionality. Thus, it can be used to clone, pirate or counterfeit a design, to develop an attack, or insert a hardware Trojan by adversaries. As described in section 1.1.1, multiple untrusted parties could be involved in design, fabrication, assembly and distribution of today's ICs, so this long and globally distributed semiconductor supply chain becomes more vulnerable to RE attacks. IC RE can occur at various development stages. However, reverse engineering a packaged chip is the most difficult, costly and time-consuming. A chip has package material to protect the die and each die has several metal layers, vias, interconnections, passivation, and active layers. Different destructive and non-destructive RE techniques have been developed. A destructive process usually produces more exact results, but the IC is no longer usable; most nondestructive methods are less exact in determining precisely what is going on inside the chip, but the IC is still usable afterwards [25].

- **Non-destructive RE**: There are two categories of non-destructive methods of reverse engineering: those methods which passively view the IC without interfering in any way (external), and those which actively interface with the accepted I/O pins to explore the inner workings of the IC (internal) [26]. Both methods have their advantages and drawbacks; the external, passive methods only work to find differences from a baseline and may not have the level of granularity needed, but the more exact internal, active methods are very slow. These methods are often known as characterizing the IC. For example, X-ray tomography is an external non-destructive method, which can provide layer-by-layer images of chips manufactured in older technology nodes and is often used for the analysis of internal vias, traces, wire bonding, capacitors, contacts, or resistors.

- **Destructive RE**: The basic destructive method for exploring an IC is relatively simple in design, but difficult to execute with the required precision. This method physically removes micrometer thin layers of the IC, paring it down slowly and taking pictures with an electron microscope at each layer. Using those images, it is then possible to reconstruct the physical layout of the chip and work upwards to the full chip design. With the right tools and techniques, this is a very accurate and exact method even for modern process nodes; it is also destructive to the extreme [25].

As the feature size of transistor shrinks further, RE at chip level is becoming more difficult regardless of non-destructive and destructive REs. Therefore, RE at the fabrication phase in foundry could be a better opportunity for attackers, because the information for all layers are available on the masks. Attackers can easily extract its gate-level netlist from the lithographic masks and even could infer its functionality. We try to tackle the RE problem occurring at the fabrication stage by untrusted foundry in this proposal. Novel approaches are needed to prevent RE with a low cost.

### 1.1.4 On-Chip ID/Key Generation

Traditional tools for uniquely identifying devices, such as product IDs and barcodes, are exposed to everyone and hence can be easily duplicated. Alternatively, an intrinsic device identifier (ID) can be digitally stored in the non-volatile memory (NVM) or fuses. It might be also applicable to ID or cryptographic key storage. Storing the ID/key using these methods is not only expensive, but also vulnerable to physical attacks (such as non-invasive, semi-invasive, or invasive) as well as software attacks (such as viruses and API-attacks [27]). Securing the ID/key is of utmost importance since leakage of even a single device ID/key could be exploited by an adversary to hack other devices or produce pirated devices.

Physical Unclonable Functions (PUFs) have been proposed as a more secure alternative to conventional ID/key storage because of their unclonability and built-in tamper resistance features [28]. Each PUF is able to generate a unique ID by exploiting the inherent uncontrollable random variations in manufacturing process (e.g. threshold voltage ($V_{th}$), channel length ($L$), and oxide thickness ($T_{ox}$)). PUFs avoid the high cost of building tamper-resistant NVM system, since any invasive attack may alter internal behavior of an IC leading to incorrect outputs. An ideal PUF takes an input (challenge) and gives a random output (response) which is unique for every device [30]. Another advantage of PUF is that a PUF can produce a large of amount of challenge-response pairs that are random and usually difficult to predict, which overcomes the limitation of insufficient number of key storage. Besides regular authentication and identification, PUFs can be also employed to detect cloned, overproduced, and remarked ICs, as discussed in Section 1.1.2, because a new or different IC will produce a distinct PUF output.

## 1.2 Previous Works and Limitations

### 1.2.1 Hardware Trojan Detection

A large variety of potential hardware Trojans are man-made and designed to be stealthy by intelligent adversaries, which is a major difference from manufacturing defects that have been extensively researched for the last few decades. Manufacturing defects are natural and unintentional, and the behaviors of such defects could be reflected with stuck-at fault, delay fault, etc. models. Automatic test pattern generation (ATPG) tools can generate structural patterns to detect certain faults. For hardware Trojans, it is impossible to create a model that fits all types of Trojans. Additionally, manufacturing defects are only produced during the fabrication process, while hardware Trojans could even be injected during design stages. Hence, there may not exist an uninfected (i.e., Trojan-free) design to even compare newly fabricated ICs with.

Due to the limitations of traditional manufacturing tests for Trojan detection, various hardware Trojan countermeasures have been developed, in order to address or mitigate potential hardware Trojan threats in the vulnerable semiconductor supply chain. Generally, they are classified into two categories, Trojan detection and design-for-trust, as shown in Figure 1.5.

#### Trojan Detection

Trojan detection is the most straightforward and commonly-used way to deal with hardware Trojans. It aims to verify the existing designs and fabricated ICs without any supplementary circuitry. Generally, existing Trojan detection approaches are performed

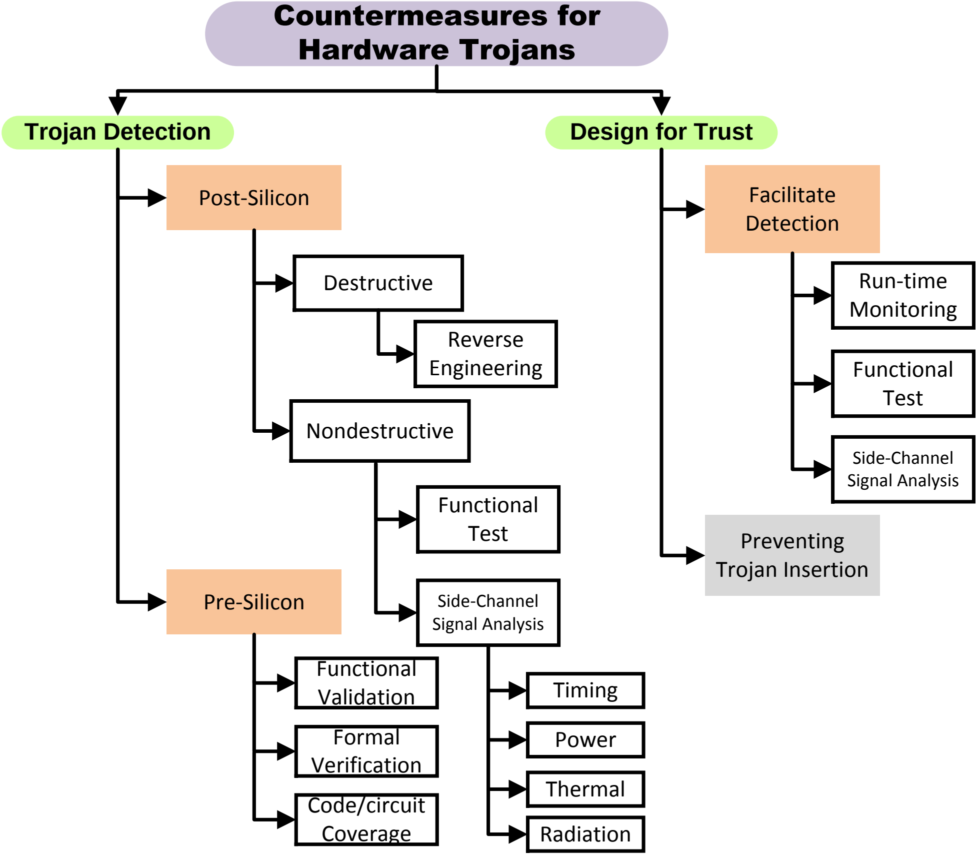


**Fig. 1.5:** The taxonomy of Hardware Trojan countermeasures.

either at the design stage (i.e., pre-silicon) to validate IC designs or after manufacturing stage (i.e., post-silicon) to verify fabricated ICs.

**Post-silicon detection techniques** can be further divided into destructive and nondestructive methods, as illustrated in Figure 1.5. *Destructive* methods typically use destructive reverse-engineering techniques to depackage an IC and obtain images of each layer in order to reconstruct the design for trust validation of the end product. Provided a golden design is available, destructive reverse-engineering has the potential of giving 100% assurance that any malicious modification in the IC will be detected, but it is high cost and could take several weeks and months to do this for an IC of reasonable complexity. Additionally, at the end of this invasive process, the IC cannot be used, and we only get the information for a single IC. Hence, in general, destructive approaches are not considered viable for Trojan detection. However, destructive reverse engineering on a limited number of samples can be attractive in order to obtain the characteristics of a golden batch of ICs. This information can be useful for trust validation through side-channel analysis. [33] proposed to adapt a well-studied machine learning method, one-class support vector machine (SVM), to identify Trojan-free ICs for the golden model.

*Nondestructive* techniques try to authenticate fabricated ICs from untrusted foundry through functional tests or side-channel signal analysis:

(1) Functional test needs to activate Trojans by applying test vectors and comparing the responses with the correct results, like manufacturing tests for detecting manufacturing defects. However, hardware Trojans result in additional functionality or modifications to the original design. Unlike manufacturing defects whose behaviors can be predicted, conventional manufacturing tests using functional/structural/random patterns perform poorly to reliably detect hardware Trojans [34]. Intelligent adversaries can design Trojans that are activated under very rare conditions, so they can go undetected under structural and functional tests during manufacturing test process. [35] [36] developed test pattern generation methods to trigger such rarely activated nets and improve the possibility of observing Trojan's effects from primary outputs. However, due to the numerous logical states in a circuit, it is impractical to enumerate all

states of a real design. Additionally, instead of changing the functionality of the original circuit [37], a Trojan may transmit information, e.g., with an antenna, or modify the specification. Functional tests fail to detect this kind of Trojans.

(2) Side-channel signal analysis approaches are able to detect hardware Trojans by measuring circuit parameters, such as delay [6] [38], power (transient [39] and leakage power [40]), temperature [41], and radiation [42]. They take advantage of side effects (i.e. extra path delay, power, heat or electromagnetic radiation) caused by additional circuits and/or activity from Trojan trigger/payload activation. However, most of these techniques assume "golden ICs" (Trojan-free ICs) are available for comparison, in order to identify Trojan-infected ICs. In addition, while side-channel analysis methods may succeed in detecting Trojans to some degree, the difficulty lies in achieving high coverage of every gate or net and in extracting the tiny, abnormal side-channel signals of hardware Trojans in the presence of process and environmental variations. As the feature size of ICs shrinks and the number of transistors grows, the increasing levels of process variations can easily mask the small side-channel signals induced by low-overhead and rarely-triggered Trojans.

**Pre-silicon Trojan detection techniques** focus on a different attack model, which are used to help SoC developers and design engineers to validate third party IP (3PIP) cores and their final designs, since hardware Trojan could be added into 3PIP cores by untrusted IP vendors, into designs by untrusted EDA tools or rogue employees, or both. Existing pre-silicon detection techniques can be broadly classified into functional validation, code/structural analysis, and formal verification.

(1) The principle idea of functional validation is the same as the functional tests described above. The functional validation is conducted with simulation, while functional tests have to be performed on a tester for applying input patterns and collecting output responses. Therefore, existing techniques for functional tests are also applicable to functional validation. Of course, function validation also inherits functional tests' pros and cons.

(2) Designs are typically described in behavioral or structural code in hardware design languages. Code analysis is performed on behavioral [43] or structural [44] codes to identify redundant statements or circuits that may be a part of a Trojan. On the other hand, structural analysis employs some quantitative metrics to mark signals or gates with low activation probability as suspicious [45] [46]. Additionally, [47] attempts to identify the vulnerabilities by extracting Trojan features from several existing Trojan benchmark circuits. The limitations of code/structural analysis techniques are that they do not guarantee Trojan detection, and manual post-processing is required to analyze suspicious signals or gates and determine if they are a part of a Trojan.

(3) Formal verification is an algorithmic-based approach to logic verification that exhaustively proves a pre-defined set of security properties that a design should satisfy [43] [48]. To check if a design honors these properties, one converts the target design into a proof checking format (e.g. Coq) [49]. However, formal verification techniques could fail to detect additional unexpected functionality introduced by Trojans while satisfying these properties.

### Design-for-Trust

Although functional validation techniques could potentially activate a Trojan and detect it, intelligent adversaries can design Trojans that are activated under very rare conditions, so they can go undetected under structural and functional tests during the post-silicon validation. In addition, detecting a quiet, low-overhead hardware Trojan is still very challenging with existing side-channel analysis techniques. A more effective way is to plan for the Trojan issue in the design phase through design-for-trust. The basic way of Design-for-Trust (DfT) approaches is to facilitate the existing detection approaches discussed above:

(1) **Facilitate Functional Test**: Triggering a Trojan from inputs and observing the Trojan effect from outputs are difficult due to the stealthy nature of Trojans. A large number of low-controllable and low-observable nets in a design significantly hinder the possibility of activating a Trojan. [50] [51] attempt to increase controllability and observability of nodes by inserting test points into the circuit. Another approach proposes to multiplex two outputs of a DFF, $Q$ and $\bar{Q}$, through a 2-to-1 multiplexer and select either of them, in order to extend the state space of the design and increase the possibility of exciting/propagating the Trojan effects to circuit outputs and making them detectable [35]. These approaches are beneficial not only to functional-test based detection techniques, but also to side-channel based methods that need partial activation of Trojan circuitry.

(2) **Facilitate Side-Channel Signal Analysis**: A number of design methods have been developed to increase the sensitivity of side-channel based detection ap-

proaches. [52] propose to minimize background side-channel signals by localizing switching activities within one region while minimize it in other regions through a scan-cell reordering technique. Additionally, some newly-developed structures or sensors are implemented in the circuit to provide a higher detection sensitivity than conventional measurements. Ring oscillators (ROs) structure [53], shadow registers [54] and delay elements [55] on a set of selected short paths are inserted for path delay measurements. RO sensors [56] and transient current sensors [57] [58] are able to improve sensitivity to voltage and current fluctuations caused by Trojans, respectively. Besides, integration of process variation sensors can calibrate the model or measurement and minimize the noise induced by manufacturing variations [59] [60].

(3) **Run-time Monitoring**: As triggering all types and sizes of Trojans during pre-silicon and post-silicon tests are very difficult, run-time monitoring of critical computations can signicantly increase the level of trust with respect to hardware Trojan attacks. These run-time monitoring approaches can utilize existing or supplemental on-chip structures or sensors to monitor chip's behaviors [61] [62] or operating conditions, such as transient power [57] [63] and temperature [41]. They can disable the chip upon detection of any abnormalities or bypass it to allow reliable operation, albeit with some performance overhead. [64] presents a design of an on-chip analog neural network which can be trained to distinguish trusted from untrusted circuit functionality based on measurements obtained via on-chip measurement acquisition sensors.

For the DfT techniques that require circuitry added in the front-end design phase, the potential area and performance overheads are the chief concerns to designers. As the size of a circuit increases, the number of quiet (low controllability/observability) nets/gates will increase the complexity of processing and produce a large time/area overhead. Thus, the DfT techniques for facilitating detection are still difficult to apply to a large design that contains millions of gates. Therefore, a more effective DfT technique with low-overhead is needed to detect or even prevent hardware Trojans in ICs.

### 1.2.2 Recycled IC Identification

Existing techniques for recycled IC detection are limited. Most recycled ICs are detected through careful physical visual inspection, blacktop testing, scanning electron microscopy (SEM), scanning acoustic microscopy, X-ray imaging, X-ray fluorescence, and so forth, since the markings or parts of the package may have been damaged during the refining process. These methods can effectively detect recycled ICs with gross defects, such as defects in package, lead, bond wires, die, and etc. However, they cannot detect recycled ICs without these physical defects. Moreover, all the ICs under physical test cannot be verified as most of these tests are based on sampling and decapping. DNA markings are commercially available to detect recycled ICs, but it is extremely time-consuming and costly, which makes it impractical for recycled part detection. Alternatively, electrical detection methods, such as functional tests and parameter tests, could be applied to all ICs under test. However, the applicability and cost of these tests to today's complex ICs are a major concern.

Several IC design approaches have been proposed to identify recycled ICs [82] [83]. [82] uses a light-weight on-chip sensor that avoids the data collection altogether and applies a "self-referencing" concept to the measurement of use time. However, this approach cannot address detection of existing and legacy ICs that have no such sensors embedded in them. Therefore, new electrical test techniques are needed to measure ICs' specifications and detect recycled ICs without changing the original design, so that they can be used on both new IC designs and legacy designs already in production or in use.

### 1.2.3 Anti-Reverse Engineering

The issues associated with RE include security risks to critical systems, profit loss for intellectual property owners, and the discouragement of innovation in hardware development. Anti-reverse engineering becomes important. Several techniques have been proposed in the past to hinder attackers from extracting original circuit structure and/or functionality. Logic obfuscation (or logic encryption) and camouflaging techniques are two major methodologies to overcome RE attacks.

Logic obfuscation attempts to hide the functionality and the implementation of a design by insertion of built-in locking mechanisms into the original design. The locking circuits become transparent and the right function appears only when a valid key is applied. The increased complexity of identifying the genuine functionality without knowing the right input vectors is able to dwarf the ability of inserting a targeted Trojan by attackers. For combinational logic obfuscation, XOR/XNOR gates could be introduced at certain locations in a design [65]. In sequential logic obfuscation, additional states are introduced in a finite state machine to conceal its functional states [67]. In addition, some papers proposed to insert reconfigurable logics for logic obfuscation [69] [70] [71]. The design is functional when the reconfigurable circuits are correctly programmed by the design house or end user.

Camouflaging is a layout-level obfuscation technique to create indistinguishable layouts for different gates by adding dummy contacts and faking connections between the layers within a camouflaged logic gate [72] [73]. The camouflaging technique can hinder attackers from extracting a correct gate-level netlist of a circuit from the layout through imaging different layers, so the original design is protected from inserted targeted Trojans. Additionally, [74] utilized the same concept of dummy contact and developed a set of camouflaging cells using the polarity controllable SiNW FETs.

However, these techniques require additional circuit or design efforts in the design phase (RTL or gate-level). The potential area and performance overheads are the chief concerns to IC designers. Thus, the existing techniques are still questionable to apply to a large design that contains millions of gates. In addition, these techniques need to insert additional gates (logic obfuscation) or modify the original standard cells (camouflaging), which could degrade the chip performance significantly and affect their acceptability in high-end circuits. Therefore, a low-cost and effective technique is necessary today to prevent RE at fabrication phase by untrusted foundry.

### 1.2.4 Physical Unclonable Function

Many different kinds of PUFs have been proposed and implemented over the past decade, such as Arbiter PUF [31], RO-PUF [30], SRAM PUF [32] [119], Butterfly PUF [32], etc. A majority of PUFs require additional dedicated circuit structures that need absolute symmetry. Apart from the substantial cost in silicon area, their integration into an SoC design needs extra effort on the placement and routing in layout design, which limit their acceptability significantly. Therefore, a reliable PUF technique without additional hardware is essential. On the other hand, a separate class of relatively few PUF implementations generates ID/key from existing on-chip structures, such PUFs that exploit random mismatch in inner node voltages of memory elements (e.g. SRAM or Flip-Flops). Memory based PUFs are a good choice for those SoCs that integrate SRAMs inside. However, the power-up value is influenced by environmental variations. Exhaustive tests across different temperatures/voltages used in prior works are expensive and time-consuming. A more effective and low-cost solution is needed. In addition, a general and reliable ID/key generation technique is important for ICs that are already in production (including legacy designs) and have no memory. The generated on-chip IDs can also be used to detect counterfeit ICs, including recycled ICs, remarked ICs, cloned ICs, and overproduced ICs.

## 1.3 Contributions

With the above motivation, this thesis is devoted to the development of a series of low-cost and effective techniques for secure and trustworthy integrated circuits (ICs). All the security issues discussed above will be dealt with from three angles: pre-silicon test methodologies, design-for-security methodologies, and security primitive enhancement methodologies. The overview of my proposed solutions is shown in Figure 1.6.

### 1.3.1 Post-silicon Test Methodologies

Nowadays post-silicon test is a common practice to identify defective ICs that are caused by uncontrollable physical imperfections during manufacturing process. The test procedure is based on design specification and fault model associated with the implementation

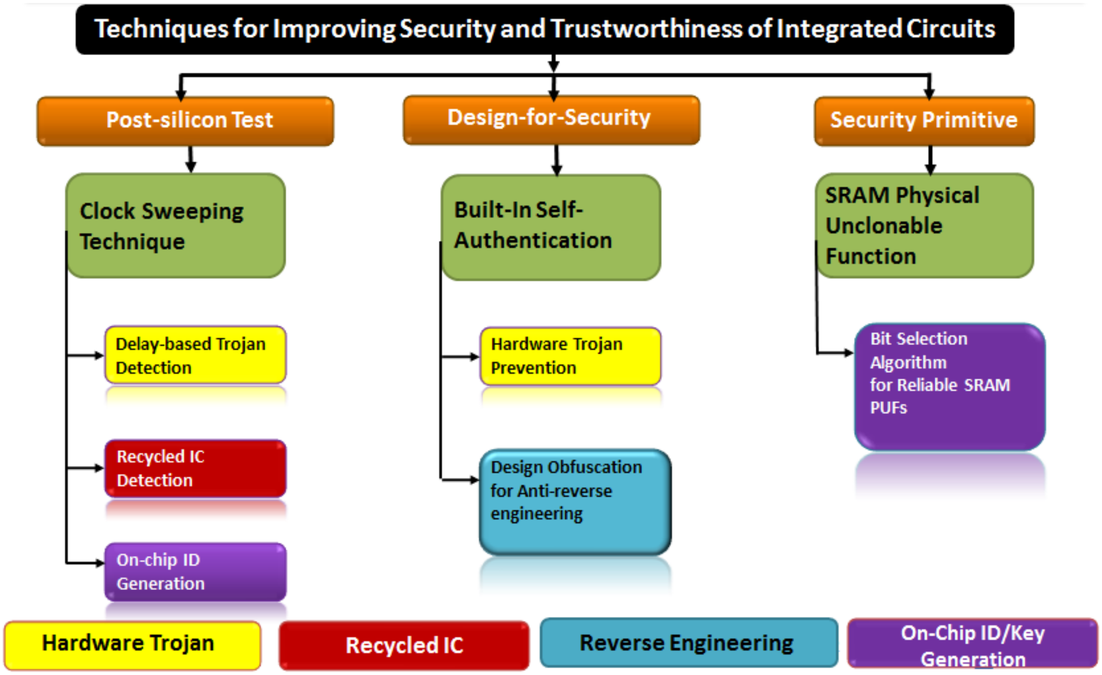

Techniques for Improving Security and Trustworthiness of Integrated Circuits
Post-silicon Test
Design-for-Security
Security Primitive
Clock Sweeping Technique
Built-In Self-Authentication
SRAM Physical Unclonable Function
Delay-based Trojan Detection
Recycled IC Detection
On-chip ID Generation
Hardware Trojan Prevention
Design Obfuscation for Anti-reverse engineering
Bit Selection Algorithm for Reliable SRAM PUFs
Hardware Trojan
Recycled IC
Reverse Engineering
On-Chip ID/Key Generation

**Fig. 1.6:** The overview of the proposed solutions in this dissertation.

technology, so security issues have not been taken into account when test engineers develop the test procedure. A new delay-based IC fingerprinting is proposed, based on conventional delay tests, to detect hardware Trojans, identify recycled ICs, and generate secure and reliable on-chip IDs/keys without any area overhead, and without using uncommon steps in the design or test processes. A clock sweeping technique will be presented to extract intrinsic path delay information and produce IC fingerprints.

### Hardware Trojan Detection

Compared to full Trojan activation and other side-channel signal techniques described in Section 1.2.1, a delay-based technique has a unique benefit because it does not need to activate the Trojan either partially or fully. Moreover, each path delay is relatively independent, so it is less affected by other paths of the chip, and a Trojan can potentially contribute more to a path delay change than total circuit power. Existing delay-based Trojan detection methods face the following challenges: (1) To ensure a maximum detection coverage of Trojans that can potentially be placed and distributed on various paths besides critical paths. (2) The measurement of paths delay at a low cost. Since there is a huge number of paths in a design, any additional hardware for path delay measurement will increase the area and silicon cost significantly. Taking into account these issues, in Chapter 2, we will use a clock sweeping technique to obtain path delay information without any additional hardware. Once the data has been collected by clock sweeping, we generate a series of delay fingerprints for ICs, and then apply statistical methods to analyze whether the chips under test contain Trojans. Since transitions are easier generated on short paths as demonstrated in [50], the Trojans on the short paths can be detected more efficiently by power-based Trojan detection techniques [39] [40]. By combining this technique with existing power based Trojan detection methods, a very high detection coverage can be achieved. Simulation results using 90nm technology and implementation results using Xilinx Spartan-3E FPGAs demonstrate the effectiveness of our method under process variations, even for Trojans as small as a few gates.

### Recycled IC Detection

Since the performance of recycled ICs must have been degraded by aging effects, compared to fresh ICs, a path-delay based fingerprinting technique is proposed to sensitize the aging effects and identify recycled ICs. For fresh ICs, the delay distribution of paths will be within a certain range. The fingerprints of the fresh ICs can be generated during manufacturing test of these ICs and stored in a secure memory for future use when identifying recovered ICs. Due to aging effects, such as negative/positive bias temperature instability (NBTI/PBTI) and hot carrier injection (HCI), the path delays in recycled ICs will be larger than those in fresh ICs. For a chip under authentication (CUA), the larger the path delays are, the higher the probability there is that the CUA has been used and is a recovered IC. In Chapter 3, we propose a fingerprinting and authentication flow for accurately identifying recycled ICs. Statistical data analysis is used to distinguish the path delay changes caused by process and temperature variations from those caused by aging. Since the path delay information is measured during the manufacturing test process, no extra hardware circuitry is required for this technique. In addition, there is no change required in current industrial design and test flows. Finally, this technique presents no area overhead, no power consumption, and is resilient to attacks.

#### On-Chip ID Generation

We will use the proposed clock sweeping technique to uniquely identify ICs. In Chapter 4, we develop a flow to create a unique binary identifier for each IC by using the path delay information. In addition, we will analyze our ability to accurately identify ICs under measurement and environmental noise. Simulation results from 90nm technology and experimental results from 90nm FPGAs demonstrate the effectiveness of our technique.

### 1.3.2 Design-for-Security Methodologies

Although post-silicon test methodologies can address or mitigate some security problems to some degree, limitations caused by existing circuit structure are also very apparent.

In order to handle security threats better, hardware security issues are considered earlier in the design process and thus some design-for-security methodologies have been developed. We develop a low-overhead built-in self-authentication (BISA) technique that can effectively prevent hardware Trojan insertion and reverse engineering by untrusted foundries.

### Hardware Trojan Prevetion

Detecting hardware Trojans is very challenging because of the diversity of Trojans and unpredictable process variations during fabrication. In Chapter 5, we will discuss how to use BISA to make hardware Trojan insertion by untrusted GDSII developer and untrusted foundry considerably more difficult and easier to detect. The unused spaces in the circuit layout represent the best opportunity to insert Trojans by these entities. BISA works by eliminating this spare space and filling it with functional filler cells, instead of non-functional filler cells. A self-testing procedure generates a digital signature that will be different if any BISA cells are changed because of hardware Trojan insertion. We demonstrate that BISA can be applied to any flat or bottom-up hierarchical design with negligible overhead in terms of area, power, and timing.

### Design Obfuscation for Anti-reverse engineering

Split manufacturing has emerged as a viable approach to protect integrated circuits (ICs) fabricated in untrusted foundries, but has high cost and/or high performance overhead. Furthermore, split manufacturing cannot fully prevent untargeted hardware Trojan insertions. In Chapter 6, we propose to insert additional functional circuitry called obfuscated built-in self-authentication (OBISA) in the chip layout with split manufacturing process, in order to prevent reverse-engineering and further prevent hardware Trojan insertion. Self-tests are performed to authenticate the trustworthiness of the OBISA circuitry. The OBISA circuit is connected to original design in order to increase the strength of obfuscation, thereby allowing a higher layer split and lower overall cost. Additional fan-outs are created in OBISA circuitry to improve obfuscation without losing testability. Our proposed gating mechanism and net selection method can ensure negligible overhead in terms of area, timing, and dynamic power. Experimental results demonstrate the effectiveness of the proposed technique in several benchmark circuits.

### 1.3.3 Security Primitive Enhancement Methodologies

A Physically Unclonable Function (PUF) is a structure that when issued a challenge, it produces a unique and reliable response which can be used as an identifier or a cryptographic key. SRAM PUFs create unique responses upon power up as certain SRAM cells output a ‘1’ or ‘0’ with high probability due to uncontrollable process variations. A current challenge in SRAM PUFs is their sensitivity to temperature and voltage variations as well as aging. By creating algorithms that isolate stable bits quickly and with minimal testing, the use of SRAM PUF should become more practical. In Chapter 7, we explore the selection of stable bits through enrollment under different conditions (temperature, voltage, and aging). The basis of this algorithm is to identify the most stable bits based upon the performance of their neighboring cells in a series of enrollment tests (testing under some extreme corner conditions). The algorithm finds the cells which consistently only output a value of ’0’ or ’1’ and then ranks their overall stability using a weighting algorithm. The weighting algorithm takes into account the number of stable and unstable neighbors a cell has. We have analyzed data from SRAM chips and demonstrated the proposed technique is able to improve the reliability of SRAM PUF significantly.

## 1.4 Thesis Outline

- *Chapter 1:* Introduction
- *Chapter 2:* Post-Silicon Test Methodology: Delay-based IC Fingerprinting for Hardware Trojan Detection
- *Chapter 3:* Post-Silicon Test Methodology: Delay-based IC Fingerprinting for Recycled IC Detection
- *Chapter 4:* Post-Silicon Test Methodology: Delay-based IC Fingerprinting for On-chip ID Generation
- *Chapter 5:* Design-for-Security Methodology: Built-In Self-Authentication for Hardware Trojan Prevention
- *Chapter 6:* Design-for-Security Methodology: Obfuscated Built-In Self-Authentication for Anti-reverse Engineering
- *Chapter 7:* Security Primitive Enhancement: Bit Selection Algorithm Suitable for High-Volume Production of SRAM-PUF
- *Chapter 9:* Conclusions

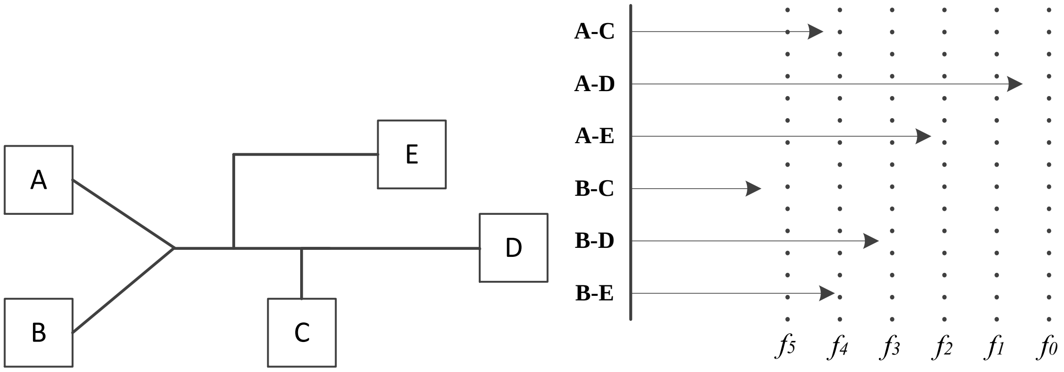
# Chapter 2

# Post-Silicon Test Methodology: Delay-based IC Fingerprinting for Hardware Trojan Detection

The threats and challenges introduced by hardware Trojan have been described in Chapter 1.1.1. Conventional delay tests have limitations to detect Trojans, which can only identify Trojans that increase a path's delay by more than its available slack. This is unlikely to happen since adversary will design Trojans hard to be detected by these patterns by avoiding critical paths and ensuring that the delay induced by Trojan is smaller than slack. In this chapter, a clock sweeping technique is developed, which is able to target Trojans on shorter paths. Then, the effectiveness of the proposed methodology will be evaluated in simulation and FPGA implementation.

## 2.1 Clock Sweeping

*Clock sweeping* is the process of applying patterns to a path multiple times with different frequencies to find a frequency at which the path cannot propagate its signal, often for purposes of speed binning. By observing the frequencies at which the path can and cannot propagate its signal, the delay of the path can be measured with some degree of precision. Our ability to perform clock sweeping on a path is limited by the degree of control we have on the clock that controls the capturing memory elements (i.e., the flip-flops), the degree to which we can excite paths in the circuit, and the lengths of the paths in the IC. For the clock sweeping technique, the conventional transition delay fault (TDF) or path delay fault (PDF) test patterns can be used. These fault models are common in automatic test pattern generation (ATPG), so no extra considerations for test pattern generation and test coverage are needed.



(a) (b)

**Fig. 2.1:** (a) An example circuit and (b) Clock sweeping on paths in the circuit.

Figure 2.1(a) shows a visual example of clock sweeping being performed on several paths that can be sensitized by test patterns. The clock period is swept from $f_0$ to $f_5$ and the sweep step size is a fixed $\Delta$t, as shown in Figure 2.1(b). As an example, path B-D is able to propagate correct values at frequency $f_0$ to $f_3$ (pass), and will produce wrong logic values at frequency $f_4$ (fail). Thus, its start-to-fail clock frequency is $f_4$, which denotes the length of path B-D is between the frequency $f_3$ and $f_4$. When the clock is swept from low frequencies to high frequencies, paths will fail sequentially, with longer paths failing before shorter paths. Path B-C will succeed in propagating its signal at all six clock frequencies in this example, because it is too short to test. All of the paths have some number of frequencies they will pass at, some they may fail at, and some they are guaranteed to fail at. When the clock is swept from low frequencies to high frequencies, paths will fail sequentially, with longer paths failing before shorter paths. If the sweep range is large enough, many paths will fail during clock sweeping, allowing their delay information to be obtained, even for hazardous patterns if the size of the hazard is larger than the sweep step size.

The ability to perform clock sweeping on a path is limited by the availability of sweeping clock frequencies, the degree to which we can excite paths in the circuit, and the lengths of the paths in the IC. The range and step size of sweeping frequencies are two critical parameters involved in the clock sweeping technique. A smaller step size could be more sensitive to small delay changes caused by the wearout mechanism and process variation. The range determines the range of testable paths (long paths). Higher maximum frequency could fail more long paths during clock sweeping. However, the maximum frequency applied to the circuit cannot go very high, because it is also restricted by the design characteristics, path-delay distribution in the design, the maximum power consumption, and the on-chip or off-chip frequency generator's limit. For example, with the Ocelot ZFP tester, the main frequency is 400MHz and the frequency step size is 1MHz.

Figure 2.2 shows the flow of the clock sweeping technique. The delay test patterns are applied to ICs at different clock frequencies. By analyzing the pass and fail values with clock sweeping, the start-to-fail frequency at each flip-flop for each pattern can

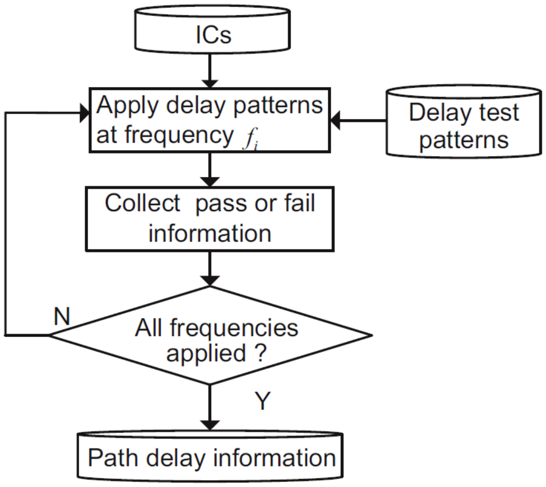

ICs
Apply delay patterns at frequency $f_i$
Delay test patterns
Collect pass or fail information
All frequencies applied ?
N
Y
Path delay information

**Fig. 2.2:** Clock sweeping flow.

be obtained. As an example, Table 2.1 presents the start-to-fail frequency for each pattern/flip-flop combination in ICs. Patterns are able to sensitize different kinds of paths at different flip-flops: long, short or no path at all. For these flip-flops at which short or no paths are sensitized (like the path B-C in Figure fig:csweeping(b)), they always capture "passing" values during the clock sweeping. These invalid pattern/flip-flop combinations need to be discarded. For example, assuming the pattern 2/flip-flop 2 (P2/FF2) is an invalid combination, the column P2/FF2 has been removed from Table 2.1. The remaining valid elements form *path delay fingerprint*, as is shown in Table 2.1. Each row of the table is a path delay fingerprint for an IC and each column is an element of the delay fingerprint.

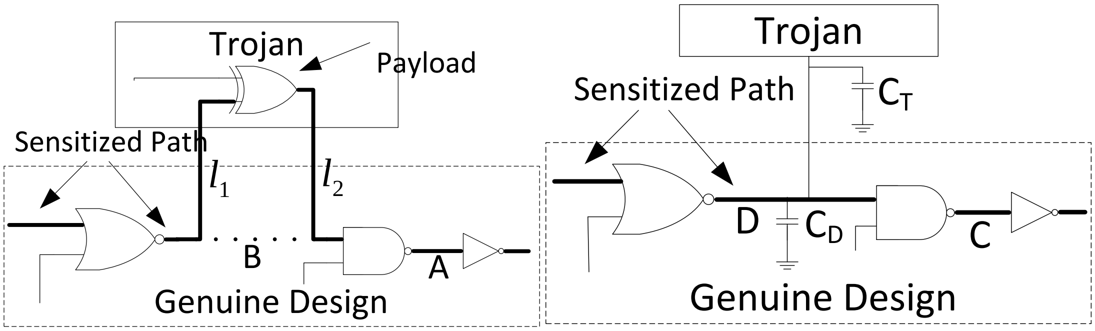
**Table 2.1:** Pattern/flip-flop (P$i$/FF$j$) combinations with start-to-fail frequencies.

| | P1/FF1 | P1/FF2 | ... | P2/FF1 | P2/FF3 | ... | P$P_1$/FF$m$ |
|---|---|---|---|---|---|---|---|
| IC 1 | $f_6$ | $f_{14}$ | ... | $f_3$ | $f_{20}$ | ... | $f_{10}$ |
| IC 2 | $f_7$ | $f_{12}$ | ... | $f_4$ | $f_{21}$ | ... | $f_9$ |
| IC 3 | $f_5$ | $f_{11}$ | ... | $f_3$ | $f_{19}$ | ... | $f_{10}$ |
| ... | ... | ... | ... | ... | ... | ... | ... |
| IC $n$ | $f_8$ | $f_{14}$ | ... | $f_4$ | $f_{23}$ | ... | $f_{12}$ |

## 2.2 Trojan's Impact on Path Delay

We can expect that intelligent adversaries will try to maintain the original design layout as much as possible and insert Trojans into unused spaces of the layout to keep the Trojan hidden. In order to simplify this problem, we consider the nodes (outputs of gates) in the genuine IC instead of the paths for the analysis in this chapter. These nodes might be affected by either a trigger or a payload from a Trojan [6]. Thus, we consider three types of Trojans depending on how they are activated and their action to the functional circuit: Trojans with only payloads (TP), Trojans with only triggers (TT), and Trojans with triggers and payloads (TTP).

For any Trojan trying to change the function of design, a payload gate has to be inserted at a node. An example of a TP is shown in Figure 2.3(a). The sensitized path in the genuine design (the bold line) passes through node $B$. This node, originally only containing a wire (the dashed line), is replaced by a logic gate with input and output interconnections. The additional delay consists of the propagation delay of the payload



(a) (b)

**Fig. 2.3:** (a) An example of TP, (b) an example of TT.

and the delay from the two wires' capacitances ($l_1$ and $l_2$). For any internally activated Trojan, the triggered parts will introduce additional interconnections which will cause unavoidable increased capacitance on the node. In Figure 2.3(b), after TT insertion, the capacitance of node $D$ is increased from $C_D$ to $C_D + C_T$, thereby increasing the propagation delay of the NOR gate and increasing the delay of all sensitized paths passing through node $D$. Besides TP and TT, TTP would have the cumulative effect of TP and TT.

**Table 2.2:** A path delay without and with TP with short and long $l_1$ and $l_2$.

| | Short $l_1$ and $l_2$ | Long $l_1$ and $l_2$ |
|---|---|---|
| W/O Trojan | 848ps | 1079ps |
| W/T Trojan | 887ps | 1277ps |
| Increased Delay | 39ps | 198ps |

In order to show the effect of TP and TT on paths delay, we performed simulations in 90nm technology. We inserted two payload gates (minimum-sized NAND) at two positions. One is physically very close to the node (short $l_1$ and $l_2$ as in Figure 2.3(a)) and the other is remote from the node (longer $l_1$ and $l_2$). In Table 2.2, delay of path going through node $B$ is measured, and the results show that a TP has increased the path delay significantly, more so for Trojans with long interconnections. That's because longer $l_1$ and $l_2$ introduce larger wire capacitance.

Next, we place a Trojan gate (minimum-sized NAND) at four different locations, with one input connecting to the node $D$ on the sensitized path. The first location is very close to node $D$ ($l_3$ is short as in Figure 2.3(b)), with locations 2, 3, and 4 being successively further away from node $D$. The delay of sensitized path is measured with and without Trojans for the different locations. The extra delay caused by TT is shown in the third row of Table 2.3. Although the increased delay is still relatively small at the location 1 (shown as Loc. 1 in Table 2.3), the TT effects at locations 2 through 4 are comparable to the effects of the payload shown in Table 2.2. Thus, as long as the standard cells in standard design style ASICs are well planned and tightly packed, the TT effect could be very obvious.

**Table 2.3:** A path delay without and with TT at four different locations.

| | W/O Trojan | Loc. 1 | Loc. 2 | Loc. 3 | Loc. 4 |
|---|---|---|---|---|---|
| Path Delay | 764.5ps | 794.5ps | 837.7ps | 890.0ps | 953.8ps |
| Increased Delay | 0ps | 30ps | 73.2ps | 125.5ps | 189.3ps |

The changes of path delay introduced by hardware Trojans could be captured by the clock sweeping technique. Suppose that a Trojan load is added to a path as shown in Figure 2.4(a); the additional capacitive load will result in a small, extra delay on paths A-E and B-E, which may push the arrow to the right and even fail path A-E at f2 and path B-E at f3. In this case, the change of start-to-fail frequency could be detected by clock sweeping. If a Trojan is activated and its payload becomes nontransparent, the required transition cannot be generated at payload gate. This faulty function indicates Trojan's existence and makes it easier to detect. Trojans without triggers and payloads would not be detected using this technique since they most likely use an antenna to receive triggers or leak information. They can be more effectively detected by power-based Trojan detection approaches [39] because they tend to use more gates and consume more power than TP, TT or TTP.

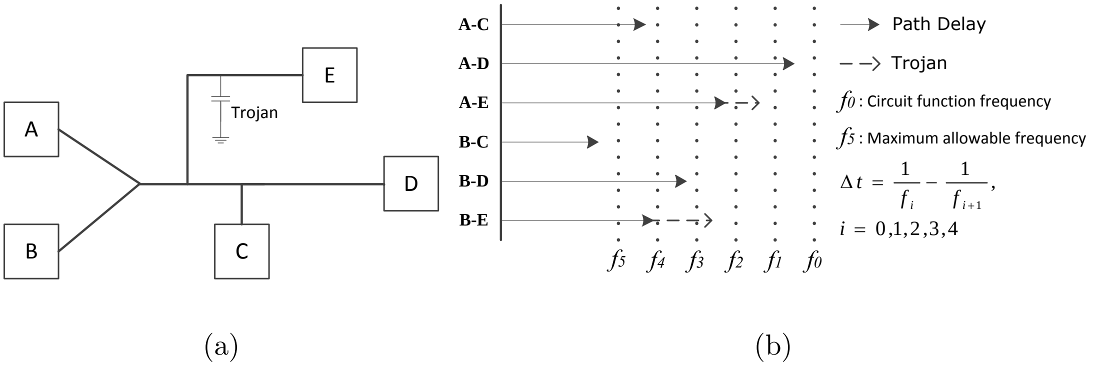

A
B
C
D
E
Trojan
(a)
A-C
A-D
A-E
B-C
B-D
B-E
$f_5$ $f_4$ $f_3$ $f_2$ $f_1$ $f_0$
Path Delay
Trojan
$f_0$ : Circuit function frequency
$f_5$ : Maximum allowable frequency
$\Delta t = \frac{1}{f_i} - \frac{1}{f_{i+1}}$,
$i = 0,1,2,3,4$
(b)

**Fig. 2.4:** (a) An example circuit with a Trojan and (b) clock sweeping on paths in the circuit with a Trojan.

## 2.3 Trojan Detection Methodology

### 2.3.1 Signature Generation Procedure

The number of paths in modern circuits is exponential relative to the number of nodes, thus, it is impractical to measure all paths to generate a signature. Additionally, measuring the delay of short paths is extremely difficult. Both the nodes on short and long paths have been taken into considerations in our proposed procedure which is shown in Figure 6.5. The following steps are performed to generate signatures for all ICs .

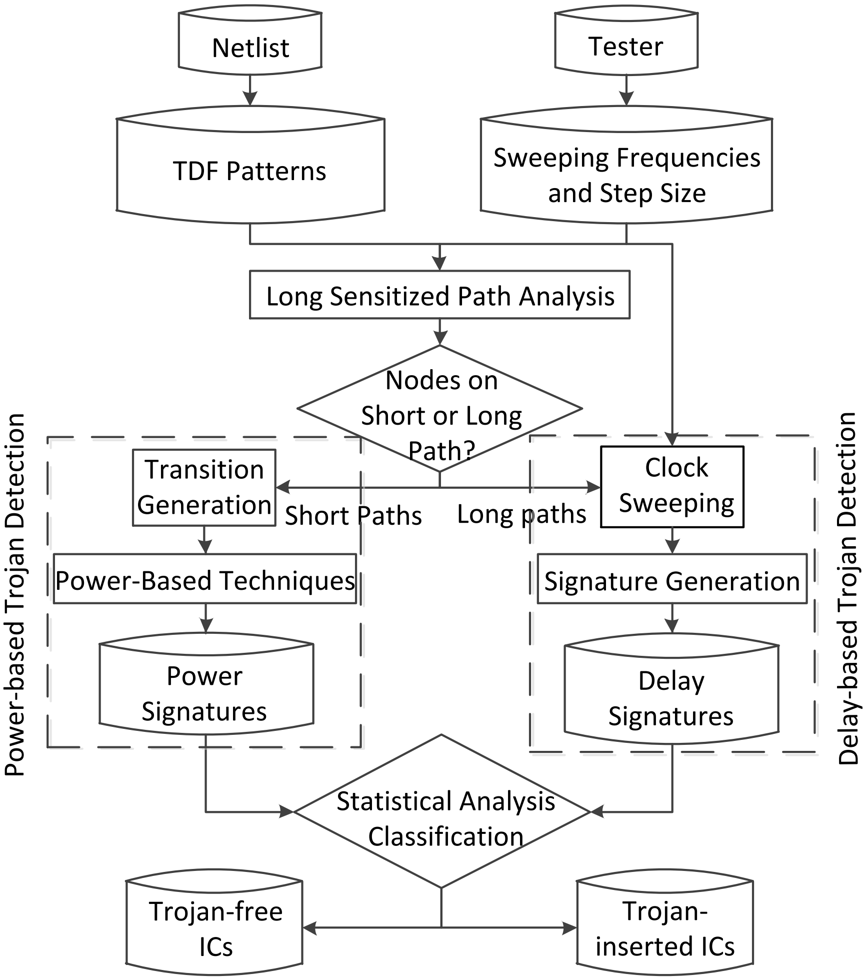


**Fig. 2.5:** The proposed signature generation procedure using clock sweeping.

**Test Pattern Generation:** Similar to the TDF model, here we assume that TP or TT affects a node in the circuit, making the corresponding gate slow to fall or slow to rise. Thus, the TDF patterns can be used in our procedure. One advantage of the TDF model is that the number of faults is linear relative to the number of nodes;

thus, we can use a portion of paths to cover all nodes in the circuit, and the cost of measuring paths delay could be decreased dramatically. Another advantage of using the TDF pattern set is that it is widely used in industry; the ATPG processes are mature and many proposed techniques can be used to increase its fault coverage [85].

**Sweeping Frequencies and Step Size Selection:** The range and step size of sweeping frequencies are two critical parameters involved in clock sweeping, because they determine the effectiveness of our Trojan detection technique. A smaller step size could be more sensitive to the small extra delay induced by Trojans. The range determines the range of long paths. Higher maximum frequency could fail more long paths during clock sweeping so that higher Trojan node coverage for delay-based Trojan detection can be achieved. The test time and data volume overhead will increase as a larger range and smaller step are selected. Additionally, the clock sweeping range and step size are both dependent on the testing equipment.

**Nodes Selection:** Once the clock sweeping frequencies are determined, the sensitized path delays larger than the maximum frequency are considered as “long paths”. Otherwise, they are called “short paths”. For example, in Figure 2.1(b), B-C is a short path and others are long paths; $f_5$ is the maximum application frequency. The delay of all the long paths can be obtained from their start-to-fail frequencies. Since the nodes on long paths can be authenticated in clock sweeping, all the patterns sensitizing these long paths will be kept. Patterns only sensitizing short paths are still useful in generating transitions for power-based Trojan detection [39] [40]. This step divides the nodes in a circuit into two groups- *nodes are authenticated either by the clock sweeping or by* *power-based detection techniques.* Moreover, removing patterns that sensitize only short paths could save a significant amount of test time during clock sweeping. In our work, we focus on the right branch of the procedure as shown in Figure 6.5.

**Clock Sweeping:** This process is similar to the conventional TDF test. The only difference is that we need to apply each pattern that sensitizes at least one long path at different clock frequencies. The logic values captured by the scan flip-flops under the different clock frequencies are shifted out.

**Signature Generation:** By analyzing the pass and fail values, we find the start-to-fail frequency at each flip-flop for each pattern. As an example, Figure 2.6(a) presents the start-to-fail frequency for each pattern/flip-flop combination in ICs. Long-path patterns are able to sensitize different kinds of paths at different flip-flops: long, short or no path at all. For these flip-flops at which short or no path are sensitized, they always capture "passing" value during clock sweeping. These invalid pattern/flip-flop combinations need to be discarded from our signature. For example, assuming the pattern 2/flip-flop 2 (P2/FF2) is an invalid combination, the column P2/FF2 has been removed from Figure 2.6(a). Thus, the final signature length will be shorter, as is shown in Figure 2.6(a). Each row of the table is a signature for an IC and each column is an element of the signature. The multidimensional signature will be processed by multidimensional scaling described in Section III.D.

### 2.3.2 Node Coverage Analysis

The objective of our technique is to recognize load capacitance induced by Trojans and capture its impact on a path. The coverage analysis in our technique can be divided

| | P1/FF1 | P1/FF2 | ... | P2/FF1 | P2/FF3 | ... | P$P_1$/FF$m$ |
|---|---|---|---|---|---|---|---|
| IC 1 | $f_6$ | $f_{14}$ | ... | $f_3$ | $f_{20}$ | ... | $f_{10}$ |
| IC 2 | $f_7$ | $f_{12}$ | ... | $f_4$ | $f_{21}$ | ... | $f_9$ |
| IC 3 | $f_5$ | $f_{11}$ | ... | $f_3$ | $f_{19}$ | ... | $f_{10}$ |
| ... | ... | ... | ... | ... | ... | ... | ... |
| IC $n$ | $f_8$ | $f_{14}$ | ... | $f_4$ | $f_{23}$ | ... | $f_{12}$ |

(a)

| Clock Period | >1CP | 0.9CP | 0.7CP | 0.5CP | 0.2CP | 0CP |
|---|---|---|---|---|---|---|
| Node Coverage | 0% | 48.59% | 61.01% | 78.87% | 95.18% | 100% |

(b)

| Stage | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Path 1 | 0.5 | 0.5 | 0.13 | 0.00059 | 0.00040 | 0.00023 | |
| Path 2 | 0.5 | 0.29 | 0.28 | 0.063 | 0.012 | 0.0024 | 0.00046 |
| Path 3 | 0.5 | 0.25 | 0.11 | 0.015 | 0.000296 | 0.000199 | 0.000115 |

(c)

**Fig. 2.6:** (a) Pattern/flip-flop (P$i$/FF$j$) combinations with start-to-fail frequencies, (b) node coverage on different clock frequencies, (c) transition probabilities on three quiet paths.

into two parts: nodes on long paths and nodes on short paths.

Clock sweeping can guarantee that all sensitized long paths will fail at a particular clock frequency. Hence, the node coverage on long paths is dependent on the TDF coverage. The TDF is widely used in VLSI testing, so there have been many approaches proposed to improve the coverage of the TDF in recent years [85]. All these techniques can be applied in our procedure to achieve maximum coverage. In the meantime, a Trojan's load capacitance can slow down both rising and falling transitions. For Trojan detection, one fault, slow-to-rise or slow-to-fall, is sufficient to indicate the existence of Trojans on that node instead of two faults as in the TDF model. Therefore, node coverage will be the ratio of all detected nodes by either slow-to-rise or slow-to-fall using TDF long-path patterns to the total number of nodes in the circuits.

Figure 2.6(b) shows the estimated node coverages at different clock frequencies in benchmark s38417 for reference. ATPG patterns which are able to reach 99.4% TDF coverage are used to sensitize all testable paths. The clock period in Figure 2.6(b) is given in the form of the percentage of the longest critical path (CP) in the circuit.

Short paths, whose delay is smaller than the maximum frequency we can apply, will never fail during clock sweeping. It is very difficult to measure short paths due to the maximum frequency limitation of the tester and the power limit of the IC.

In general, the quietest nodes which have very few transitions are usually on the long paths in a circuit [50]. The low transition probability of a gate is resulted from one or more inputs with low transition probabilities. After propagating signal through one more gate on a path, the transition probability may further decrease. Thus, a very quiet gate is usually located on a long path since its low transition probability is caused by previous multiple stages of gates. In order to verify this inference, we apply random patterns to primary inputs and scan chain, and then calculate transition probability for each gate. The gates with low transition probabilities are selected to be analyzed. We choose three quiet paths and their transition probabilities at different stages are shown in the Figure 2.6(c). As the number of stages of the path increases (1 through 7), the transition probability goes down.

Trojan gates have a higher probability to switch when they connect to nodes on short paths, which means they tend to consume more power [50]. Power-based Trojan detection [39] can effectively detect Trojans on short paths, so adversaries could put Trojans on long paths to hide them. Our clock sweeping technique is able to make up for the deficiency involved with power-based Trojan detection. Increasing the test coverage in both power-based and delay-based approaches can improve the possibility to identify infected chips.

### 2.3.3 Statistical Analysis Method

Trojan detection is extremely difficult due to process and environmental variations, especially when the Trojan is small and has very short wire connections. In order to detect Trojans, a statistical analysis method is used to separate Trojan-free ICs and Trojan-inserted ICs.

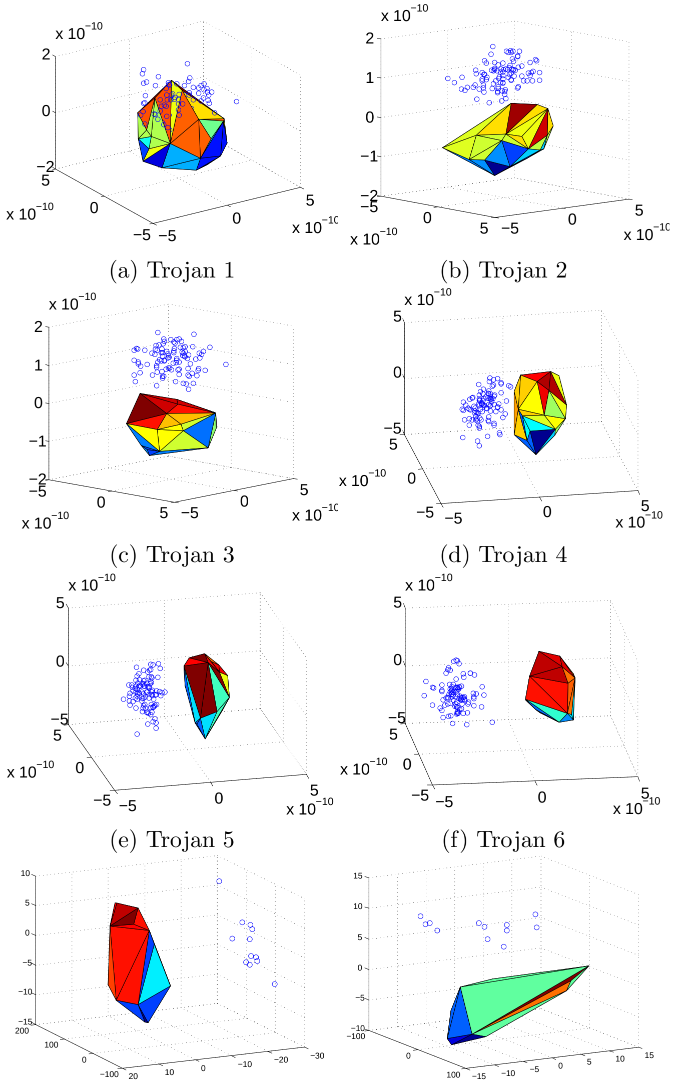
Multidimensional Scaling (MDS) is a method for visualizing dissimilarity in data. The typical goal of MDS is to create a configuration of points in one, two, or three dimensions, whose interpoint distances represent the original dissimilarities. In a similar manner, [6] was first to use PCA/Convex for hardware Trojan detection. The different forms of MDS use different criteria to quantify dissimilarity, such as metric and nonmetric multidimensional scaling [86]. The MDS we used in our method maps the original high dimensional space to a lower dimensional space, at the same time attempts to preserve the pairwise distance and finally isolate the dissimilar chips which may carry Trojans. As described in Section III.A, the signature of a chip is composed of a set of path delays. One element in a signature is considered as a dimension. so that each IC can be represented as an $x$-dimensional point ($x$ being the total number of valid pattern/flip-flop combinations). Their Euclidean distances in high-dimensional space depend on the effects of both process variations and Trojans. Since outlying points in high dimensions contain Trojans and their corresponding points in low dimension are still outliers, we can use outlier points in low dimensions to predict statistical likelihood of Trojan presence. For our technique, the $x$-dimensional signatures for Trojan-free ICs will be mapped to a 3-dimensional space by MDS, and then a convex hull will be constructed. If the signature of an IC under authentication is located outside of the convex hull, this IC is considered suspicious and may contain Trojans.

## 2.4 Results and Analysis

### 2.4.1 Simulation Results

In order to demonstrate the effectiveness of our proposed technique, the simulations were performed on an implementation of the ISCAS'89 benchmark s38417 using a 90 nm technology library. After synthesis, the s38417 benchmark circuit has 1564 flip-flops and 4046 logic gates. The layout was completed with the Synopsys physical design tool IC Compiler. The Trojan gates were inserted and routed in unused spaces in the layout by using IC Compiler. The impact of process variations on threshold voltage ($V_{th}$), oxide thickness ($T_{ox}$) and channel length ($L$) have been taken into considerations as well. Both the inter-die and intra-die process variations for each of the three previously mentioned parameters are 5% in our simulations. 300 Monte Carlo Simulations, including 200 simulations for Trojan-free chips and 100 for Trojan-inserted chips, were performed at a temperature of 25°C for each Trojan using HSPICE.

For our delay-based Trojan detection technique, we do not concern ourselves



(a) Trojan 1

(b) Trojan 2

(c) Trojan 3

(d) Trojan 4

(e) Trojan 5

(f) Trojan 6

(g) Trojan 7

(h) Trojan 8

| Step Size | 10ps | 20ps | 40ps | 60ps | 80ps | 100ps |
|---|---|---|---|---|---|---|
| Trojan 2 | 100% | 65% | 55% | 38% | 14% | 9% |
| Trojan 3 | 100% | 100% | 100% | 100% | 42% | 11% |
| Trojan 4 | 100% | 100% | 100% | 100% | 100% | 100% |
| Trojan 6 | 100% | 100% | 100% | 100% | 100% | 100% |

(i) Detection rate at different step sizes.

**Fig. 2.7:** Outlier analysis using MDS for Trojans 1-6 using simulation ((a)-(f)) and Trojans 7-8 on FPGAs ((g)-(h)), and step sizes analysis (i).

with Trojan's type or how many gates the Trojan has. Instead, we focus on how many triggers and payloads the Trojan will bring to the original design. We have inserted a large number of Trojans in this design among which we selected six Trojans to present results in details. Every Trojan is composed of a few minimum-sized gates, and these gates are placed in the nearest available unused space to keep the wire connections as short as possible. This will make the Trojan harder to detect. These six Trojans were constructed as follows: Trojan 1 has one payload and one trigger (TTP type). Trojan 2 has the same structure as Trojan 1, but is inserted at a different node. Trojan 3 has two payloads with very short connections (TP type). Trojan 4 has four triggers and no payload (TT type). Trojan 5 has three triggers and two payloads (TTP type). Trojan 6 has six triggers and four payloads (TTP type).

The maximum frequency, functional frequency and step size in our simulations are 1.5GHz, 700MHz and 10ps, respectively. By using the methodology described in Figure 6.5, 300 signatures for 200 Trojan-free ICs and 100 Trojan-inserted ICs are generated. After MDS processing described in Section III.D, the outlying results for Trojans 1 through 6 are shown in Figures 2.7(a)-(f). The convex hull is formed using signatures of 200 Trojan-free ICs. These points are placed much closer to each other than the rest of hollow dots obtained from Trojan-inserted ICs. According to the distance between outliers and convex hull, two classes of ICs are separated. While Trojan 1's detection rate is 64% (64 out of 100 Trojan-free ICs are detected), from Trojan 2 to Trojan 6, their detection rates are all 100%. The Trojan 1 is the worst case for Trojan with payload and trigger because it has one minimum sized NAND gate which is used as a payload and only one sensitized path passes through this payload. More triggers, payloads and sensitized paths passing through Trojan nodes make detection from Trojan 2 to 6 easier. There might be some limitations associated with the MDS algorithm since it treats the signatures from the ICs as linear. As a part of our future work, we might apply different statistical classification algorithms (such as Diffusion Map and Support Vector Machine) with the aim of further improving the detection rate. However, in our analysis (particularly for Trojans 2 to 6), MDS seems to easily classify all the chips shown in Figures 2.7(b)-(f). Note that we have randomly inserted Trojans on a very large number of locations in the circuit and was able to observe similar results.

### 2.4.2 Trojan Size and Location Analysis

Generally, larger Trojans might have more triggers and payloads, so they bring a larger impact to the original circuit. From Trojan 1 to Trojan 6, as the size of Trojans increases, these triggers and payloads affect larger number of nodes in the circuit. In other words, the larger Trojan size means that more paths, sensitized by the TDF patterns, will be impacted by the Trojan. Although Trojans' impacts may be masked by process variations, a larger number of sensitized paths always lead to higher detection possibility. Additionally, larger size Trojan most likely results in longer interconnection for triggers and payloads due to the limited unused space nearby for Trojan insertion. The extra delay induced by the larger Trojan will then increase. This can be seen in Figure 2.7; as we move from (a) to (f), the points also move away from the convex hull, i.e., the distance between Trojan-inserted ICs and Trojan-free ICs becomes larger. Thus, as the size of the Trojan increases, it becomes easier to separate Trojan-inserted and Trojan-free ICs by using the proposed technique.

In addition to Trojan size, the Trojan's location also has a significant influence on the results. Scan flip-flops can be considered pseudo-primary inputs and outputs. Sensitized paths spread out like a cone from scan flip-flops' outputs and converge at scan flip-flop's inputs. The nodes closer to a scan flip-flop will have more of a chance to influence these sensitized paths. Thus, these Trojans are easier to be detected. In our simulations, although Trojans 1 and 2 have same size, Trojan 2 is closer to scan flip-flops and Trojan 1 is farther away. In Figure 2.7, hollow dots in (b) are farther from convex hull than that in (a), which means Trojan 2 is easier to be separated than Trojan 1. Thus, we have obtained 100% detection rate for Trojan 2, while Trojan 1 only has a 64% detection rate.

### 2.4.3 Clock-Sweeping Step Size Analysis

For clock sweeping, path delay gained from simulation needs to be translated to their nearest achievable clock frequency according to the clock step size. The range and step size selection are described in Section III.A. From the results shown in Figure 2.7(i), we can clearly see that the detection rate reduces as the step size increases. The impact of step size becomes less for larger Trojans, because it has more triggers or payloads on sensitized paths and introduces larger extra delay.

### 2.4.4 FPGA Implementation

The benchmark s9234 was implemented on 90nm Xilinx Spartan-3E FPGAs with 145 scan flip-flops and 571 TDF patterns. Considering the limitations of the DCM, 23 different clock frequencies, sweeping from 5ns (maximum clock frequency) to 9.4ns (functional clock frequency) with a step size ($\Delta t$) of 200ps, are generated by a Digital Clock Management (DCM) in the FPGA. To reduce measurement noise, each frequency was measured three times. Since the step size, limited by accuracy of the clock crystal and DCM, is larger than the predicted impact of any trigger, we only focused on TPs in the FPGA implementation.

In this experiment, two types of Trojan with payload are inserted separately in the layout by using Xilinx FPGA Editor. The payload gate of the first Trojan (Trojan 7) was inserted close to a scan flip-flop, while the second Trojan's payload (Trojan 8) was inserted farther away from any scan flip-flops. 44 separate FPGA boards were used, with 32 being Trojan-free and 12 being Trojan-inserted. We randomly chose 80 patterns from the 571 TDF patterns for analysis to reduce test and measurement time, so the total number of pattern/flip-flop combinations is $80 \times 145 = 11,600$. After removing the invalid pattern/flip-flop combinations which cannot fail any path in the clock sweeping range as described in Section III.A, the remaining number of pattern/flip-flop combinations is 786. These 786 pieces of delay information are used as the signature for each chip. The entire measurement, pattern application, and clock sweeping was automated and the data from scan chains were sent to a computer for further analysis. Figure 2.7(g)-(h) shows the scaled signature by using MDS for Trojan detection. The convex hull is drawn according to the signatures of the 32 Trojan-free FPGAs, and the 12 hollow dots represent the signatures from FPGAs with Trojans. While all hollow dots are totally separated from the convex hulls, and the detection rate is 100% for both

Trojans, we note that the first Trojan is easier to separate as the distance between the hollow dots and convex hull is larger. The reason for this is that the payload gate closer to the scan flip-flops influences 79 sensitized paths, while the other Trojan only affects 9 sensitized paths.

## 2.5 Conclusion

In this chapter, a clock sweeping is proposed to obtain the critical and non-critical path delay and then generate fingerprints for ICs for the purpose of detecting hardware Trojans. Post data processing method has proved to be effective at identifying Trojan-inserted ICs in the presence of process variations, as demonstrated by both our simulation and FPGA implementation results.

# Chapter 3

# Post-Silicon Test Methodology: Delay-based IC Fingerprinting for Recycled IC Detection

The counterfeiting of integrated circuits (ICs) has become a major issue for the electronics industry. Recycled ICs that find their way into the supply chains of critical applications can have a major impact on the security and reliability of those systems. This chapter will present a post-silicon test methodology for detection of recycled ICs. An IC fingerprinting procedure using the clock sweeping technique is introduced. Statistical data analysis will be used to distinguish the delay changes caused by process variations from those changes caused by silicon aging [87].

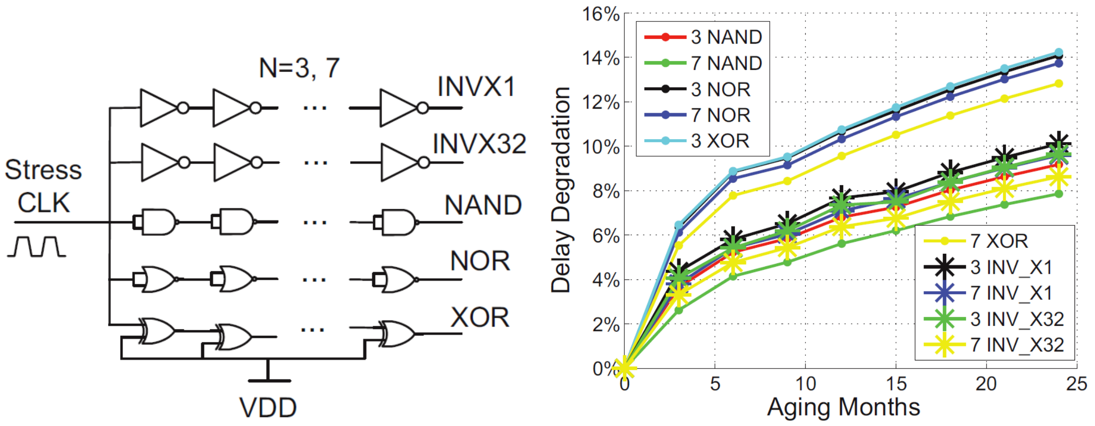
## 3.1 Aging Degradation on Path Delay

When a chip is used in the field, aging effects cause some of its parameters to shift over time. This section will briefly discuss the mechanics of the two most common wear-out mechanisms for a CMOS device: Negative Bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI). Both mechanisms deteriorate transistor performance.

- NBTI occurs when traps are accumulated on the boundary of the Si-SiO2 interface, which usually happens when PMOS is reversely biased. NBTI increases the absolute value of the PMOS threshold voltage and results in decreasing drain current, decreasing transconductance of a MOSFET and consequently increasing gate propagation delay [89] [90].

- HCI is another major wear-out mechanism. Drain-source electrical field accelerates the charge carriers to reach a high kinetic energy, which enables the avalanche of secondary carriers through impact ionization. The process consequently creates traps at the gate dielectric/silicon substrate interface, as well as dielectric bulk traps, and therefore degrades device characteristics including voltage threshold [91] [92]. While the HCI was considered less dominant than the NBTI in the past, continued semiconductor scaling has increased its impact significantly.

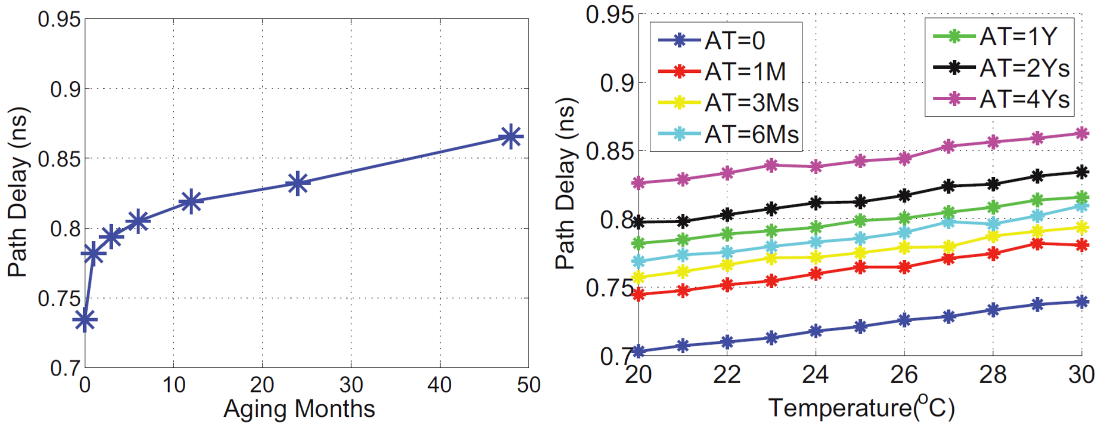
Since recycled ICs have experienced the above silicon aging in the field, the path delay of recycled ICs will be different from those of fresh ICs. In order to demonstrate the impact of aging on path delay in ICs, in a simple manner, different gate chains were simulated using 45nm technology [93] as shown in Figure 3.1(a). The simulation was conducted by HSPICE MOSRA with the built-in aging model and combined NBTI and HCI aging effects at a temperature of 25°C. Standard threshold voltage (SVT) INVX1, INVX32, NAND, NOR, and XOR gate chains of different lengths were simulated for up to 2 years of usage. Figure 3.1(a) shows that all chains are experiencing stress from a 500MHz clock. Any other stress (e.g., DC stress which is a constant “0” or “1”, or AC stress with different duty ratios) and usage time could be used in this simulation. Figure 3.1(b) presents the delay degradation caused by 2 years (24 months) of aging. From the figure, we can see that different gate chains age at slightly different rates, which depends on the structure of the gates. The XOR gate chain has the fastest aging rate amongst these chains. Comparing the delay degradation rates of the INVX1 and INVX32 chains, we can conclude that larger gates will age at a lower rate than smaller gates under the same stress. In addition, the workload (input value and the switching frequency of each gate) also has a significant impact on the aging rate. ICs may be recycled from different used boards from different users who may have applied different workloads to the IC at different times. Therefore, it is practically impossible to know the exact input vectors applied by the user. The impact of workload on a chips path delay degradation will be discussed in detail in Section 3.2.



(a) (b)

**Fig. 3.1:** (a) An illustrative circuit with NAND-gate, NOR-gate, XOR-gate, and inverter chains and (b) Delay degradation of the chains.

Figure 3.2(a) shows the delay of a randomly selected critical path $P_i$ (this path includes 22 gates) from the ISCAS'89 benchmark s38417 with stress from a random workload (functional patterns) applied to the primary inputs. The path was aged for 4 years with NBTI and HCI effects at room temperature 25°C. The simulation results indicate that the degradation of path $P_i$ after 1 year is around 10% while if the circuit is used for 4 years, the degradation is about 17%, indicating that most aging occurs at the early usage phase of the design. Therefore, if there are no environmental or process variations, such degradation should provide great a opportunity to identify recycled ICs by measuring one path delay from the circuit. However, these variations have a significant impact on the path delay. On the other hand, different paths age at different rates as demonstrated earlier in this section. Figure 3.2(b) shows the delay of path $P_i$ under different temperatures at different aging times. In the figure, $AT$ denotes aging time, $M$ represents months, and $Y$ denotes years. From Figure 3.2(b), we can see that the delay of path $P_i$ increases as the temperature goes up and paths age with a faster speed at a higher temperature.



(a)

(b)

**Fig. 3.2:** (a) Delay degradation of path $P_i$ and (b) $P_i$ delay increases with increased temperature.

In theory, the specifications and functionality of different ICs of the same design should be identical. In practice, this is not the case due to uncontrollable manufacturing variations that limit our ability to accurately create IC structures at smaller technology nodes. Thus, values such as the threshold voltage of a transistor ($V_{th}$), the gate length of the transistor ($L$), and the oxide thickness of the transistor ($T_{ox}$) can only be guaranteed to be within some ranges. Variations on these and other parameters are important to take into consideration because these parameters directly affect device performance. For example, the delay of a traditional CMOS inverter can be expressed in two equations (3.1) and (3.2) [94], where the high-to-low and low-to-high propagation delays $t_{PHL}$ and $t_{PLH}$ are affected by variation in the three previously mentioned parameters, as shown in (3.3) and (3.4).

$$t_{pHL} = ln(2)R_{eqn}C_L \tag{3.1}$$

$$t_{pLH} = ln(2)R_{eqp}C_L \tag{3.2}$$

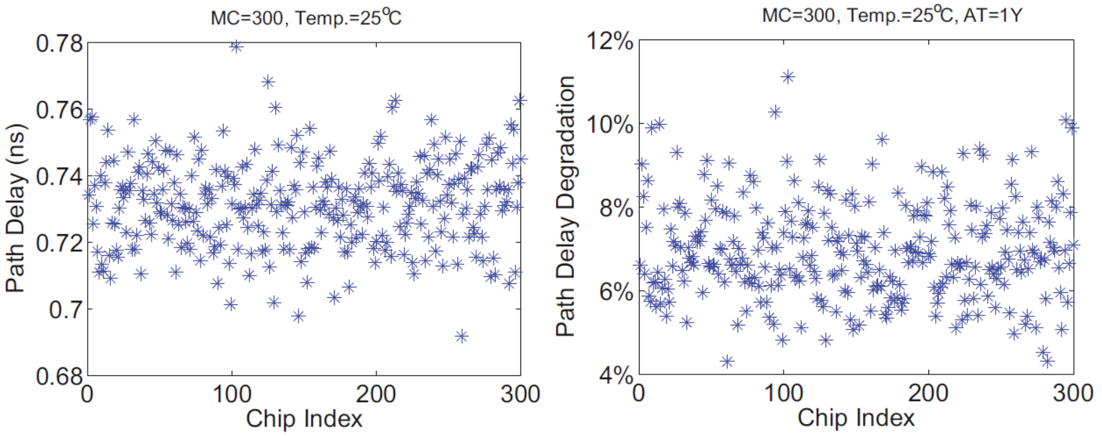
with

$$R_{eq} = \frac{1}{V_{DD}/2}\int\limits_{V_{DD}/2}^{V_{DD}} \frac{V}{I_{DSAT}(1+\lambda V)}dV \approx \frac{3}{4}\frac{V_{DD}}{I_{DSAT}}(1-\frac{7}{9}\lambda V_{DD}) \tag{3.3}$$

and

$$I_{DSAT} = k'\frac{W}{L}((V_{DD}-V_{th})V_{DSAT} - V_{DSAT}^2) \tag{3.4}$$

To analyze the impact of process variations on a path $P_i$'s delay, Monte Carlo simulations were performed using HSPICE on s38417. 300 Monte Carlo simulation results of $P_i$ at 25°C are shown in Figure 3.3(a), with 3-sigma 2% $T_{ox}$, 5%$V_{th}$, and 5% $L$ inter-die and 1% $T_{ox}$, 5%$V_{th}$, and 5% $L$ intra-die process variations. It shows that $P_i$'s delay varies around 12% due to process variations. In addition, process variations also have a significant impact on the aging rate of the path delay, as shown in Figure



(a) (b)

**Fig. 3.3:** (a) Delay degradation of path $P_i$ and (b) $P_i$ delay increases with increased temperature.

3.3(b). $P_i$'s delay degradation in the 300 ICs varied around 8% (4%∼12%) for 1 year of aging. *These variations evidently make the detection difficult, thus, the path delay shifts caused by aging effects in recycled ICs must be separated from those caused by process variations in fresh ICs.*

## 3.2 Path Delay Fingerprinting Considering Aging

The clock sweeping technique has been described in Section 2.1. Figure 3.4 shows the flow for identifying recycled ICs using path delay fingerprints and statistical analysis. The flow is divided into three major steps. First, paths are simulated and selected according to their aging rate. Next, the delay information of these paths is measured by a clock sweeping technique in fresh ICs (either during manufacturing test on all ICs or during authentication on a sample of fresh ICs) and in any chip under authentication (CUA). Finally, statistical analysis is used to decide whether the CUAs are recycled ICs or not.

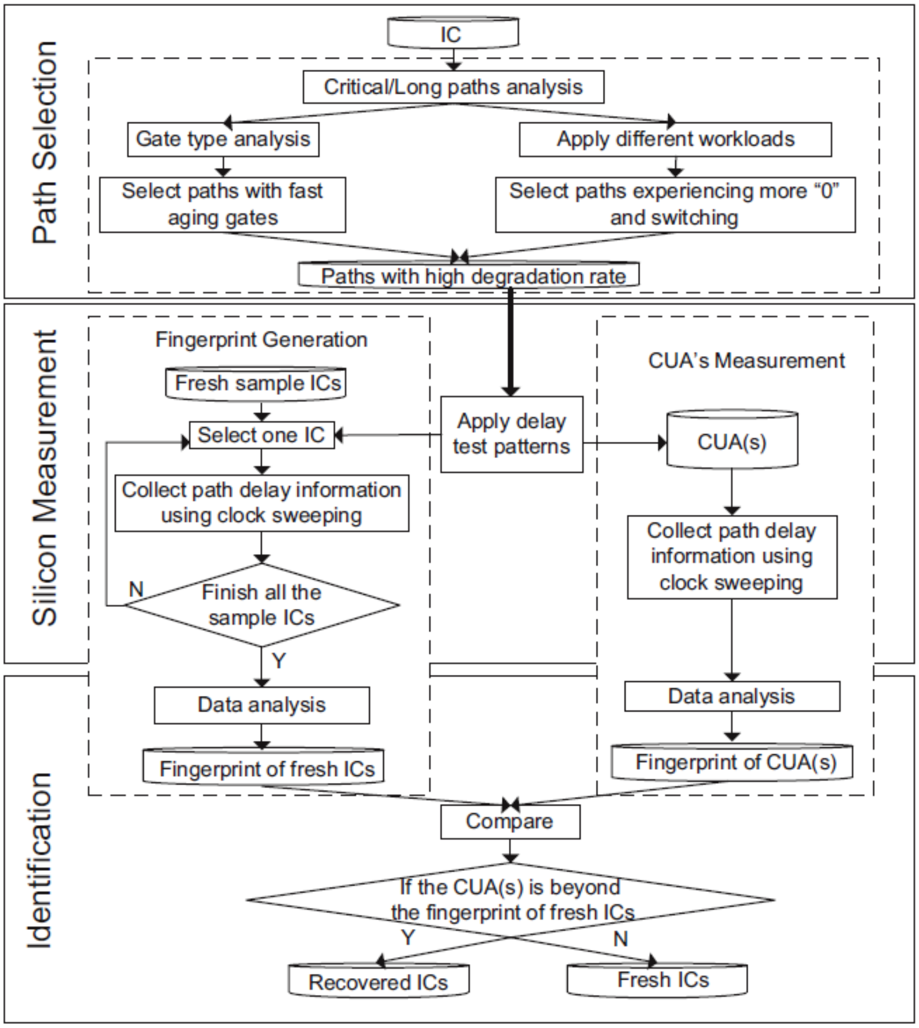

Path Selection
IC
Critical/Long paths analysis
Gate type analysis
Apply different workloads
Select paths with fast aging gates
Select paths experiencing more "0" and switching
Paths with high degradation rate
Silicon Measurement
Fingerprint Generation
Fresh sample ICs
Select one IC
Collect path delay information using clock sweeping
N
Finish all the sample ICs
Y
Apply delay test patterns
CUA's Measurement
CUA(s)
Collect path delay information using clock sweeping
Data analysis
Fingerprint of fresh ICs
Data analysis
Fingerprint of CUA(s)
Identification
Compare
If the CUA(s) is beyond the fingerprint of fresh ICs
Y
N
Recovered ICs
Fresh ICs

**Fig. 3.4:** Recycled IC identification flow.

### 3.2.1 Path Selection

Due to the large number of critical and long paths in a circuit, the first step is to select paths which age at faster rates by analyzing the gate types in different paths and simulating the circuit with different workloads. Paths with higher rates of aging are preferred for delay fingerprint generation, since the differences in the delay of those paths between recycled ICs and fresh ICs will be much larger than the differences in paths, which degrade more slowly. Delay fingerprints generated by fast-aging paths could help identify recycled ICs used for a shorter time. However, there are several parameters impacting the aging rate of a path, including the type of gates composing the path and the workload (as discussed previously in Section 3.1). Based on these parameters and the observations made from the simulation shown in Figure 3.2, the following rules are developed to select the paths that age at the fastest rates: (*i*) paths with more fast-aging gates, such as NOR or XOR gates, will be selected, and (*ii*) paths that experience more zeroes and more switching activity will be selected. More zeroes in the path will increase the effect of NBTI on the PMOS transistors, and a high switching frequency will increase the HCI effects on gates, increasing the path delay degradation more significantly.

Paths with more fast-aging gates would be identified by analyzing the type of gates composing the paths. However, it is very difficult to identify paths that experience more zeroes and more switching activity without knowing the specific workload. Therefore, in this work, different random workloads (input combinations) are applied to ICs primary inputs during logic simulation. For each gate on a critical path, the average switching activity and the zeroes it has experienced are calculated. Paths with more switching activity and zeroes are then selected using our flow. These paths, along with those composed of the more fast-aging gates, are used to generate path delay fingerprints to identify recycled ICs. The number of selected paths could be adjusted according to the design, its testing procedure, and the desired detection confidence. In our simulations (see Section 3.4.1), the top 50 paths with fast-aging gates are selected and the top 50 paths experiencing more switching activity and zeroes in the benchmark circuit.

### 3.2.2 Silicon Measurement

The second step in Figure 3.4 is to collect the selected paths delay from the ICs. Note that the signature generation can be done during a manufacturing test of a large sample of ICs before shipping them to the market or on a number of fresh ICs from each production kept by the design house for the purpose of authentication or recycled ICs identification. The larger the size of the sample is, the wider the range of process variations that will be included in the signature, reducing the probability of wrongly identifying fresh ICs with large process variations as recycled ICs. Path delay information from the fresh ICs is measured by performing test procedures on the ICs. Traditionally, test patterns are generated by ATPG before fabrication to detect path and transition delay faults. These patterns will be applied to all fresh ICs using clock sweeping techniques, as described in Section 2.1. A path delay fingerprint will be generated for each IC as shown in Table 2.1.

### 3.2.3 Identification

Once the path delay in all sample chips is measured, statistical data analysis will be used to generate a signature for fresh ICs. The more the sample chips are, the more process variations will be covered, reducing the probability that fresh ICs with large process variations will be identified as recycled ICs. For a circuit under authentication (CUA) taken from the market, the same test patterns will be applied in a controlled environment (near-identical to that used for the fresh chips). The path delay information of the CUA will be processed by the same statistical data analysis methods. In a simple analysis, if the signature of the CUA is outside of the range of the fresh ICs' signature, there is a high probability that the CUA is a recycled IC. Otherwise, the CUA is likely a fresh IC. The longer the CUA has been used, the more aging effects it will have experienced, making it easier to identify.

Without extra hardware circuitry embedded into the ICs, our recycled IC identification technique imposes no area or power overheads. It provides a negligible test time overhead during manufacturing test on a sample of ICs, since only a few patterns must be applied several times at different frequencies. Also, there is no change in the current IC design and test flow since there is no additional circuitry in the IC used for detection. In addition, this method is resilient to tampering attacks. It is inherently difficult for recyclers to mask the impact of aging on the recycled ICs' path-delay signatures during the recycling process. The only limitations of this technique are that the design must be known to get the path delay information and signatures from fresh chips are required to compare with the CUAs for classification.

## 3.3 Statistical Data Analysis

Two statistical data analysis methods are used in this paper: simple outlier analysis (SOA) and principal component analysis (PCA) [95]. When performing SOA, the flow randomly selects a single path from the selected path set, and uses its delay range in fresh ICs to generate a signature. The process variations of the CUA may or may not be the same as those within the sample ICs. The selected path delay of the CUA and sample ICs will follow the same distribution, which makes SOA effective in certain conditions. However, a single-path based analysis will not be very effective, due to the limited aging information collected. In general, this method is expected to be effective in distinguishing recycled ICs used for a long time from fresh ICs with small process variations, as demonstrated by our results in Section 3.4. This basic approach is limited by the path that has been selected. One way we could improve this would be to use SOA independently on multiple paths and then make a final decision (recycled or fresh) according to a majority vote.

PCA is a dimensionality reduction technique that transforms the data into a reduced number of dimensions that captures most of the variations in the data and the PCA has been effective for detection of hardware Trojans in prior work [6] [38]. The PCA is applied to this problem as follows: The path delay information of all selected paths, which have been measured by clock sweeping, will be processed by PCA. In the simulations, the top 100 paths with faster aging rates were selected to generate signatures. The delay of each path is one of the variables for PCA to use. Therefore, with $N$ ICs, the dimension of the data set for PCA to generate signature is $N \times 100$. We apply PCA and then plot the first 3 principal components (the three that capture the most variance). A convex hull is created around this data to represent the signature for fresh chips. The path delay information of the CUA was also analyzed by the same process and plotted in the same figure. If the CUA is outside of the convex created by the fresh ICs, there is a high probability that the CUA is a recycled IC.

## 3.4 Experiment Results and Analysis

In order to verify the effectiveness of the recycled IC identification flow and data analysis methods, they were implemented on several benchmarks in 45nm technology. HSPICE MOSRA was used to simulate the effects of aging on the path delay of different benchmarks. The supply voltage of the 45nm technology is 1.1 Volts. Random workloads (random functional input patterns) were applied to several ISCAS'89 benchmarks. The path delay fingerprint was generated using clock sweeping at different aging times. Different process and temperature variations were also simulated to analyze their impact on the effectiveness of our recycled IC identification methods.

### 3.4.1 Process and Temperature Variations Analysis

Table 3.1 shows the three process variations rates that were used in simulations. Moving from PV0 to PV2, both inter-die and intra-die variations become larger. PV1 represents a realistic rate of process variations that a foundry might have. Four sets of Monte Carlo simulation (MCS) were run using different levels of variations, as shown in Table 3.2. For each set of MCS, 300 Monte Carlo simulations were run to generate 300 chips. During the simulations, the aging effects of NBTI and HCI were simulated with random stress for the benchmark s38417. From the top 500 paths, the paths $P_1$, $P_2$,..., $P_{50}$ with fast-aging gates and the paths $P_{51}$,$P_{52}$,...,$P_{100}$ with more zeros and higher switching activities were selected to generate path delay fingerprint as described in Section 2.1.

**Table 3.1:** Process variation rates.

| | Inter-die ($3\sigma$) | | | Inter-die ($3\sigma$) | | |
|---|---|---|---|---|---|---|
| | $V_{th}$ | $L$ | $T_{ox}$ | $V_{th}$ | $L$ | $T_{ox}$ |
| PV0 | 3% | 3% | 2% | 2% | 2% | 1% |
| PV1 | 5% | 5% | 2% | 5% | 5% | 1% |
| PV2 | 8% | 8% | 2% | 7% | 7% | 2% |

**Table 3.2:** Simulation setup.

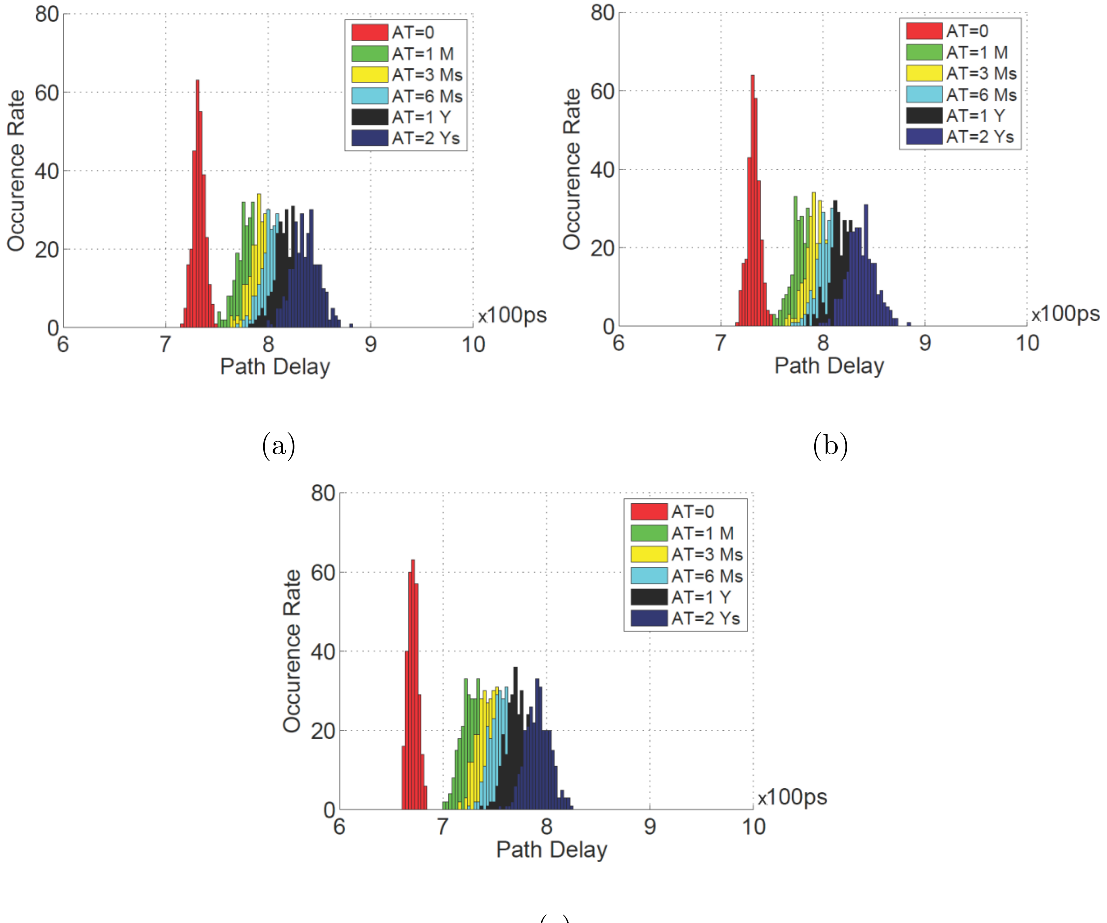
| Experiments | Process Variations | Temperature |
|---|---|---|
| MCS1 | PV0 | 25°C |
| MCS2 | PV1 | 25°C |
| MCS3 | PV2 | 25°C |
| MCS4 | PV1 | 25°C±10°C |

**Analysis using Simple Outlier Analysis (SOA):** First, 300 Monte Carlo simulations were run in MCS1. The maximum aging time is 2 years. Here, SOA was used to process the delay fingerprint. Three paths ($P_1$, $P_2$, and $P_{51}$) were selected to show the results of SOA. Figures 3.5(a), (b), and (c) show the path delay distribution of the three paths from 300 ICs used for different aging times. Similar results were obtained for the other 97 paths as well. For each path, the range of the path delay at AT='0' is the signature of the fresh ICs. Note that M and Y denote months and years of aging respectively. If the path delay of the CUA is out of the range specified at AT='0', there is a high probability that IC is a recycled one. Note the 300 different

Monte Carlo simulations are used for recycled ICs from those used as sample fresh ICs. Figure 3.5 shows that the delay distribution of each path in recycled ICs shifts to the right, relative to the distribution of delays in fresh ICs. This is because the path delay in recycled ICs increases due to aging. The longer the ICs have been used, the more path delay degradation they will have experienced. In addition, Figure 3.5 also demonstrates that the path delay variation increases as the aging time increases. The reason for this is that ICs with different process variations age at different speeds, and the path delay variations become larger as the aging time increases.

Figure 3.5(a) shows the distribution of path $P_1$'s delay. In the figure, the smallest delay of $P_1$ in recycled ICs used for one month is smaller than the largest delay in fresh ICs. Therefore, the detection rate of recycled ICs used for 1 month is less than 100% (98.3%) when the signature generated by SOA from path $P_1$ is used. However, the detection rate of recycled ICs used for three months or longer is 100%, which demonstrates that it is easier to detect recycled ICs that have been used for longer amounts of time. If the path $P_2$ is chosen to detect recycled ICs, the detection rate of ICs used for one month (95.7%) is slightly less than when using path $P_1$. However, if path $P_{51}$ is used, which has the fastest aging rate among the 100 paths, the detection rate is 100% even if the ICs are only used for one month. $P_{51}$ is the most effective path for identifying recycled ICs in this benchmark. From the above analysis, a conclusion is that different paths generate different signatures due to their different aging speeds, which makes SOA slightly less effective.

Figures 3.6(a) and (b) show the delay distribution of path $P_{51}$ across 300 Monte



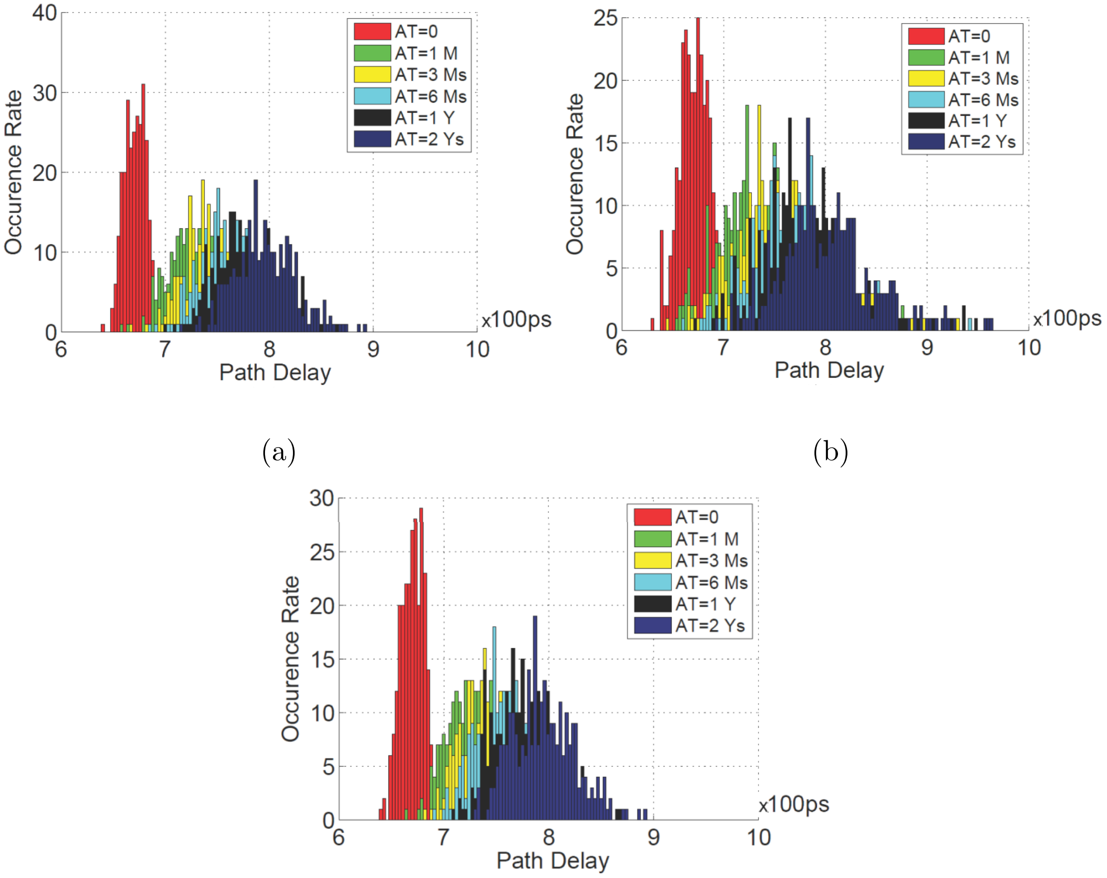
(a)

(b)

(c)

**Fig. 3.5:** Path delay distribution in ICs with PV0 in MCS1 at different aging times (a) Path $P_1$, (b) Path $P_2$, and (c) Path $P_{51}$.

Carlo simulations in MCS2 and MCS3. Overall, Figures 3.5(c), 3.6(a), and 3.6(b) present the delay distribution of the same path ($P_{51}$) in ICs with different process variations. By comparing these figures, it clearly shows that the larger the process variations are, the larger the path delay variations in fresh ICs will be, which makes it more difficult to detect recycled ICs. Even when using the most effective path $P_{51}$, the detection rates of ICs used for one month with PV1 and PV2 drop from 100% with PV0 to 78.0% and 50.7%, respectively. A 100% detection rate could be achieved if the



(a)

(b)

(c)

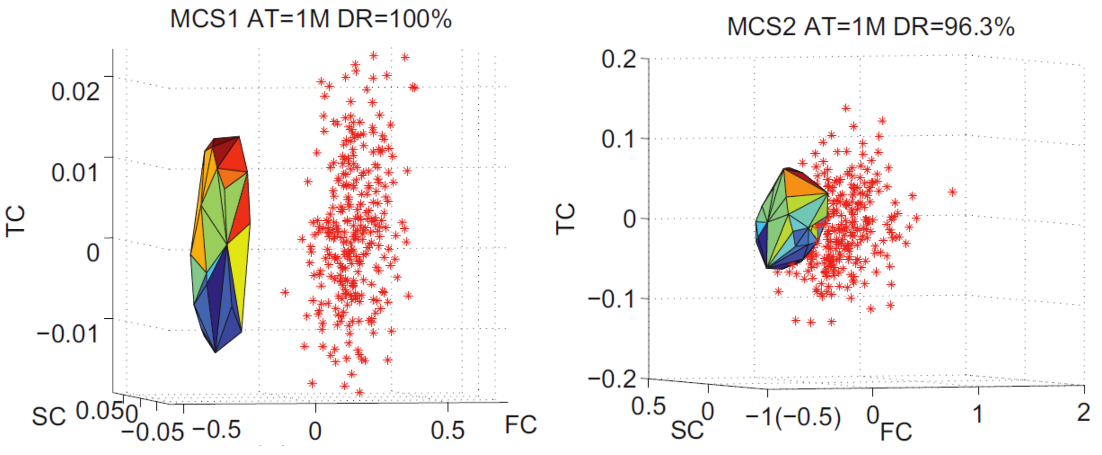
**Fig. 3.6:** Path $P_{51}$ delay distribution in ICs at different aging times (a) in MCS2, (b) in MCS3, and (c) in MCS4.

ICs were used for one year or longer with PV1, or longer than 2 years with PV2.

300 Monte Carlo simulations were also run with ±10°C temperature variation during the aging process in MCS4 as shown in Figure 3.6(c). The measurement temperature is 25°C. It shows the delay distribution of path $P_{51}$ and the detection rate of ICs used for 1 month using it is 67.7%. Comparing Figure 3.6(a) and (c), the result presents that the larger the temperature variation is, the larger the path delay variation

is, which makes it even more difficult to detect recycled ICs.

**Analysis using Principal Component Analysis (PCA):** A similar analysis is done using PCA for different MCSs. Figure 3.7(a) shows the PCA results of the 100 paths in s38417 with 300 chips in MCS1. FC denotes the first component from PCA, SC represents the second component, "TC" is the third component, and DR denotes the detection rate. The convex hull boundary is built up from fresh IC data, and represents the signature for fresh ICs. The red asterisks represent chips used for one month. In the figure, the 300 used ICs were completely separated from the signature of the fresh ICs. Thus, the detection rate using timing signatures generated by PCA is 100% for recycled ICs used for *one month*. For recycled ICs used for a longer time, the detection rate will obviously be 100% as well.



(a)

(b)

**Fig. 3.7:** PCA results of ICs under 25°C (a) used for one month with PV0 in MCS1, (b) used for one month with PV1 in MCS2, and (c) used for three months with PV1 in MCS2.

The path delay information from the remaining three sets of MCSs was also analyzed by PCA. Figure 3.7(b) shows the analysis results of fresh chips and recycled ICs used for 1 month in MCS2. The 3-dimensional figure shows that some of the recycled ICs are close to the fresh ICs signature. The detection rate is 96.3%, which is much higher than using SOA. Comparing Figure 3.7(b) and (a), it shows that (i) the convex hull built up from fresh ICs in MCS2 is much larger than that in MCS1 (note that the convex hull in MCS1 looks larger than MCS2 due to its small scale of axes), and (ii) the recycled ICs in MCS2 are closer to fresh ICs than those in MCS1, which makes the detection rate in MCS2 less than that in MCS1. The path delay information of 300 ICs used for 3 months in MCS2 were also processed, and the results are shown in Figure 3.7(c). Comparing Figures 3.7(b) and (c), it shows that the longer the chips have been used, the farther they will be from the fresh ICs signature. The detection rate of recycled ICs used for *3 months or longer* with PV1 at 25°C is 100%.

Figure 3.8 are the PCA results of ICs in MCS3. The detection rate of recycled ICs used for 1 month, 3 months, 6 months, and 1 year are 72.7%, 89.3%, 99.3%, and 100%, respectively. The figures of PCA results of recycled ICs used for 1 month and 3 months are not shown here since the detection rates are so far from 100%. Figures 3.8(a) and (b) show the fresh ICs' signatures and the recycled ICs used for 6 months and 1 year, respectively. The recycled ICs used for longer times are easier to detect, as seen by comparing Figures 3.8(a) and (b). The detection rates in these simulations prove that it is more difficult to detect recycled ICs that have higher levels of process variations. The 99.3% detection rate of ICs used for 6 months and the 100% detection rate of ICs used for *1 year* in MCS3 shows the effectiveness of the detection technique. In addition, PV2 is an extremely high variation compared to what is expected in practice (e.g., PV1).

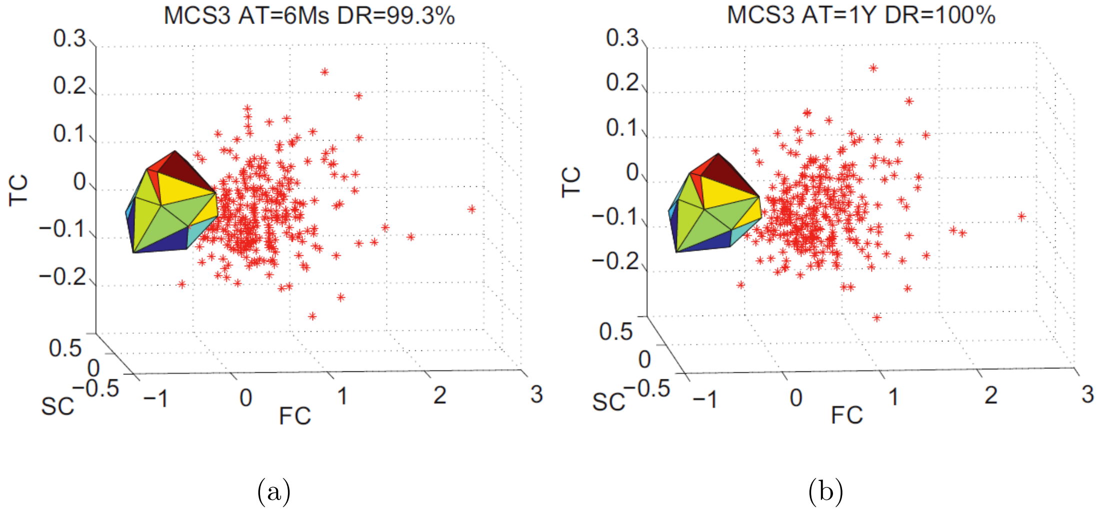
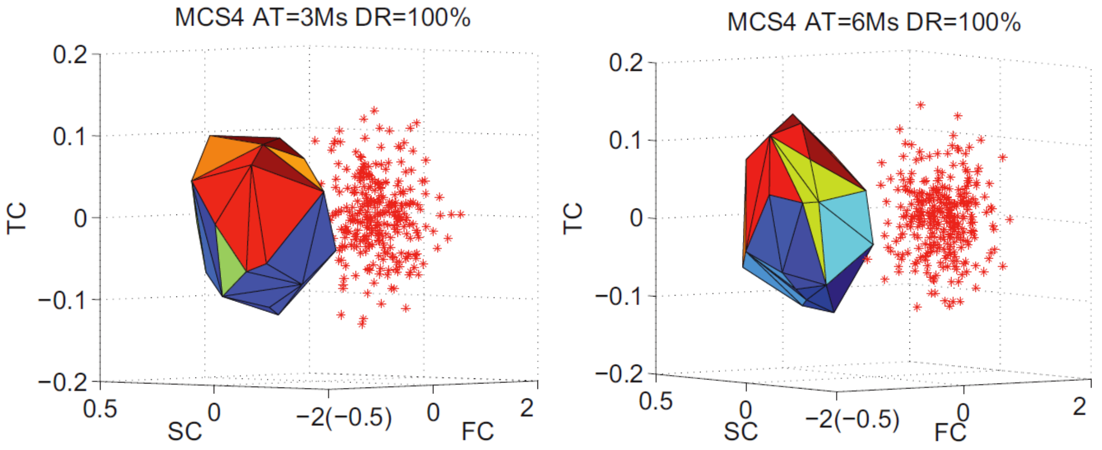


**Fig. 3.8:** PCA results of ICs with PV2 under 25°C in MCS3 used for (a) six months and (b) one year.

With the same measurement temperature 25°C, ±10°C temperature variation is used in MCS4 during the aging process. The detection rate of ICs used for 1 month, 3 months, and 6 months in MCS4 are 90.6%, 100%, and 100%, respectively. The fresh ICs' signature and the detected recycled ICs used for 3 months and 6 months are shown in Figure 3.9. Comparing Figure 3.9(a) with Figure 3.7(c), it clear shows that the recycled ICs used for 3 months in MCS4 are closer to the signature than recycled ICs used for 3 months in MCS2. This phenomenon demonstrates that temperature variations could increase the path delay variations in fresh ICs and make it more difficult to detect recycled ICs. However, the 100% detection rates of ICs used for 6 months in MCS4 demonstrates the effectiveness of our method with process and temperature variations.

**Comparison between SOA and PCA:** Figures 3.5 through 3.9 present some



(a)

(b)

**Fig. 3.9:** PCA results of ICs with PV1 and ±10$^{c}irc$C temperature variations in MCS4 used for (a) three months and (b) six months.

detailed results relating to using this technique on s38417 with SOA and PCA. Table 3.3, however, tabulates these results in addition to some other results obtained using both statistical analysis approaches. These results clearly show that PCA is more effective than SOA when it comes to identifying ICs used for shorter periods of time.

**Table 3.3:** Recycled IC detection rates for s38417.

| | SOA | | | | PCA | | | |
|---|---|---|---|---|---|---|---|---|
| | 1M | 3M | 6M | 1Y | 1M | 3M | 6M | 1Y |
| MCS1 | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| MCS2 | 78% | 96.7% | 99.7% | 100% | 96.3% | 100% | 100% | 100% |
| MCS3 | 50.7% | 76.3% | 85.3% | 95.6% | 72.7% | 89.3% | 99.3% | 100% |
| MCS4 | 67.7% | 93.3% | 98% | 100% | 90.6% | 100% | 100% | 100% |

### 3.4.2 Benchmark Analysis

In addition to s38417, the ISCAS'89 benchmarks s9234 and s13027 were also simulated to demonstrate the efficiency of this technique on different designs. The process variation and temperature variation rates used in MCS4 were applied to these two benchmarks. The aging stress causing NBTI and HCI degradation in these benchmarks comes from random workloads. 300 MCS were run for each benchmark for a maximum 2 years of aging. The path selection method was also applied to these benchmarks, and 100 paths from each benchmark were used to run statistical data analysis using PCA.

Table 3.4 shows the recycled IC detection rate for all three benchmarks under MCS4 for up to a year of aging. The detection rate for ICs used for 3 months in the benchmarks s9234 and s13207 is 100%, which matches the results obtained from s38417.

**Table 3.4:** Recycled IC detection rates - Benchmark comparison under MCS4 using PCA.

| Benckmark | 1M | 3M | 6M | 1Y |
|---|---|---|---|---|
| s9234 | 88% | 100% | 100% | 100% |
| s13207 | 89.6% | 100% | 100% | 100% |
| s38417 | 90.6% | 100% | 100% | 100% |

The results shown in this section clearly demonstrate that the recycled IC detection method using CTS with PCA is very effective, even in designs with large process and temperature variations.

## 3.5 Conclusion

In this chapter, we present a recycled IC identification method using path delay fingerprinting. Paths with fast aging speed were selected to generate a fingerprint for the chip under authentication. The path delay signature from a recycled IC is beyond those from new ICs due to aging. With no additional hardware circuitry required, this method introduces no overhead on area and power consumption. The simulation results of different benchmarks with different process and temperature variations demonstrate the effectiveness of this method.

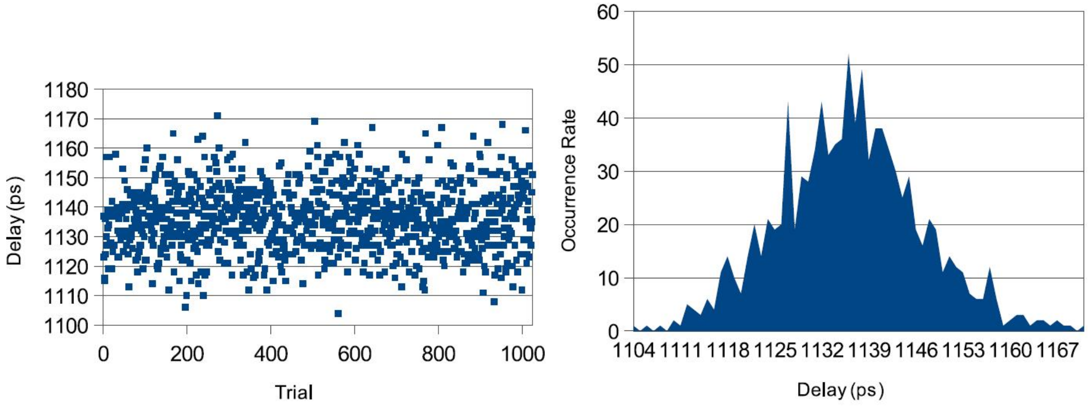
# Chapter 4

# Post-Silicon Test Methodology: Delay-based IC Fingerprinting for On-Chip ID Generation

This chapter will introduce a post-silicon test methodology to generate identifiers for fabricated ICs through path delay analysis. This technique has no overhead in terms of area, timing, or power, since it can extracts the intrinsic path delay variation information of the IC. An ID creation approach is proposed in this chapter. In addition, several optimization techniques are developed to improve the stability of IDs.

## 4.1 Process Variation on Path Delay

In theory, the specifications and functionality of different ICs of the same design should be identical. In practice, this is not the case. This happens because of our inability to accurately create IC structures at smaller technology nodes, due to process variations. Thus, values such as the threshold voltage of a transistor ($V_{th}$), the length of the gate of the transistor ($L$), or the oxide thickness of the transistor ($T_{ox}$) can only be guaranteed to be within some range of values. Variation on these and other parameters is important to take into consideration, because these parameters directly affect device performance. For example, the delay of a traditional CMOS inverter can be expressed in two equations, where the high-to-low and low-to-high propagation delays $t_{PHL}$ and $t_{PLH}$ are effected by variation on the three previously mentioned parameters.



(a) Scatter plot

(b) Distribution plot

**Fig. 4.1:** Path delays for 1,024 simulations of the most critical path in s38417.

Figures 4.1(a) and (b) show how process variations can affect the delay of a path in a set of ICs. We measured the delay of the most critical path in the ISCAS'89 benchmark s38417 1,024 times at 25 degree Celsius using HSPICE. Differing levels of process variations between the Monte Carlo simulations resulted in the path having a different propagation delay in each simulation. Figure 4.1(a) shows the path delay for each simulation, and Figure 4.1(b) shows the distribution of these path delays. These delays are centered around an average value because process variation results in small changes to the parameters which control the path delay.

## 4.2 Methodology

A methodology to create a unique binary identity (ID) for each IC using IC fingerprinting will be introduced in this section. The unique IDs are generated from intrinsic timing characteristics and are difficult to pirate and clone, so they can be used to address other counterfeit types that introduce new ICs, such as cloned, overproduced, and remarked. The procedure for uniquely identifying ICs with IDs can be broken down into two mains parts: IC enrollment and IC identification.

### 4.2.1 IC Enrollment

When the clock sweeping tests presented in Chapter 2.1 have been applied in the manufacturing test stage, IC fingerprint is obtained and can be utilized to generate a binary ID for each IC. Three steps are performed to enroll every manufactured IC:

1. **Stability checking:** a path selection procedure that selects the most stable measured paths from each IC.
2. **ID generation:** a path delay comparison procedure which generates an unoptimized ID for each IC.
3. **ID optimization:** an analysis and optimization procedure which increases the randomness and decreases the size of the IDs. The optimized ID will be stored in database for later access and authentication.

More details concerning the above steps are discussed in the forthcoming subsections.

#### ID Generation and Optimization

We are using path delay information to generate IDs. Path delay measurements using clock sweeping do have the potential to be affected by measurement and environmental noise. To reduce the effect of this issue, we will attempt to discard unstable paths early

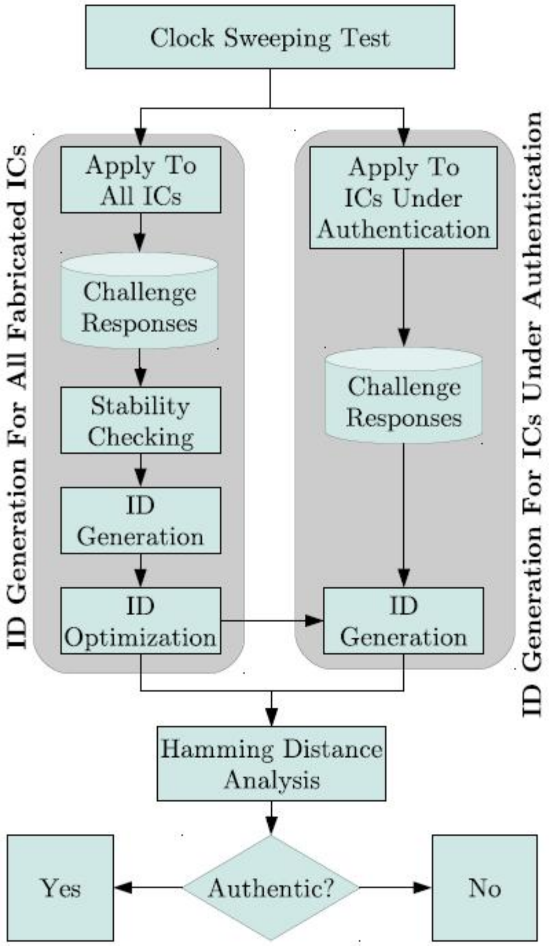


**Fig. 4.2:** Test generation and authentication flow.

on. Measuring an IC multiple times gives us multiple measurements of each path in that IC. Analysis of multiple measurements of a path can uncover whether that path is stable-it reported the same delay during each measurement- or unstable-it reported different delays on different occasions. To select stable paths, we will measure some subset of the IC set multiple times and select the $m$ most stable paths to be used for further analysis on all $n$ ICs.

### ID Generation

Once the $m$ most stable paths have been selected, we begin to generate IDs for each IC. To generate an ID, we will perform comparisons between the delays of different paths from the ICs. The goal is to perform a comparison that is somewhat uncertain, so that the result the comparison generates is random. We obtain a list of similar paths by sorting a list of $m$ paths from an IC in ascending or descending order. We will call this sorting the *golden ranking*, and apply this ordering to the $m$ paths of all $n$ ICs. Once the golden ranking has been obtained, the path delay data for every IC is sorted by this ordering. For each IC, we will traverse this ordered list of delays one element at a time. Whenever a delay in the list is greater than the next delay in the list, we will append a '1' onto the ID for that IC. Whenever a delay in the list is less than the next delay, we will append a '0' onto the ID for that IC. If $m$ paths have been analyzed for each of the $n$ chips, we will end up with $n$ IDs of $m - 1$ bits each. The first/golden IC, which was used to determine the ordering, will have an ID of all ones or zeros, depending on whether the sort was ascending or descending.

### ID Optimization

There are several optimization techniques we can use on these ICs. Some paths that we are comparing in the ID Generation step may produce the same value for all ICs because the path delays are very different. This will reduce the average Hamming distance between IDs as this bit will be the same in all ICs. If enough paths have been measured, it may be possible for an IC designer to pick and choose which comparisons they are going to perform to create their IDs. The comparisons can be chosen by analyzing each bit of the m-1 bit “unoptimized IDs”, and deciding which bits have the least bias across all IDs. A bit should be included in the “optimized ID” if the number of zeros in that bit position is close to 50% of the number of IDs-if the number of zeros in that bit position is in the range of $\frac{n}{2} \pm \varepsilon$, where $\varepsilon$ is some small constant. This process selects the most unpredictable comparisons across the IC data set. By performing this optimization technique, we obtain with shorter, more random IDs, at the cost of discarding some of the data obtained during the measurement process.

### 4.2.2 IC Identification

This process will be run on every IC when they are first manufactured and going through manufacturing tests. These tests will produce an ID for every manufactured IC, will will be stored in a database for later access. Parts of this process will be run again later when we wish to identify an IC. Figure 4.2 shows these two scenarios. In order to identify an IC from the market, referred to as an IC under authentication (IUA) in Figure 4.2, we will need: (1) the ID that the IC is presenting, and (2) the database of IDs from when the ICs were manufactured. Once we have the ID for the IUA, we can check to see if it is in the database using Hamming distance analysis. Hamming distance analysis is done by comparing the ID from the IUA to every ID in the database. If the ID from the IUA is an exact match to one of the IDs in the database, then we have identified the CUT.

Noise in the measurement process might make it so that there are a small number of bits in the ID which are different between the IC’s ID in the database and its ID in the field. We addressed this issue through the stability checking technique described in Section **??**, but differences may still be present. Therefore, we define a *Hamming distance threshold*, based on analysis of the IDs during manufacturing tests. This threshold could be based off the minimum Hamming distance between two different ICs in the manufactured set. The threshold could also be based off of the maximum Hamming distance between IDs from the same IC, which would be created during the stability checking process. Exact thresholds would be derived from analysis of the ID set and measurement noise rates.

### 4.2.3 Overhead Analysis

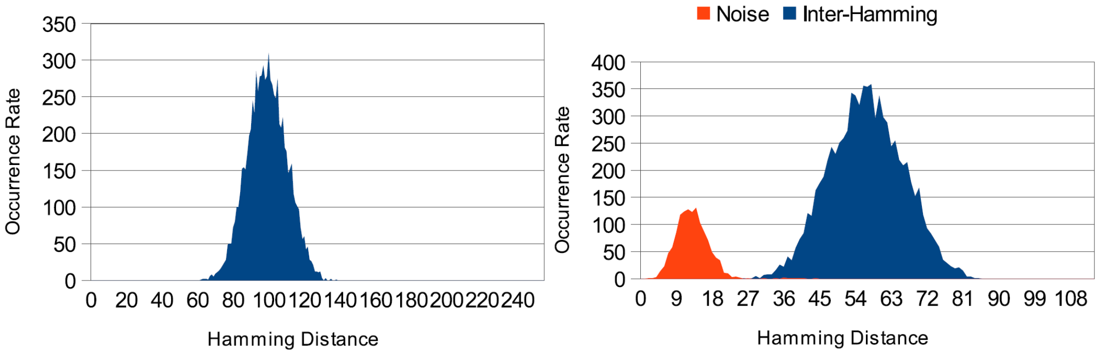
This IC identification procedure incurs no area, power, or timing overhead. However, there is an overhead in testing time and computational time. Each IC must undergo an additional $k$ rounds of TDF testing at $k$ different frequencies. If there are $n$ ICs with $P_{f_1}$ TDF patterns and $N_{sff}$ flip-flops that will be tested at $k$ frequencies, the worst case increase in testing time would be $n \cdot (k-1) \cdot P_{f_1} \cdot N_{sff} + N_{sff}$ clock cycles. This is a worst-case scenario as not all $P_{f_1}$ TDF patterns will need to be tested at all $k$ frequencies, as shown in Figure 3(a). The stability checking procedure requires test engineers to measure a subset of $n_r \ll n$ ICs $j$ times each, so there is a testing overhead of $n_r \cdot j \cdot (k-1) \cdot P_{f1} \cdot N_{sff} + N_{sff}$ clock cycles in the worst-case scenario. Judging the relative stabilities of each path in the ICs requires test engineers to individually analyze each of the $m$ paths that were measured $j$ times each in $n_r$ ICs, resulting in a complexity of $O(n_r m j)$ time; however, $n_r$ is much less than $n$, and $j$ is going to be on the order of dozens to hundreds. The ID generation procedure requires test engineers to individually analyze each of the $m$ paths in each of the $n$ ICs, resulting in an $O(nm)$ runtime. The ID optimization technique analyzes each of the $m1$ bits across all of the $n$ IDs for a general runtime of $O(nm)$ as well. In general, $m$ will be much smaller than $n$ or $n_r$.

## 4.3 Results and Analysis

### 4.3.1 Simulation Results

To evaluate this methodology, a series of simulations in HSPICE were performed on an implementation of the ISCAS'89 benchmark s38417. This circuit was simulated at the 90nm technology node. To measure the path delay, we identified the 256 top critical paths from the circuit. and performed 128 Monte Carlo simulations. This provides us with 128 simulated ICs worth of data, with 256 data points for each IC. During the Monte Carlo simulations, the threshold voltage $V_{th}$ had at most 5% inter-die variation and 5% intra-die variation, the oxide thickness $T_{ox}$ had at most 2% inter-die variation and 1% intra-die variation, and the transistor gate length $L$ had at most 5% inter-die variation and 5% intra-die variation, with these values representing 3-sigma deviations from specified values. All paths for all ICs were simulated over a range of temperatures from 23°C to 27°C to represent small deviations from room temperature.

Using the ID generation methodology described in Figure 4.2, 128 255-bit IDs were created from the simulation data. Figure 4.3(a) shows the results of Hamming distance analysis on these unoptimized IDs. The 255-bit IDs have, on average, a 99-bit or 39% inter-Hamming difference. The fact that this average is less than 50% means that some bit-positions in the IDs have a bias towards zero or one. By optimizating the IDs to remove these bits, using the optimization procedure described in Section **??**, we improve the average Hamming distance of the ID set and reduce the size of the IDs, as shown in Figure 4.3(b). The optimization process reduced the size of the IDs from 255 bits to 114 bits, and increased the average inter-Hamming distance from 39% to 50%.



(a) Unoptimized IDs

(b) Optimized IDs with noise

**Fig. 4.3:** Hamming distance analysis on 128 simulated s38417 circuits.

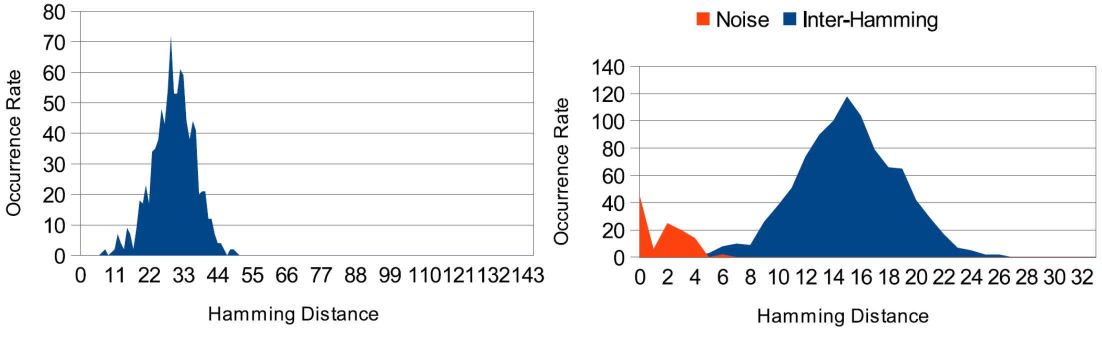
Figure 4.3(b) also shows the noise rates of the simulated IDs. Each IC was simulated at temperatures from 23°C to 27°C at 1°C increments to represent small variations around room temperature. Hamming distance analysis was performed on each set of IDs coming from the same IC to see how many bits of the ID would change over the ID range. The average number of bits changing as a result of temperature was 13 bits, or about 11%.

The Hamming distance distribution and noise rate distribution are both roughly normal distributions. By treating them both as normal distributions with their own averages and standard deviations, we can come up with probability density function (PDF) for each set. We can use these two PDFs to find the Hamming distance at which it is equally likely that the two IDs are from different ICs or from the same IC. In the sets shown in Figure 4.3(b), this Hamming distance is 27 bits. If two IDs are compared and have less than 27 different bits between them, they are most likely from the same IC. If two IDs are compared and have more than 27 different bits between them, they are most likely from different ICs.

### 4.3.2 FPGA Implementation Results

This methodology was also evaluated on 44 90nm Xilinx Spartan-3E FPGAs. The ISCAS'89 benchmark s9234 was implemented on our FPGAs, along with RAM for TDF pattern storage and structures for clock control. The IDs were sent from the FPGA boards to a computer for analysis by using an additional microcontroller. The paths in the FPGA were swept at the frequencies $f_{1-16}$, with $f_1 = \frac{1}{6.4ns}$ and $f_{16} = \frac{1}{3.4ns}$. The frequency step size was 200ps. To obtain noise information, four of the implementations were measured eight times each. After selecting stable paths, the unoptimized IDs were 145-bit strings, with a Hamming distance distribution as shown in Figure 4.4(a). The average inter-Hamming distance was 30 bits, or about 21%. This is lower than in the simulation example. The optimization process reduces the size of the ID from 145 to 33 bits long, and the distribution of the inter-Hamming distances between these 33-bit IDs are shown in Figure 4.4(b). By optimizing the IDs, the average inter-Hamming distance is changed from 30 out of 145 bits to 15 out of 33 bits, or about 45%. In practice, a 33-bit ID would be short, but creating IDs of a greater size only requires measuring more paths.

Figure 4.4(b) also shows the noise rates for these IDs. Four different implementa-



(a) Unoptimized IDs

(b) Optimized IDs with noise

**Fig. 4.4:** Hamming distance analysis on 44 FPGA Implementations of s9234 circuit.

tions of s9234 were measured 8 times each and the IDs from each implementation were compared to each other. On average, there was a 2 out of 33 bit difference between IDs from the same implementation, or about a 6% difference.

Again, we can perform further analysis to compare the noise and inter-Hamming rates. At a difference of 5.75 bits, it is equally likely that any two IDs being compared are from the same IC or from different ICs. IDs with a difference of less than 5.75 bits are probably from the same IC and IDs with a difference of more than 5.75 bits are probably from different ICs.

## 4.4 Conclusion

In this chapter, we present a technique that allows us to uniquely identify ICs without any area, power, or timing overhead. In both the simulation and implementation results, we were able to create unique IDs with no collisions for a set of different implementations of the same circuit. In both cases, the average Hamming distance between the IDs was nearly 50%, and the levels of noise were sufficiently low that we could distinguish between

IDs from the same IC and IDs from different ICs. These results would improve with a larger data set, a more accurate clock with a smaller frequency step size, and larger levels of process variations.

# Chapter 5

## Design-for-Security Methodology: Built-In Self-Authentication to Prevent Hardware Trojan Insertion by Untrusted Foundry

The post-silicon test techniques can address some hardware security problems, but their coverage are limited by circuit structures. Sometimes, design-for-security methodology might be a better solution to deal with security issues. Typically, design-for-security methods need additional circuitry, so area overhead is a big concern. In this chapter, we propose a built-in self-authentication (BISA) and an obfuscated built-in self-authentication (OBISA) technique which introduce no area overhead and can be applied to conventional ASIC/SoC layout design flow. The BISA is able to prevent hardware Trojan insertion by untrusted foundry (Chapter 5), while OBISA can further obfuscate design and consequently protect the original design from reverse engineering through split manufacturing process (Chapter 6).

Chapter 2 presents a side-channel approach to detect hardware Trojans. In general, side-channel analysis methods do succeed in detecting Trojans to some degree, but the difficulty lies in achieving high coverage of every gate or net and in extracting the tiny, abnormal side-channel signals of hardware Trojans from the large background signals, especially in the presence of process and environmental variations. As the feature size of ICs shrinks, the increasing levels of process variations can easily mask the small side-channel signals induced by low-overhead and rarely-triggered Trojans. In order to enhance controllability and observability of nets, some design-for-trust techniques have been developed. [35] [50] attempt to increase controllability or observability of nets in circuit and generate patterns that can create switchings on those hard-to-activate nets. Proposed design-for-trust techniques [50] [52] require an alteration to the original design and the addition of more circuitry during the front-end design phase. As the size of a circuit increases, the number of quiet (low controllability/observability) nets/gates will increase the complexity of processing and produce a large time/area overhead. Thus, such techniques are still difficult to apply to a large design that contains millions of gates. Finally, all these techniques require a golden IC or a golden model that characterizes the correct behavior of the IC under test. However, obtaining a golden IC would be extremely challenging if not impossible in many scenarios. In this chapter, we take an alternative approach that focuses on the hindering hardware Trojan gate insertion by untrusted GDSII developers and untrusted foundry. Note that parametric Trojans [97] do not need any additional circuitry and thus is out of scope of our work.

This chapter will propose a technique that prevents the insertion of additional Trojan gates in circuit layout and mask. The principal idea is to fill *all* unused spaces with functional standard cells, instead of non-functional filler cells, during layout design [75–77]. The inserted standard cells are then connected to form a circuitry called built-in self-authentication (BISA) circuit, which is independent of the original circuit. The BISA structure can be used to test the functionality of its inserted cells. If any of the inserted cells are replaced or otherwise modified, the BISA test procedure will be able to detect it. The main advantage of BISA is that it has *no golden chip requirement*. In contrast, most detection approaches need golden chips either fabricated by a trusted foundry or verified to be Trojan-free through reverse-engineering, both of which are prohibitively expensive. Since BISA relies on logic testing, process variation is not a factor either, as compared to Trojan detection techniques based on side-channel analysis. As an additional advantage, BISA has a negligible impact on original design in terms of area and power. To demonstrate the feasibility of BISA, we have implemented it on several benchmarks (large and small).

## 5.1 Built-In Self-Authentication

Placement tools are typically conservative, aiming to limit density and spread cells evenly to assure routability. This often leaves small gaps between cells, and it is impossible for EDA tools to fill 100% of the area with regular standard cells in VLSI designs. In practice, after completing placement and routing, all unused spaces will be filled with filler cells or decoupling capacitor (decap) cells; these cells do not have any functionality. For intelligent attackers, the most covert way to insert Trojans in a circuit layout is by replacing filler cells because removing these non-functional filler cells does have the smallest impact on electrical parameters. If attackers redesign the original layout for Trojan insertion, moving gates' locations and altering wire interconnections will result in significant changes of the electrical parameters, such as power and path delay. These can be detected much more easily by delay-based and power-based techniques [5,39,40].

The principal idea of BISA is to fill all unused spaces with functional standard cells, called BISA cells, instead of conventional non-functional filler cells. BISA cells are connected to each other to construct a combinational circuit that is independent from the original circuit. This combinational circuit can have an arbitrary function. By testing the function of the BISA circuit, test engineers are able to check whether these cells have been altered after fabrication. If the adversary inserts a Trojan by changing, removing or modifying any cell in a BISA circuit, the designer can easily detect it using structural test. To cover all BISA cells, we try to construct a combinational BISA circuit without redundant gates; in other words, all gates in the BISA circuit are testable by test patterns that target stuck-at faults. For a design with BISA, Trojan cell insertion will become practically impossible without tampering with the BISA cells. Furthermore, BISA cells are of the same type of standard cells as in the original circuit, so identifying these cells will be extremely challenging if not impossible.

The BISA structure is intended to detect any abnormal alteration on BISA by producing a different signature than expected. As to how to test the BISA circuit with low overhead, our inspiration came from the logic built-in self-testing (BIST) for VLSI circuits. The BIST was developed to apply random test sequences to determine the correctness of a circuit by examining a signature obtained from the test responses [98] [99]. Basically, BISA consists of three parts: the BISA circuit under test, the test pattern generator (TPG), and the output response analyzer (ORA), as shown in Figure 5.1(a). The BISA circuit under test is composed of all BISA cells that have been inserted into unused spaces during layout design. In order to increase its stuck-at fault

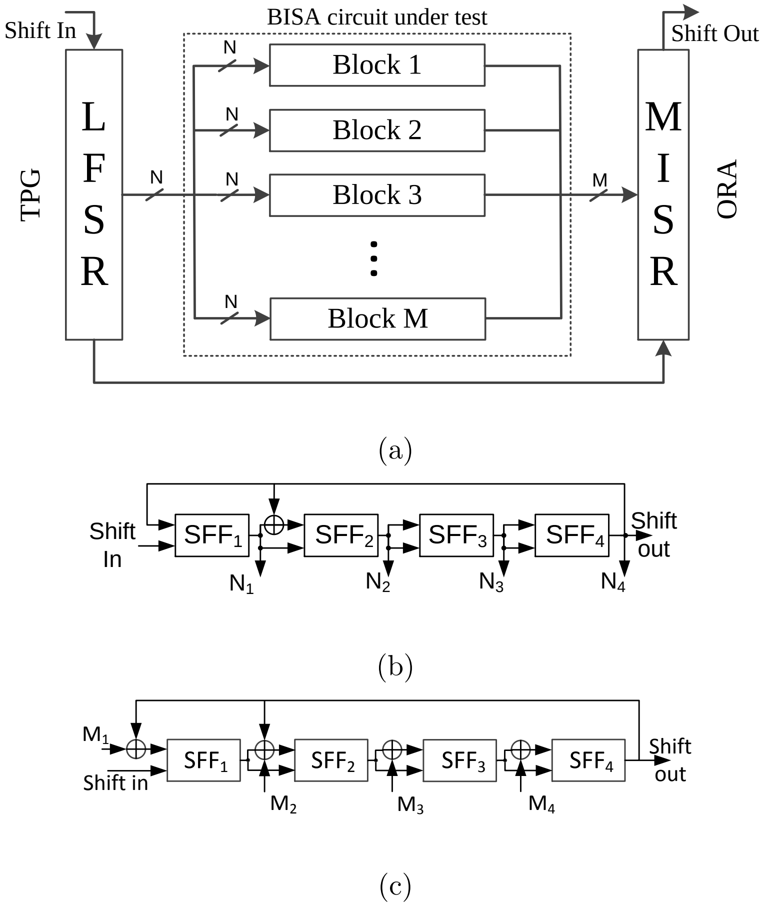


**Fig. 5.1:** The Structure of (a) BISA, (b) a 4-stage LFSR, and (c) a 4-stage MISR.

test coverage, the BISA circuit is divided into a number of smaller combinational logic blocks, called BISA blocks shown in Figure 5.1(a). Each BISA block can be considered as an independent combinational logic block. The TPG generates test vectors that are shared by all BISA blocks. The ORA will process the outputs of all BISA blocks and generate a signature. In this chapter, we use a LFSR as TPG and MISR as ORA. Examples of 4-stage LFSR and 4-stage MISR are shown in Figure 5.1(b) and (c). They are used to generate random vectors and compress responses to a signature. SFF in the figure represents scan flip flop. Other types of TPG and ORA can also be applied [99].

There are two operating modes associated with the chip, as shown in Table 5.1. In functional mode, the original circuit is working normally, but the BISA is in idle mode by disabling LFSR and MISR. BISA stays quiet and does not affect the original circuits, but can fulfill the role of decap cells (see Section 5.3.2). Authentication mode consists of two phases. During the test phase, the LFSR generates N-bit test patterns at every clock cycle and the N-bit test patterns are shared by all BISA blocks. At the same time, each BISA block outputs one bit so that the MISR receives a total of M bits from M blocks (see Figure 5.1(a)). The test phase ends until a sufficient number of test patterns have been applied. The number of test patterns is decided in simulation at the BISA design phase, according to a target test coverage. At this time, the value in the MISR is the signature generated from the responses of BISA blocks and it can be shifted out through scan chain. We refer to this as the shift phase. Comparing the obtained signature on a fabricated chip with the correct signature from previous simulations, we would then know whether the BISA structure has been tampered with. Since all the registers in LFSR/MISR and other registers in circuit are connected in a chain, in the shift phase, the signature in MISR can be shifted out while, at the same time, the new seed for LFSR is shifted in if more patterns need to be applied.

In BISA, two control signals are needed to manage the system and trigger the authentication stage. An authentication mode selection (AMS) signal from the primary is required to choose between functional mode and authentication mode (as depicted in Table 5.1). Therefore, one extra input pin for this signal is required. In the authentication mode, the system needs to switch between shift and test phase. The test mode selection (TMS) signal generated in design-for-test for controlling scan chain can be used here as well, so no additional pin is added. Note that an external clock is applied in all modes. A particular clock will be generated by a tester or other external clock generators for every mode of BISA. Typically, shift clock and authentication clock are much slower than functional clock [99]. The authentication clock frequency is related to the number of gates in a BISA block and the gate types. As the gate number goes up, the depth of the tree structure circuit increases. After BISA design, the depth of every tree structure circuit is determined and known. The timing constraint can be obtained using static timing analysis tool to find the longest path. In the authentication mode, a very slow test clock can be used to ensure no timing violation occurs in BISA circuit. For example, suppose a tree-structure BISA circuit has a logic depth of 10 and up to 55 gates. If the propagation delay of each stage is 500ps, the total delay is 5ns (200MHz). A test clock slower than 20MHz could be used for the authentication step. With such slow speed, the authentication would still finish in few milliseconds.

**Table 5.1:** Operation modes in a design with BISA.

| | Functional mode | Authentication mode | |
|---|---|---|---|
| | | Shift phase | Test phase |
| Original circuit | Working | Idle | Idle |
| BISA circuit | Idle | Shift seed/signature | Test BISA |

## 5.2 BISA Design Flow

The BISA technique is developed to address insertion of Trojans by untrusted GDSII developers and foundries. We assume that the design has passed all necessary verifications at design phase, so it is genuine before being shipped to GDSII developer. Figure 5.2 shows the proposed BISA design flow and where it fits within the conventional ASIC design flow. The white rectangles in the figure are steps taken in a conventional ASIC design flow, and the grey ones are the additional steps for inserting BISA circuitry. We will describe them in detail in the following subsections.

### 5.2.1 Preprocessing

First, some information of standard cells in the technology library needs to be known and stored for later steps. Dimension of each standard cell, such as length and width, is needed for identifying and analyzing unused spaces. Once the placement direction and location of a particular cell are given, the coordinates of its four corners can be calculated by the cell's length and width. Additionally, the number of input pins and the name of a cell are saved for later usage. After obtaining the necessary information for all standard cells, BISA cells will be selected from them and marked according to the following criteria:

1. BISA cells must be the minimum-size cell for every logic, so they are resistant to a resizing attack by the adversary (see Section 5.3.3).
2. The amount of decoupling capacitance the cells can provide and the input count should be considered as well. The normalized input count represents the number

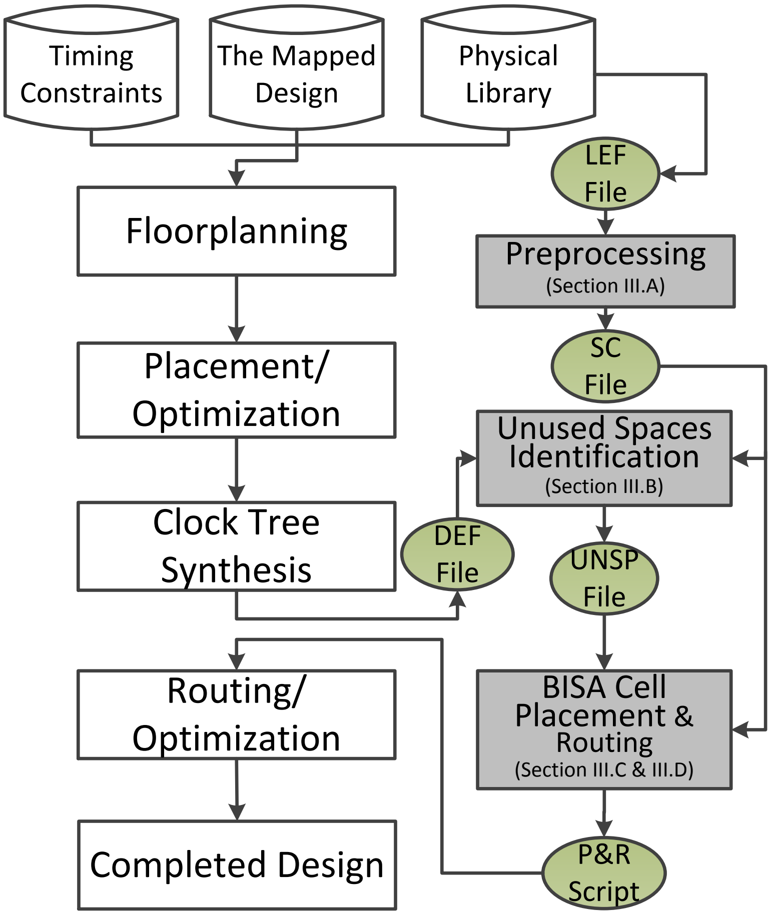


**Fig. 5.2:** BISA design flow.

of inputs of a standard cell if the same cell has the same area of the minimum-size cell (e.g., INVx0 in Table 5.2). Fewer inputs help to improve test coverage, which will be explained in Section 5.2.4.

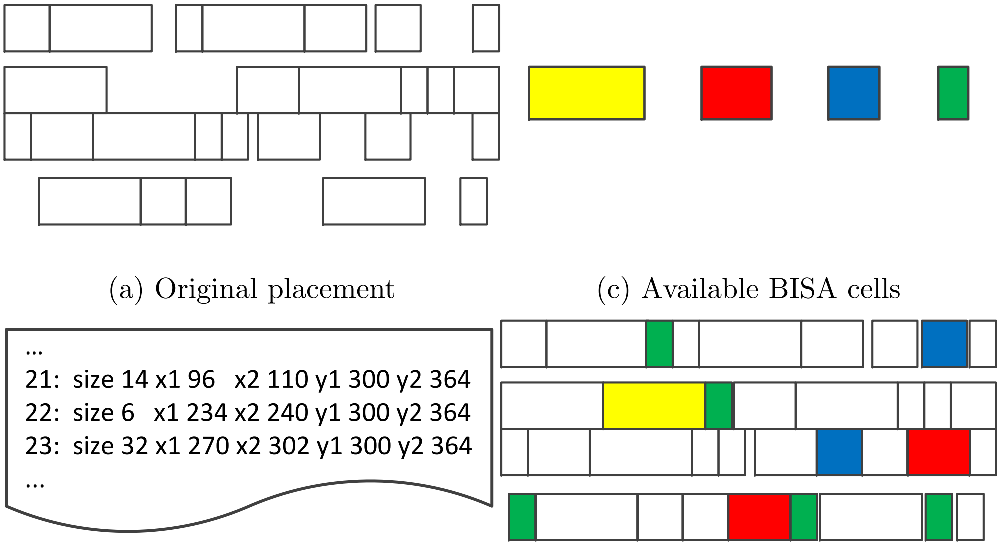
3. The smallest cell in the library must also be included in order to ensure that no cell can be inserted in any remaining unused space.

Table 5.2 gives an example generated with Synopsys 90nm library. The columns labeled *area* and *input* show each cell's area and input count. The normalized input count is listed in the fourth column. Once a cell is selected, it will be marked in the last column of the table. A cell with fewer inputs and a larger decoupling capacitance has higher priority to be a BISA cell. Note that cell INVx0, NAND2x0, and NOR2x0 are three smallest cells in this library. Since two inverters could potentially connect in serial to form a buffer and the buffer is redundant logic, INVx0 would not be chosen as a BISA cell. On the other hand, the NAND2x0 cell and the NOR2x0 cell would be selected to ensure BISA circuit contains the smallest cell.

**Table 5.2:** Cell information collected at preprocessing step.

| Name | Area | Input cnt | Normalized Input | BISA |
|---|---|---|---|---|
| INVx0 | 1920 | 1 | 1 | No |
| NAND2x0 | 1920 | 2 | 2 | Yes |
| NOR2x0 | 1920 | 2 | 2 | Yes |
| INVx1 | 2240 | 1 | 0.86 | No |
| AND2x1 | 2560 | 2 | 1.5 | Yes |
| NAND3x0 | 2560 | 3 | 2.25 | Yes |
| OR2x1 | 2560 | 2 | 1.5 | Yes |
| AND3x1 | 2880 | 3 | 2 | Yes |
| NOR3x0 | 2880 | 3 | 2 | Yes |
| NAND4x0 | 2880 | 4 | 2.67 | Yes |
| NAND2x2 | 3200 | 2 | 1.2 | No |
| ... | ... | ... | ... | ... |

A program has been developed to acquire all the above-mentioned information from the LEF file in technology library and output an in-house Standard Cell (SC) file format. This SC file must be ready before starting BISA design flow. Figure 5.2 also shows different input files needed and output files generated when going through the BISA insertion process. Other files (DEF, UNSP, P&R Script) are described in the following subsections.



(a) Original placement

(c) Available BISA cells

```
...
21:  size 14 x1 96   x2 110 y1 300 y2 364
22:  size 6   x1 234 x2 240 y1 300 y2 364
23:  size 32 x1 270 x2 302 y1 300 y2 364
...
```

(b) Unused spaces file (.unsp)

(d) Placement after BISA insertion

**Fig. 5.3:** BISA cell insertion and placement

### 5.2.2 Unused Space Identification

In the BISA design flow shown in Figure 5.2, IC designers perform floorplanning, placement, and clock tree synthesis just as they do in conventional layout design. After finishing clock tree synthesis, the original circuit has been placed in the layout, so the flow starts to search for and locate all unused spaces of the layout. We utilize a matrix, which is similar to bitmap in computer graphics, to record the state of each point in the layout. The initial state ‘0’ represents “empty”. If one cell occupies this point in the layout, the state is switched to ‘1’. Therefore, every standard cell placed in the layout will be processed one by one and eventually the bitmap reveals the location and size of unused spaces. To obtain the location of every placed cell, layout design tools such as Synopsys IC Compiler (used in this work) can write a DEF file (.def) that contains coordinates of all placed standard cells. By analyzing coordinate (.def) and its corresponding size (.sc) of every placed cell, unused spaces in layout can be identified. A Perl script has been developed to read the DEF and SC file, locate used spaces and output an Unused Spaces file (.unsp). The format of the UNSP file is shown in Figure 5.3(b). The required information, such as the size and coordinates of the four corners, are listed in the UNSP file.

### 5.2.3 BISA Cell Placement

After unused space identification, the flow completely fills the unused space with BISA cells selected at the preprocessing step. A dynamic programming algorithm is employed to find an optimal solution (see Section 5.2.5 for details). The algorithm is implemented in a program which simulates the cell insertion process and produces a Tcl command script for EDA tools. Available BISA cells and the layout after applying BISA is shown in Figure 5.3(c) and (d). Note that once BISA has been applied, there may still be some spare spaces left between cells, but not even the smallest cell (green) can be inserted. Therefore, attackers cannot insert any extra cells either.

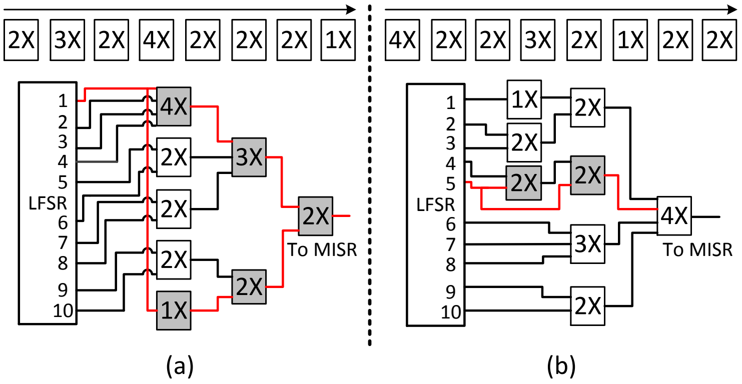


**Fig. 5.4:** Routing a BISA block.

### 5.2.4 BISA Cell Routing

Unlike regular filler cells which are placed without functionality, all placed BISA cells need to be connected in order to construct a number of combinational BISA circuits (referred to as BISA blocks). The test coverage of the BISA circuit is a key point we focus on when we are connecting BISA cells. A higher test coverage enhances the confidence level of results from BISA, since more BISA cells have been tested. Several approaches are employed to enhance stuck-at fault test coverage:

- First, we try to create as many BISA blocks as possible to make each BISA block with fewer gates so that higher test coverage is easier to achieve. Since the output of every BISA block will connect to MISR, the number of BISA blocks is determined by the size of MISR. If M is the size of MISR, all placed BISA cells are divided into M groups (BISA blocks).

- Second, redundant gates could deteriorate controllability and observability of the circuit and lower the test coverage significantly, so a tree-structure circuit is proposed to eliminate redundant gates, as shown in Figure 5.4. If every input is independent of other inputs in a tree-structure circuit, every net is controllable and observable, so the theoretical test coverage of stuck-at fault is 100%. Here, we construct a tree-structure BISA block according to the sequence of cells in a block set.

Figure 5.4 shows that two different sequences lead to two different tree-structure circuits. The first gate becomes the root of the tree-structure circuitry, i.e., it is on the top (first) level of tree. The outputs of the next $x$ cells ($x$ being the number of inputs of the root cell) are connected to its inputs as its children cells, on the second level. On the third level, we do the same to connect new cells to cells on the second level. Cells are sequentially connected to cells on upper levels until all of them are processed, as shown in Figure 5.4 (a) and (b). After complete routing in each block, all inputs of each block should connect to LFSR sequentially to avoid sharing inputs. In the end, the M bits output from M BISA blocks connect to a MISR with size of M.

### Testability of Tree-Structure BISA Block

Once a number of BISA cells are assigned to one BISA block, the input count of the BISA block is fixed and can be calculated, regardless of how the BISA cells are connected. Suppose there are $n$ BISA cells in one BISA block, and the *k-th* BISA cell has $I_k$ inputs. The BISA block's total number of inputs $S$ can thus be calculated:

$$S = \sum_{k=1}^{n} I_k - n + 1 \tag{5.1}$$

Consider the circuit in Figure 5.4 as an example. The total number of inputs is $S =$ 18(sum of cells' inputs)-8(cell number)+1 = 11. Although Figure 5.4(a) and (b) show two different tree-structures, their input counts are same. As more cells are added to one block, the number of inputs will increase consistently. If the number of inputs of one block is greater than what LFSR can provide, some inputs have to share LFSR outputs in a broadcasting fashion (i.e., using fanout). The correlation among these shared inputs could potentially result in redundant gates in BISA blocks, thereby affecting test coverage. Let us take the BISA block in Figure 5.4 (b) as an example. This BISA block has 11 input pins, but the size of LFSR is 10, which is 1 bit less than the BISA block. Thus, one input pin has to share with another input pin in this BISA block. In Figure 5.4 (a) and (b), all nets with correlation due to the shared input are highlighted in red. The input correlation can impact controllability or observability of the dark gates in the figure. One straightforward way of eliminating redundant gates is to change the tree structure to reconstruct the BISA block by adjusting the cell order, since the stuck-at fault test coverage is highly related to the circuit structure, If one slightly changes the sequence, the structure of BISA block will completely change and thus the test coverage will vary significantly. After trying different sequences in one BISA block, the sequence that has the highest test coverage is kept.

### BISA Design Strategies

Two BISA design methods, static and dynamic, are developed in the chapter. In the case of static BISA insertion, LFSR and MISR are included in the original design. After inserting BISA cells, the BISA circuits are connected to the pre-designed LFSR and MISR. However, static BISA design offers a couple of potential challenges. First, if the original design does not include LFSR and MISR (i.e. does not use built-in self-test (BIST)), BISA cannot be established. Additionally, if the BISA circuit is larger than the testing capabilities of pre-designed LFSR and MISR, sharing fanout in a BISA block will lower the test coverage and further decrease the reliability of BISA. In order to address this problem effectively, a dynamic BISA design algorithm is proposed to insert both BISA cells and LFSR/MISR into unused space and adjust their composition to ensure LFSR and MISR can fully test the BISA circuitry. The detailed algorithm will be described in Section 5.2.5. In general, the dynamic BISA design method is generally very effective for designs with many large unused spaces, like SOC design. Its shortcoming is that it may not be as effective for very compact designs (those with little unused space available), which will be discussed in Section 5.4.5.

**Routability**

Additional connections will be introduced by BISA circuits and will take some routing resources from the routing of the original circuit. In order to ensure routability of the original circuit and reduce the routing resources taken by the additional BISA circuit, we usually put the nearest cells in one BISA block to shorten their interconnections. Additionally, BISA can save more routing resources by sacrificing its own speed. The minimum size for metal wire and high metal layers could be used for BISA routing so that more room can be created for routings of the original design. Correspondingly, a slow test clock is used for BISA in authentication mode. This allows us to ignore BISA's timing constraints and focus on obtaining maximum coverage with limited routing resources.

At this step, our program can generate a routing script for Synopsys IC Compiler to create nets and connect them logically, as shown in Figure 5.2. The whole design, including BISA, will be physically routed at routing/optimization step. Once the timing and sign-off of the design are successful, the last step involves the generation of a GDSII/OASIS format of the design for final tape-out.

### 5.2.5 Place & Route Algorithms

#### Dynamic Programming Algorithm

During BISA placement, a method is needed to completely fill each unused space using BISA cells with the fewest total input count. Minimizing the total input count can not only curtail the routings required in BISA circuitry, but also lower the requirements for a LFSR and MISR. In this chapter, we take a dynamic programming approach for BISA placement. Dynamic programming is one of the classic heuristic algorithms, where a complex problem is solved by breaking it down into smaller sub-problems [100]. The optimal solution of the current problem is based on the optimal solution of its sub-problems. When applicable, this method takes far less time than heuristic search methods that do not take advantage of the sub-problem overlap (like depth-first search). For the problem of filling each unused space, all potential solutions can be derived from adding a BISA cell to a corresponding smaller space that has been solved. Comparing all the possible solutions, we will choose the filling solution with the fewest inputs. A simple example is illustrated in Figure 5.5. We are trying to fill an unused space with the size of $L$. Assume the optimal solution for any unused space smaller than $L$ ($\leq L$) has already been solved. Suppose that a total number of four BISA cells ($B_i$, $i = 1, 2, 3, 4$) are available, their sizes are $d_1$, $d_2$, $d_3$, and $d_4$, and their input count are $k_1$, $k_2$, $k_3$, and $k_4$ respectively, as shown in the brackets in the figure. The optimal solution ($S(L)$) for filling the unused space ($L$) must therefore be from the solutions of a BISA cell $B_i$ plus a sub-solution $S(U_i)$ of the corresponding unused space $U_i$ ($i = 1, 2, 3, 4$, $U_i = L - d_i$). Definitely, unused spaces $U_1$, $U_2$, $U_3$, and $U_4$ are smaller than $L$ and are $L$'s sub-problems, so the optimal solutions to fill them are already obtained. From these four potential solutions, the one with the fewest inputs will be selected. In the example, the solution of the BISA cell $B_4$ plus the $U_4$ is selected for the space of size $L$ because its input count (9, $S(U_4) + d_4$) is smaller than others' (12, 11, 12). Next, a larger unused space will be explored until the maximum area of the unused space has been investigated. Our dynamic programming algorithm can be presented visually as

$$S(n) = Min[S(n - d_m) + k_m, m \in \text{BISA cells}] \tag{5.2}$$

where $n$ is area of current space and $0 \leq n \leq L$).

The dynamic programming approach seeks to solve each sub-problem only once, thus reducing the number of computations: once the solution to a given subproblem has been computed, it is saved for later usage. The method can save significant computation time during the placement process because there are usually many unused spaces having the same size. Additionally, a dynamic programming algorithm will examine all possible solutions to the problem and will pick the best one, so it guarantees finding the optimal solution. Alternatives, such as the greedy algorithm used in [75], cannot guarantee the optimality of the solution.

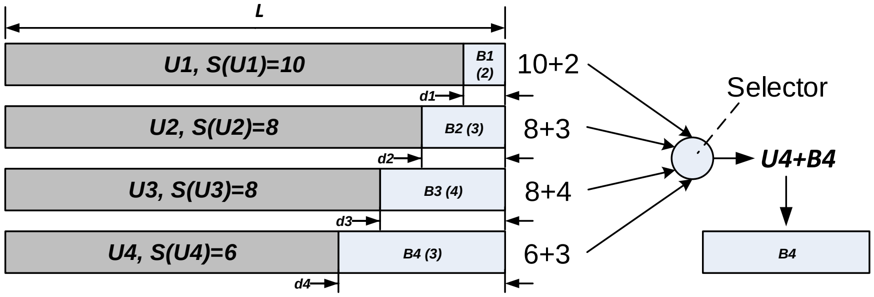


**Fig. 5.5:** An example using dynamic programming.

### Dynamic BISA Design Algorithm

The previous static BISA design algorithm [75] is not suitable for a design without pre-designed LFSR or MISR or a design with many large unused spaces, as discussed in Section 5.2.4. Thus, a dynamic BISA design method is necessary, in order to ensure high test coverage. Therefore, all BISA cells, scan flip-flops and XORs for LFSR/MISR will be inserted into unused spaces during the dynamic BISA design. Since unused spaces in layout are fixed, the key point is balancing the number of BISA cells for BISA circuit and the number of scan flip-flops inserted for LFSR and MISR in certain unused spaces. If most of the unused spaces are filled with BISA cells and the size of LFSR/MISR is small, such an LFSR/MISR cannot provide or receive long-enough vectors and thus test coverage will not be very high. If LFSR and MISR are much larger than the corresponding inserted BISA cells, the excessive scan flip-flops could result in a huge power consumption during BISA testing because of switchings in a large number of scan flip-flops at every clock cycle. Thus, the size of LFSR/MISR and BISA circuit should be balanced, in order to achieve 100% test coverage with smallest size of LFSR/MISR.

The dynamic BISA design algorithm, depicted in Algorithm 1, has been developed to address this problem. From line 1 to line 3, required information produced at previous steps–like BISA cell size, unused space area, and polynomial primitives of LFSR/MISR [99], etc.–is imported. The dynamic programming algorithm described in Section 5.2.5 finds solutions for all unused spaces smaller and equal to the maximum unused space (line 4). On the line 5, unused spaces are filled with scan flip-flops only as much as possible. Then, the corresponding number of XORs (in the polynomial primitive file) needs to insert. Remaining spaces are then filled with BISA cells, as described in line 7. Next, the number of BISA cells and their inputs is obtained (line 8). Once the BISA cell count, input count, and the size of LFSR and MISR are known, one can determine if current LFSR and MISR are able to fully test BISA circuits. We define a *test index* to represent the testability of current BISA circuits, which can be calculated using the Equation (3).

$$\text{TestIndex} = (\text{SumOfCellInputs}) - (\text{SumOfCells}) - (\text{MISRSize}) * ((\text{LFSRSize}) - 1) \tag{5.3}$$

A positive integer of the test index means that current LFSR and MISR can fully test the BISA circuits, so the process can move ahead to shorten LFSR/MISR and place more BISA cells. The following loop tries to clear the unused space that has the most scan flip-flops and refill it with BISA cells (line 11-15). If the new test index is still greater than a threshold, the action will be approved (line 16). All the BISA placement information is updated, and then the process moves forward to try the next unused space with the most flip-flops (line 17-20). If the test index is smaller than the threshold, this

**Algorithm 1** The dynamic BISA design procedure.

```
Read SC and UNSP files
Read polynomial primitive [99] for LFSR and MISR
Read dynamic programming filling results
Insert scan flip-flops to fill all unused spaces
Insert appropriate XOR gates, remove scan flip-flops if necessary
Insert BISA cells to fill remaining unused spaces
Count current BISA cell input # and BISA cell #
Calculate Test Index
while (Test Index > threshold) do
    Search an uninvestigated unused space with most flip-flops
    Clear the unused space (remove scan flip-flops, XORs and BISA cells in the unused space)
    Insert BISA cells to fill remaining unused spaces
    Count current BISA cell input # and BISA cell #
    Update Test Index
    if (Test Index > threshold) then
        Update inserted BISA cells
        Update scan flip-flops and XORs
        Update LFSR and MISR
        Update Test Index
    end if
end while
```

action is unsuccessful. The reason is most likely that the size of LFSR/MISR decreases too much, because excessive scan flip-flops for LFSR and MISR are removed. Therefore, a new loop goes back to line 11 and begins to explore the unused space with the second greatest number of flip-flops. The process will be repeated until all unused spaces with flip-flops have been investigated. Finally, the test index will be a value greater than but very closed to the threshold. It means LFSR can offer more inputs than input count of every BISA block, so there is no redundant gate in the BISA circuitry.

### 5.2.6 BISA Design in System-on-Chips (SOCs)

In this section, we extend the BISA framework to System-on-Chips (SOCs). SOC is typically a bottom-up hierarchical design, and the top module includes other pre-designed sub-modules, or what are known as intellectual property (IP) cores. We assume that each IP core in the SOC has been implemented with BISA already by the flow described in Section 5.2. In this case, each IP core can be simply treated like a standard cell, so BISA design in the top module of a SOC is nearly the same as that in a single-module design. The main difference is that IP cores are much larger than standard cells and cannot be placed in a row. In addition, we also have to determine how to organize the LFSR/MISR. For an SOC design, therefore, we just need to make some small changes on the step of unused spaces identification, and other steps are exactly same.

Since BISA will be applied to all modules in a hierarchical design, three topological structures (distributed, centralized, and hybrid) to organize LFSR/MISR are illustrated in Figure 5.6. “Distributed structure” means each IP core and top module has its own LFSR/MISR, as shown in Figure 5.6(a). Two additional pins are reserved in each sub-module for shifting data in shift mode. Although distributed structure has simple wire interconnections and requires fewer pins, having these independent LFSRs/MISRs in every module could result in the larger area overhead. In "centralized structure", only one centralized LFSR and one centralized MISR (in the top module or one of sub-modules) are used to generate test patterns and compress responses for BISA blocks in both the top module and other IP cores. Centralized structure can potentially increase the complexity of BISA cells routing, requires more pins than others and could potentially increase area. Each module will use N+M pins to receive N-bits test vectors and output M-bits responses. "Hybrid structure" is, as the name suggests, an amalgam of the two above structures. If some sub-modules have dedicated LFSR/MISR and some do not have due to limited IP core resources, "Hybrid structure" is more flexible.

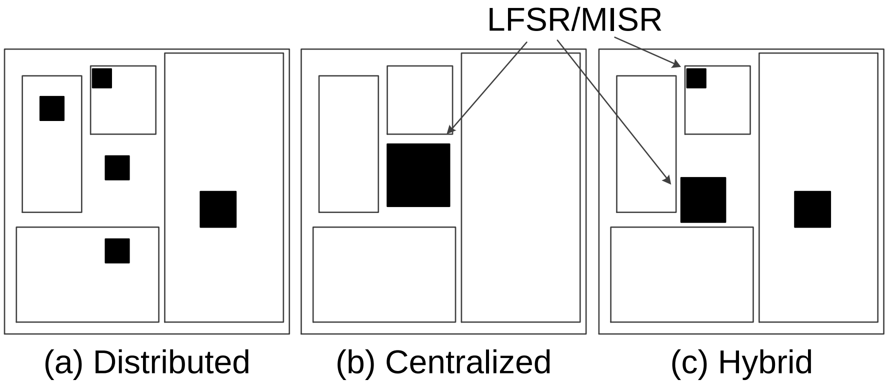


**Fig. 5.6:** Three possible structures used to organize LFSRs/MISRs in a SOC design.

## 5.3 Feasibility and Reliability of BISA

### 5.3.1 ECO

In chip design, Engineering Change Order (ECO) [101] [102] is the process of making some small changes in the processed design. Before the chip masks are made, ECOs are usually done to save time by avoiding the need for full ASIC logic synthesis, technology mapping, layout design, layout extraction, and timing verification. After masks have been developed, ECOs could be performed to save money. If a change can be implemented by modifying only a few of the layers of a design, then the cost is much less than it would be if the design was rebuilt from scratch. Our proposed BISA technique does not impact either pre-mask or post-mask ECOs. All inserted BISA cells can be used as spare logic gates. When a BISA cell is selected for ECO, the BISA cell needs to disconnect from its BISA circuits. Then another two wire connection changes are required to skip the selected BISA cell and reroute the original BISA circuits. Since BISA circuits do not have timing constraints, these two routing changes can be implemented using limited space. Moreover, our BISA design algorithm will fill unused spaces with different types of gates and ensure the variety of BISA cells.

### 5.3.2 Filler Cell and Decoupling Capacitor Cell

In conventional layout design, after placement and routing, unused spaces will be filled with filler cells or decoupling capacitor (decap) cells in order to reduce the design rule check (DRC) violations created by the base layers and to ensure power rail connection [103]. Moreover, filler cells and decap cells are able to provide power and ground capacitances, which can effectively minimize power supply noise. However, filler cells and decap cells do not have functionality, so they cannot be tested and can be easily identified and removed by attacker. Although no space is left for decap cells after applying BISA, the BISA cells are able to provide decoupling capacitance to some degree [104]. For decap cell, the decoupling capacitance comes entirely from dedicated gate oxide capacitance sources, but BISA cell utilizes its diffusion capacitance for the decoupling capacitance. Typically, diffusion capacitance is less than gate capacitance. However, N. Weste and D. Harris [105] pointed out that the diffusion capacitance $C_{sb}$ and $C_{db}$ of contacted source and drain region was comparable to the gate capacitance. In the BISA design, BISA cells are connected together to form a combinational circuit, so most source/drain diffusions are contacted for making connections. Therefore, BISA cells are able to supply decoupling capacitance to functional cells nearby. To illustrate their power and ground capacitance, we use the cell characterization function in HSPICE to report the node capacitance of standard cells. Table 5.3 lists the power ($C_{VDD}$) and ground capacitance ($C_{VSS}$) of functional standard BISA cells and filler cells in two academic standard cell libraries. Top portion for each library corresponds to the BISA cells while the bottom portions correspond to the filler and decap cells. The last column in the table shows the power and ground capacitance per unit area. We observe that standard cells do not lose much power/ground capacitance, and even some BISA cells have larger capacitance. For example, in the NANGATE 45nm standard cell library, NAND2_x1 and NOR2_x1 can offer larger average capacitance than its filler cells. Note that the source/drain capacitor is operation mode dependent [105]. When the system is in functional mode, the BISA circuit will be idle and all BISA cells are in a pre-specified mode depending on the value in the LFSR. Some cells are in the mode where they can provide the maximum decaps, while others are may not be. Since different types of BISA cells are inserted at the BISA placement step, a large number of inserted BISA cells and their diversity can ensure that average decoupling capacitance is relatively stable. Analyzing this shall be part of our future research plan.

### 5.3.3 Potential Attacks

We pointed out earlier that BISA cells are the same as other circuit cells, so it is extremely difficult for an adversary to identify them. If this were possible, however, attacks could proceed. BISA, as an authentication circuitry, should be immune to different attacks that attempt to create space for Trojan gates via removal, redesign, resizing, or bypassing. Three potential attacks on these two targets, filler cells or original standard cells, are discussed as follows.

#### Removal Attack

Removal attack is the most direct and simplest way to create space for Trojan gates. Simply removing BISA cells will change the functionality of BISA blocks. Test patterns will test the functionality of BISA blocks, and BISA signatures can tell if the BISA cells have been tampered with. If some functional standard cells from the original design are removed by adversaries, the original functionality will be altered. Both functional and structural tests are able to detect removal attacks. We recommend using as few unnecessary non-functional standard cells, such as buffers, as possible. Even if some

**Table 5.3:** The power and ground capacitance of filler cells and a portion of functional standard cells in two academic standard cell libraries.

| NANGATE 45nm standard cell library | | | | |
|---|---|---|---|---|
| Cell | $C_{VDD}$ | $C_{VSS}$ | Area | $(C_{VDD},C_{VSS})$/Area |
| INV_x1 | 38.5a | 348.5a | 3800 | (0.010,0.092) |
| NAND2_x1 | 83.6a | 536.4a | 5700 | (0.015,0.094) |
| NOR2_x1 | 86.6a | 544.9a | 5700 | (0.015,0.096) |
| XOR2_x1 | 156.6a | 1.965f | 5700 | (0.027,0.345) |
| AND3_x1 | 76.5a | 1.179f | 9500 | (0.008,0.124) |
| FILLCELL_x1 | 25.3a | 96.2a | 1900 | (0.013,0.051) |
| FILLCELL_x2 | 38.8a | 109.8a | 3800 | (0.010,0.029) |
| FILLCELL_x4 | 64.8a | 135.8a | 7600 | (0.008,0.018) |
| FILLCELL_x8 | 123a | 254.6a | 15200 | (0.008,0.017) |
| FILLCELL_x16 | 234.2a | 422.9a | 30400 | (0.008,0.014) |
| FILLCELL_x32 | 451.1a | 687.8a | 60800 | (0.007,0.011) |
| SAED 90nm standard cell library | | | | |
| Cell | $C_{VDD}$ | $C_{VSS}$ | Area | $(C_{VDD},C_{VSS})$/Area |
| INVx0 | 290.9a | 230.9a | 1920 | (0.152,0.120) |
| NAND2x0 | 516.6a | 264.4a | 1920 | (0.269,0.138) |
| XOR2x1 | 691.8a | 727.9a | 4800 | (0.144,0.152) |
| XOR3x1 | 1.045f | 1.188f | 7680 | (0.136,0.155) |
| DECAP | 317.5a | 206.8a | 1920 | (0.165,0.108) |
| DHFILLLHL2 | 175.6a | 203.6a | 640 | (0.274,0.318) |
| DHFILLHLH2 | 290.9a | 373.7a | 640 | (0.455,0.584) |
| SHFILL1 | 56.44a | 50.17a | 320 | (0.176,0.157) |
| SHFILL2 | 201.0a | 184.2a | 640 | (0.314,0.288) |
| SHFILL64 | 1.144f | 1.159f | 20480 | (0.056,0.057) |
| SHFILL128 | 2.193f | 2.215f | 40960 | (0.054,0.054) |

buffers are deleted and cannot be detected, the limited space left from removing buffers would still hamper Trojan insertion.

### Redesign Attack

If the adversary changes one or more gates that alter the functionality, it will be detected by functional tests regardless in BISA circuit or the original circuit [35,36]. If the adversary is forced to redesign the layout in order to achieve the same functionality and insert Trojan gates as well, then the chip dimensions will probably alter. In addition, these alterations could result in changes on placements and routings for some or all design components and their interconnections. Modifications to original circuitry would impact the circuit timing that could be detected using side-channel based techniques [39,40]. Therefore, the redesign attack could be more effective on BISA than original circuit, because BISA is redundant circuit in functional mode and does not impact circuit performance. However, there are major difficulties making it practically impossible. First, it is nontrivial for the foundry to differentiate the original circuit from BISA in order to create the space for hardware Trojan insertion. BISA cells are the same standard cells as in the original design. BISA cells form combinational circuits and are connected to scan flip-flops of LFSR/MISR, which look like a standard BIST design. The foundry would need to analyze the entire layout to identify the BISA cells. While, this is not necessarily impossible, it does require a significant amount of effort and expertise on the part of the attacker to insert a Trojan compared to conventional approaches that do not use BISA. Second, should an adversary actually identify BISA cells, they must then analyze the BISA circuitry and find a functionally equivalent logic with a smaller area. Finding a feasible solution for a tree-structured circuit is another challenge. For example, two 2-input NAND gate converging in another 2-input NAND gate can be replaced by a 2-input OR gate and two 2-input AND gate. The area of the NAND gate implementation is 16.59 ($\mu m^2$) using the SAED90nm standard cell library, but the OR gate plus AND gates need a space of 22.12 ($\mu m^2$). Thus, it cannot be used to create unused spaces for inserting hardware Trojans. The most possible way is that some certain gates can be function-equivalently replaced by an available complex standard cell in standard cell library, such as AO22x1, AOI221x1 and etc. Since these complex cells are well designed and compacted as much as possible, such as sharing diffusion areas and sharing connection wires. Usually, the complex cell has a smaller area than its functionally equivalent circuit comprised of basic cells (like NAND, NOR, AND, and OR). For instance, the area of an AO22x1 cell (11.98 $\mu m^2$) is smaller than the area of a 2-input AND cell plus a 2-input OR cell (14.74 $\mu m^2$). Therefore, these logics that complex cells have should be avoided in a tree-structured BISA block. If found, a new logic can be formed by slightly changing cells order in the BISA block. Moreover, this attack can also be made more challenging if the BISA cells that are connected are not directly neighboring. Even if there is an equivalent circuit, the area might be larger than any unused space created by removing any one of these distributed BISA cells. Taking the layout in Figure 3 for example, we suppose that a yellow cell can replace two blue cells without change in functionality and its size is smaller than total size of two blue cells. Since a yellow cell is larger than one blue cell, the new yellow cell cannot be inserted into any places where two original blue cells locate without touching original circuit. In addition, some techniques can be used to prevent the logic minimization at the BISA design stage. Synthesis tools have the ability to minimize logic and can be used to verify if a BISA block is the implementation with the minimum area. If not, one can reorder the sequence of gates to form a new tree structure BISA block until it cannot be optimized. This step does not change the placement of BISA cells, but it needs a longer processing time and modifies routing in the BISA block. Third, if the attacker cannot simply replace the BISA cells with a functionally equivalent cell with smaller area, the only alternative is to begin modifying the original design as well. As mentioned earlier in the chapter, BISA does not address the issue of redesigning the original circuit. However, modifications in the original circuit could impact the performance of the original design and change its side-channel (delay, power, etc.) behavior. While an attacker could overcome the obstacles above with enough effort, Trojan insertion is still significantly hampered with BISA compared to other approaches.

### Resizing Attack

In BISA, all BISA cells are already of minimum size and cannot shrink any further. If adversaries continue to reduce the cell size, it will violate design rules and result in a very low yield. Some standard cells in the original design could be resized to smaller standard cells; however, the performance will be affected by using smaller cells.

### Attack LFSR/MISR

LFSR and MISR are used to generate patterns and signatures respectively, so they are also possible targets for the adversaries. Fortunately, both TPG and ORA are quite secure against different kinds of attacks. Any modification to the register or XOR gate in TPG or ORA will result in a totally different patterns and signatures. No matter what change is made, a different signature will indicate that the chip has been tampered with. Admittedly, attackers may acquire generated test patterns and the corresponding signature by attacking TPG or ORA. They might store these results, bypass BISA, and output them pretending BISA is working normally. However, even if they could obtain all this information, the bypassing attack will be detected. Different seeds in LFSR will result in different signatures. The seed and its corresponding signature are similar to a challenge-response pair. Since the seed used for authentication is not fixed and an arbitrary seed can be shifted in through scan chain, it is nearly impossible for attackers to speculate the final signatures without knowing the seeds that test engineers will use. The trustworthiness of BISA can be guaranteed by applying various seeds.

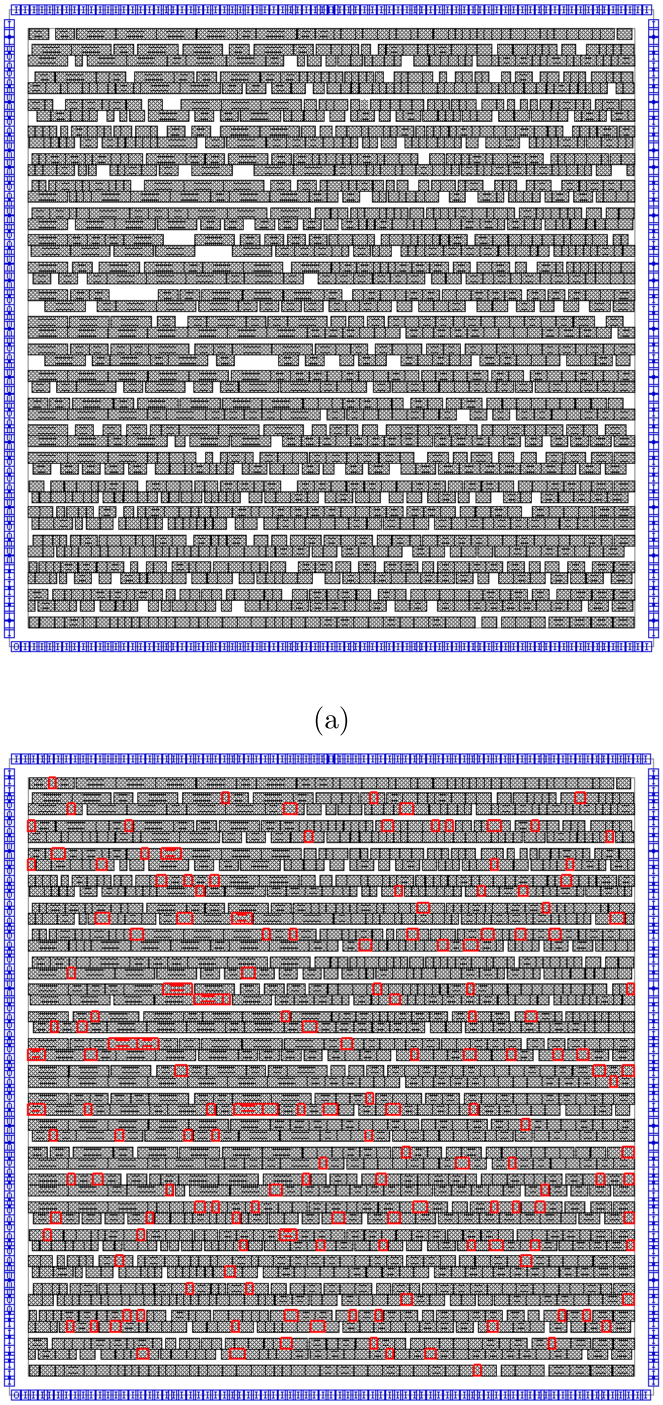
### 5.3.4 Yield

The discussions above rely on the fact that BISA is genuine and working without manufacturing defects. However, BISA contains LFSR, MISR and many BISA cells which may also be affected by silicon defects during fabrication. One chip producing a faulty signature may contain hardware Trojans, but it also could have been caused by defects in BISA, while the original circuits are working well. Of course, we do not want to discard good chips with defects only in their BISAs, reducing yield and increasing costs. Fortunately, it should be possible to reliably tell the difference. Hardware Trojans are intentionally inserted into all or a portion of chips by adversaries, while defects occur randomly due to imperfect manufacturing processes. Therefore, the probability of two chips with the same defects would be very low, but chips with the same Trojan would always produce the same faulty signatures–this provides us with opportunities to separate chips with random defect from Trojans. Note that masks used for fabrication cost millions of dollars, so it is infeasible for adversaries to make one mask for each chip in order to imitate random defects. Chips with the same correct signatures and the same faulty signatures will therefore be suspected as infected chips. If the faulty signatures from one chip are completely different from other chips, the faulty signatures probably result from defects in BISA.

## 5.4 Results and Analysis

### 5.4.1 BISA Implementation

In this section, we implement BISA on 15 designs from various benchmark suites. Four of them (OpenSparc, Leon3mp, Netcard and VGA_lcd) are SOCs containing hierarchical sub-modules and the others are single-module designs. All benchmarks are synthesized using Synopsys Design Compiler with Synopsys 90nm technology library. Overall, 32 standard cells were selected as BISA cells from the 320 in the library and each cell has a unique function. Synopsys IC Compiler has been employed for layout design. For our implementations of BISA, a series of programs have been developed to extract standard library information, to identify unused spaces, and to generate placing & routing scripts for the IC Compiler. Figure 5.7 shows the layouts of benchmark DES3_area with the core utilization of 70%. Before applying BISA, we can see many unused spaces in Figure 5.7(a). In Figure 5.7(b), all these unused spaces are filled with BISA cells, highlighted



(a)

(b)

**Fig. 5.7:** Implementation of a single module design.

in red, after BISA insertion. Although there still are some spare spaces left, these remaining spaces are too small to place a cell. Another benefit of these left spare spaces is that they can be routing channel for low-layer metals, especially for critical paths.

Figure 5.8 shows the implementation of one Sparc core in OpenSparc T1 benchmark [106]. The Opensparc T1 is the first 64-bit open-source microprocessor, released in 2006 by ORACLE. The Opensparc core is composed of seven sub-modules (hard macros): exu, ffu, ifu, lsu, mul, spu, and tlu. The layout floorplan and proportion of every sub-module are depicted in Figure 5.8(a) and (b), respectively. We try to compact the floorplan as much as possible in the layout, but gaps still inevitably exist between sub-modules. With the implementation of BISA, BISA circuit can perfectly fill all these gaps. Since the OpenSparc benchmark has more than half million gates, it is very difficult to see those inserted BISA cells. To show the BISA circuitry between sub-modules, three screenshots at different locations in the layout are shown in Figure 5.8(c)-(e). The black parts are cores (sub-modules) and the red components represent the inserted BISA cells.

Similarly, BISA has been applied to all benchmarks, and their implementation results are presented in Table 5.4. Compared to single-module designs, SOC designs need many more BISA cells. Therefore, in these SOC designs, a large dynamic LFSR and a MISR are generated in these unused spaces to ensure a very high test coverage without any area overhead. It is worth nothing that the size of the BISA circuit is not proportional to the original circuit and it is related to the circuit structure and its timing constraints. For example, the AES_core benchmark is half the size of the

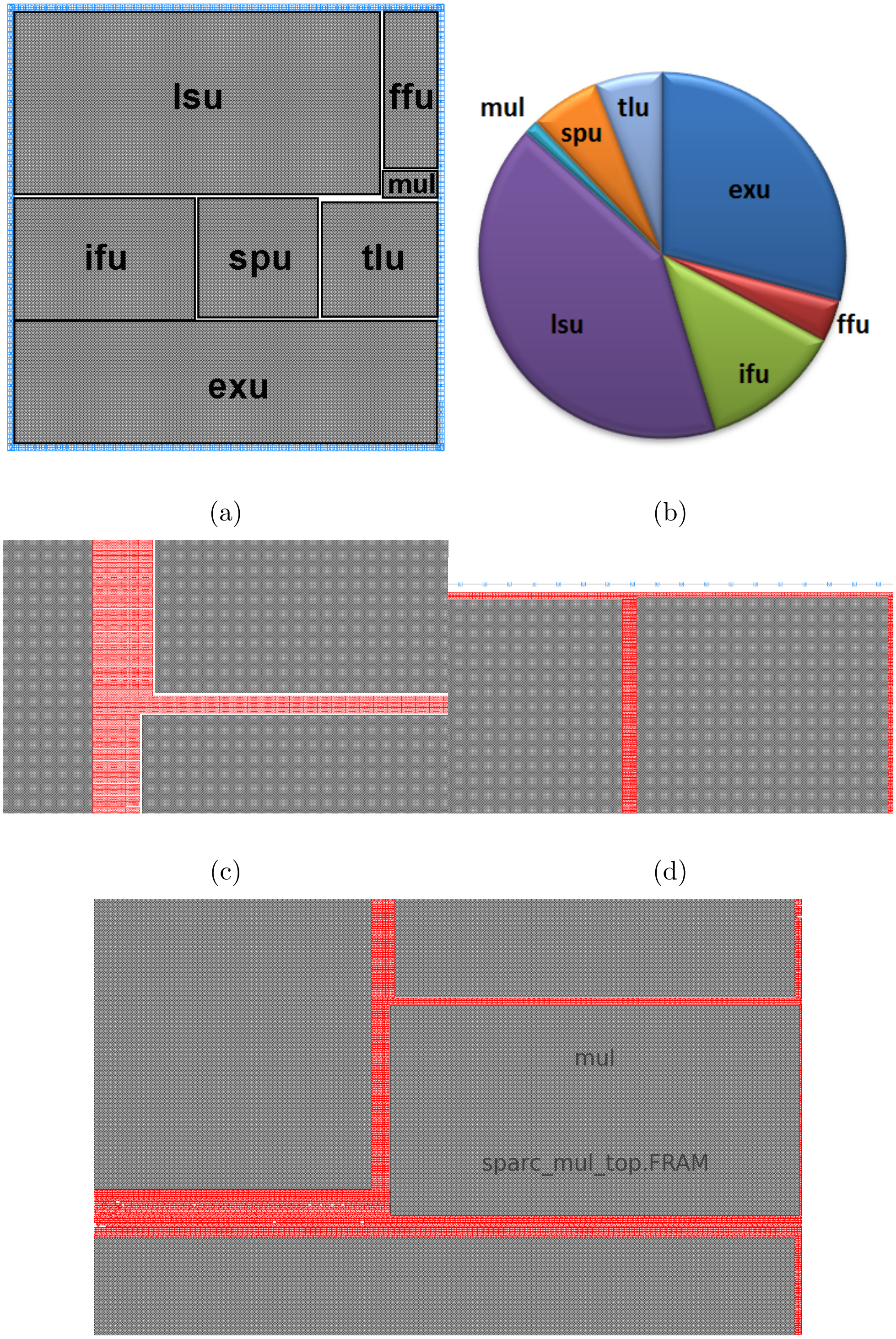
DES_perf benchmark, but its BISA circuit is more than twice as large as that in the DES_perf using the same core utilization.

**Table 5.4:** BISA implementations.

| Benchmark | OpenCore | | | | | | Faraday Technology Corp. | | | ITC'99 | SOC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DES3_area | USB_funct | AES_core | Ethernet | DES_perf | VGA_lcd | DMA | DSP | RISC | b18 | Leon3mp | Netcard | OpenSparc Core |
| Cell # | 1,559 | 6,445 | 26,447 | 29,153 | 49,517 | 124,031 | 6,873 | 17,895 | 30,229 | 24,300 | 545,836 | 317,033 | 781,321 |
| BISA cell # | 133 | 440 | 5,961 | 1,170 | 2,092 | 5,280 | 1,544 | 5,212 | 3,756 | 2,050 | 25,405 | 19,364 | 30,903 |

### 5.4.2 Filling Approach

A filling approach using the dynamic programming algorithm has been developed in Section 5.2.5. Theoretically, this algorithm can produce an optimal filling solution with the fewest number of inputs. Four randomly picked unused spaces have been investigated. The greedy algorithm (one classic heuristic algorithm) used in [75] has also been employed to fill these four spaces with the same set of BISA cells. Table 5.5 is the comparison results between the two algorithms under the same condition. Rows 5 and 6 show the total input count of inserted BISA cells and the leftover space after BISA insertion, respectively. In all four cases, the dynamic programming algorithm (DP) outperforms the greedy algorithm in the total input count and therefore should obtain lower LFSR overheads and higher test coverage more easily. In all but one case (case 3), the dynamic programming algorithm leaves the same amount of spare space



(a)

(b)

(c)

(d)

(e)

**Fig. 5.8:** Implementation of the OpenSparc microprocessor core.

as the greedy algorithm. In case 3, it leaves significantly less space. Therefore, the dynamic programming algorithm is at least as good as the greedy algorithm in filling unused spaces, if not better.

**Table 5.5:** Algorithms performance.

| | Space 1 | | Space 2 | | Space 3 | | Space 4 | |
|---|---|---|---|---|---|---|---|---|
| Area | 15000 | | 20000 | | 30000 | | 48000 | |
| Algo. | DP | Greedy | DP | Greedy | DP | Greedy | DP | Greedy |
| Cell # | 2 | 2 | 4 | 3 | 5 | 4 | 7 | 7 |
| Input # | 7 | 9 | 9 | 10 | 12 | 15 | 19 | 20 |
| Left Area | 1880 | 1880 | 160 | 160 | 240 | 1520 | 0 | 0 |

### 5.4.3 Test Coverage Analysis

As described in Section 5.2.4, two factors can influence the eventual test coverage of a BISA circuit: the testability of the BISA circuit and test patterns. The testability of the BISA circuit is determined by the number of redundant gates at the BISA design phase. As LFSR and MISR become larger, their testing capabilities enhance and the number of redundant gates goes down. Figure 5.9 shows testability of the BISA circuit in benchmark USB_funct by performing fault simulation in Synopsys TetraMax. Different sizes of LFSRs, such as 32-bit, 16-bit and 8-bit are investigated. As we expected, the test coverage goes up gradually as LFSR and MISR become larger. The testable coverage is a theoretical test coverage that can be achieved, but the actual test coverage also is dependent on test patterns, i.e. the number of random patterns generated from LFSR.

The test coverage over a different number of test patterns are illustrated in Figure

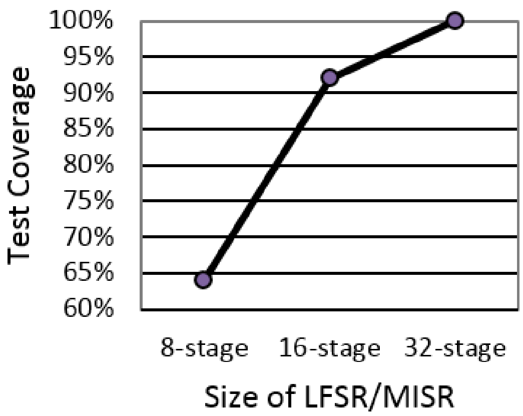
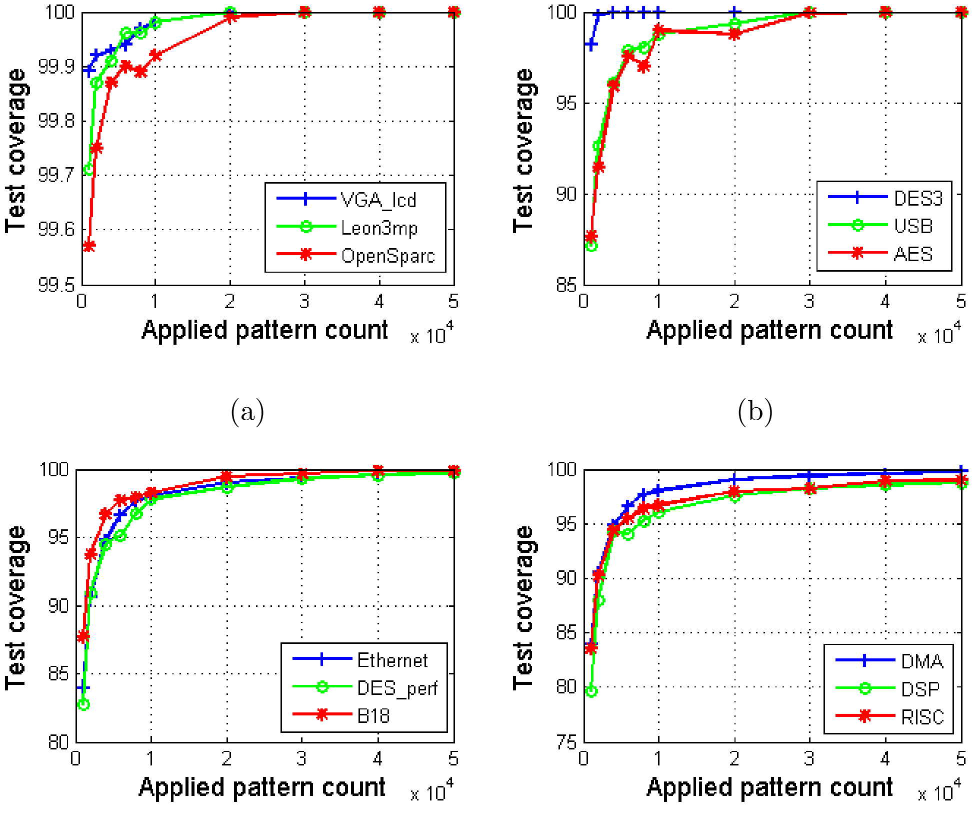


**Fig. 5.9:** Testability of BISA circuit with three types of LFSR/MISR.



(a) (b)

(c) (d)

**Fig. 5.10:** Test coverage over a different number of test patterns.

5.10. As more random test patterns have been applied, the test coverage increases and then gradually goes to saturation in all 12 benchmarks. The top left figure shows the results of three SOC designs. Although these SOC designs have more BISA cells and larger BISA circuits, compared to other single-module benchmarks, their test coverage can be 100% after applying 30,000 random patterns. For other single-module designs with smaller BISA circuits, their test coverage is lower than the SOCs', which defies our intuition. This is because, for the BISA circuits in SOCs, XOR gate is much more than other types of gates, such as AND, OR, NOR, and NAND. A XOR gate can produce half of ones and half of zeroes with neutral input values. In other words, XOR is more controllable and observable than other types of gates. Therefore, these BISA circuits with much more XORs can have a higher test coverage. Three examples (DMA, DSP and RISC) are shown in the bottom right figure. Their test coverage is around 98% after applying 50,000 test patterns–lower than other benchmarks. If we further apply a number of random patterns, their test coverage still keeps increasing, but very slowly. We find that the remaining faults can be tested by some specific patterns with a large portion of deterministic bits that have a very small probability of occurring in random patterns. However, in order to get the full test coverage in a short time, certain seeds for LFSR can be shifted in so that these specific patterns can be generated as well.

### 5.4.4 Dynamic BISA vs. Static BISA

For the static BISA design, the original design must contain LSFR and MISR. The LFSR and MISR have been designed before layout design, so it is difficult to accurately predict the size of the BISA circuit at that time. If the pre-designed LFSR and MISR are not large enough to effectively test the entire BISA circuit when it is implemented in layout, the test coverage will not be very high, especially for a low-utilization design or SOC design. The dynamic BISA design approach described in Section 5.2.5 makes up the shortcoming of the static BISA design, which is more suitable for designs with large unused spaces. All designs have been revaluated to demonstrate if the dynamic BISA design can enhance the testability of BISA circuitry. Figure 5.11 shows the results of a portion of benchmarks using the static and dynamic BISA design approaches. In comparison, the dynamic BISA design algorithm can avoid redundant gate in BISA circuit, so their test coverage is always 100%. The test coverage in fault simulation for the static BISA design is around 95%, and the AES_core is even less than 80%. We can make an obvious conclusion that the dynamic BISA insertion can enhance the testability of large BISA circuits significantly.

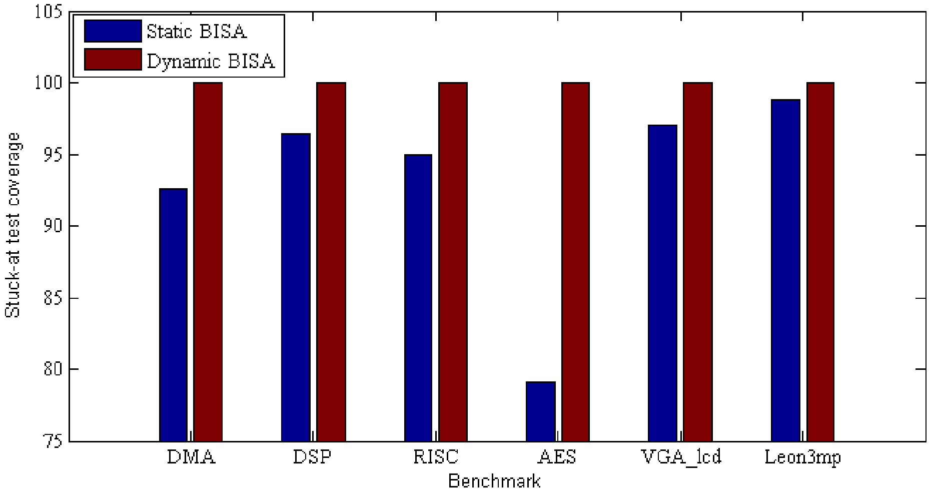


**Fig. 5.11:** Test coverage using static and dynamic BISA designs.

### 5.4.5 Compaction Ratio

Although the dynamic BISA design can realize a higher test coverage than the static design, the dynamic method requires more large unused spaces to place sufficient scan flip-flops. Therefore, the dynamic BISA design is not suitable for very compact designs. In a layout, the number and size of unused spaces are dependent on the layout compaction ratio (core utilization). If the core utilization is very high, standard cells are placed tightly in each row. This will result in fewer and smaller unused spaces. There will not be enough scan flip-flops to construct large entough LFSR and MISR, so the LFSR and MISR cannot fully test the BISA circuits. Table 5.6 shows the results using various core utilizations in the two largest OpenCore benchmarks.

**Table 5.6:** Compaction Ratio.

| Benchmark | AES | | | DES_perf | | |
|---|---|---|---|---|---|---|
| Utilization | 65% | 68% | 70% | 67% | 70% | 75% |
| Unused Space # | 22343 | 21518 | 19971 | 23653 | 19557 | 11963 |
| BISA # | 12156 | 10936 | 10417 | 7380 | 5011 | 2028 |
| LFSR & | 149 | 139 | 109 | 107 | 131 | 15 |
| MISR | 149 | 140 | 109 | 107 | 132 | 15 |
| Success | Yes | Yes | No | Yes | Yes | No |

According to the results, the dynamic BISA design works very well for SOC designs and single-module designs with a lower compaction ratio. In most cases, designers want to compact designs as much as possible. Pre-designed LFSR and MISR are a better choice for these high-compact designs. Since the number of BISA cells in a high-compact design is very small, the pre-designed LFSR and MISR should be large enough to test them. On the other hand, in a large design, the core utilization cannot be very high due to conservative floorplanning. For example, 93% is the highest core utilization we can implement for the benchmark AES. If the layout design is not as compact as designers' predictions in the netlist design, then there will be too many unused spaces, requiring that an excessive number of BISA cells be inserted. The inserted BISA cells will be beyond the test capabilities of pre-designed LFSR and MISR. To solve this problem, the dynamic BISA technique can be used to insert flip-flops, increase the size of LFSR/MISR a little bit and, meanwhile, reduce the number of BISA cells. Thus, test coverage and area overhead could be improved greatly.

### 5.4.6 Timing Overhead

Although the BISA circuits have no interconnections to the original circuits, there may still be a concern that the BISA cells will negatively impact timing of the original design. The additional metal interconnections introduced by BISA circuits will result in less routing space and more cross-talk. These effects could potentially make critical paths violate timing constraints that were verified without the BISA circuits. We have conducted experiments to explore how much extra delay will be introduced by the BISA circuits. Benchmark b18 has been implemented with and without BISA. 200 critical paths have been extracted from designs with BISA and without BISA by Synopsys Primetime, and the HSPICE has been employed to simulate their propagation delays. To obtain information on path delay changes with and without BISA insertion, the same paths are used for comparison. Since the BISA circuit size is dependent on the core utilization of a design, different core utilizations for one design are investigated as well. Table 5.7 shows results from the benchmark b18. The third column presents the average increased delay after BISA insertion across 200 critical paths. The fourth and fifth columns indicate the average percentage of the increase and largest increase respectively. As these results show, more unused spaces require more BISA cells as the core utilization goes down, thus increasing the impact on path delay. We can see that the impact is much less than the wafer-to-wafer process variation, because the largest impact is less than 2%. Therefore, the impact on timing is not a critical issue for the proposed BISA technique.

**Table 5.7:** Timing analysis on 200 critical paths in designs with and without BISA circuits.

| Benchmark: b18 | | | | |
|---|---|---|---|---|
| Utilization | BISA cell # | Avg. Inc. Delay | Avg. Per. | Max Per. |
| 78% | 299 | 23.8 ps | 0.21% | 0.36% |
| 75% | 2000 | 111ps | 0.96% | 2.17% |
| 72% | 3512 | 191ps | 1.66% | 2.38% |

### 5.4.7 Attack Detection

In this subsection, we perform ten different attacks to see if the BISA can reflect these attacks. The benchmark USB_funct design is selected, and 440 BISA cells are inserted to fill unused spaces. LFSR and MISR with a size of 32 are used to form the BISA structure. 616 ATPG (automatic test pattern generation) patterns can reach 100%

testable coverage. When 500 random patterns from LFSR are applied, the stuck-at fault test coverage is only 81%. Five kinds of gates are selected to be removed from different BISA blocks separately. In addition, another five types of gates are selected to be changed, separately, to other types of gates in different BISA blocks. Their signatures are shown in Table 5.8. In this table, we define some terms to represent cells in a tree-structure block. An "internal cell" is the cell of a tree that has children cells. Similarly, a "leaf node" is a node that does not have children cells. Five cases are developed for removal and change respectively. In every case, the signature generated from MISR is different from the genuine signature, which shows that BISA can detect these attacks effectively, even if the test coverage is not 100%. Once more test patterns are applied, a higher test coverage will enhance the BISA's detection ability for attacks.

## 5.5 Conclusion

In this chapter, we present a novel BISA structure to address IC trust issues due to untrusted GDSII developers and foundries. BISA will fill in all unused spaces in an IC to prevent or hamper the Trojan insertion process after layout design is complete, by leaving no space for Trojan gates. BISA cells are inserted in the unused space and connected to form a functional circuit. We have applied BISA on different designs from diverse benchmark suites. Different types of attacks can be detected by low-cost logic self-test in the BISA, which ensures that BISA's result is trustworthy. By comparing signatures, a designer would know whether a chip has been tampered with or not, as demonstrated by the implementations of various benchmarks.

**Table 5.8:** Potential attacks.

| Case | Type | Attack Description | Signature |
|---|---|---|---|
| 0 | Genuine | None | 0712022D |
| 1 | Removal | Remove a leaf cell | 0DA8936E |
| 2 | Removal | Remove an internal cell | 157F4929 |
| 3 | Removal | Remove a leaf cell | 0ED740FC |
| 4 | Removal | Remove an internal cell | D5E2706E |
| 5 | Removal | Remove an internal cell | 43D51D83 |
| 1 | Change | Change a leaf cell<br>OR3X1 to AND3X1 | F2308684 |
| 2 | Change | Change an internal cell<br>AOI222X1 to OAI222X1 | 157F4929 |
| 3 | Change | Change a root cell<br>AOI222X1 to OAI222X1 | F39C3B1E |
| 4 | Change | Change a leaf cell<br>AND3X1 to NAND2X1 | 157F4929 |
| 5 | Change | Change an internal cell<br>NAND4X1 to NAND3X1 | 0B17041F |

# Chapter 6

## Design-for-Security Methodology: Obfuscated Built-In Self-Authentication via Split Manufacturing

The threats of reverse-engineering, IP piracy, and hardware Trojan insertion in the semiconductor supply chain are greater today than ever before. Split manufacturing has emerged as a viable approach to protect integrated circuits (ICs) fabricated in untrusted foundries, but has high cost and/or high performance overhead. Furthermore, split manufacturing cannot fully prevent untargeted hardware Trojan insertions. In this paper, we propose to insert additional functional circuitry called obfuscated built-in self-authentication (OBISA) in the chip layout with split manufacturing process, in order to prevent reverse-engineering and further prevent hardware Trojan insertion. Self-tests are performed to authenticate the trustworthiness of the OBISA circuitry. The OBISA circuit is connected to original design in order to increase the strength of obfuscation, thereby allowing a higher layer split and lower overall cost. Additional fan-outs are created in OBISA circuitry to improve obfuscation without losing testability. Our proposed gating mechanism and net selection method can ensure negligible overhead in terms of area, timing, and dynamic power. Experimental results demonstrate the effectiveness of the proposed technique in several benchmark circuits.

## 6.1 Introduction

Split manufacturing, or split fabrication (used interchangeably in this paper), has been proposed recently as an approach to enable use of state-of-the-art semiconductor foundries while minimizing the risks to IP [107]. Split manufacturing divides a design into Front End of Line (FEOL) and Back End of Line (BEOL) portions for fabrication by different foundries. An untrusted foundry performs (higher cost) FEOL manufacturing, then ships wafers to a trusted foundry for (lower cost) BEOL fabrication.

Two types of split manufacturing have been proposed in prior work: 2D integration and 3D integration based split fabrication. [108] first proposed the use of 3D integration of two tiers (the computation plane and the control plane) manufactured in separate foundries to ensure the performance of the computation plane and the security of the control plane. One tier is stacked on the top of another tier and conventional 3D stacked integration techniques are required to merge two tiers with vertical interconnections called through-silicon-vias (TSVs). Unfortunately, the semiconductor industry has not adopted 3D ICs as quickly as many in the industry expected. Given the barriers to 3D, 2D and 2.5D based split manufacturing are discussed more, such as those in [109–112,114]. [115] developed a technique that makes FEOL and BEOL fabricated separately in different foundries and can make connections between them with wafer-bonding at a fine enough pitch similar to TSVs. [109,110] demonstrated the feasibility of split fabrication after metal 1 (M1) on test chips and evaluated the chip performance. Although the split after M1 attempts to hide all inter-cell interconnections and can obfuscate the design effectively, it leads to high manufacturing costs. [111,112] employed another integration approach. The back-end layers can be manufactured directly on top of the front-end layers with mask alignment techniques in a trusted foundry, which was studied on an FPGA chip fabricated in a split manufacturing process [112]. Finally, [113] presents a $k$-security metric to select wires to be lifted to a trusted tier (BEOL) to ensure the security when split at a higher layer. However, lifting a large number of wires in the original design will introduce large timing($\geq$73%) and power($\geq$54%) overhead and significantly impact chip performance [113], since delay and power are strong functions of wire length. In addition, though $k$ similar elements can be created by lifting sufficient wires, it cannot prevent adversaries from tampering all these similar elements with untargeted Trojans.

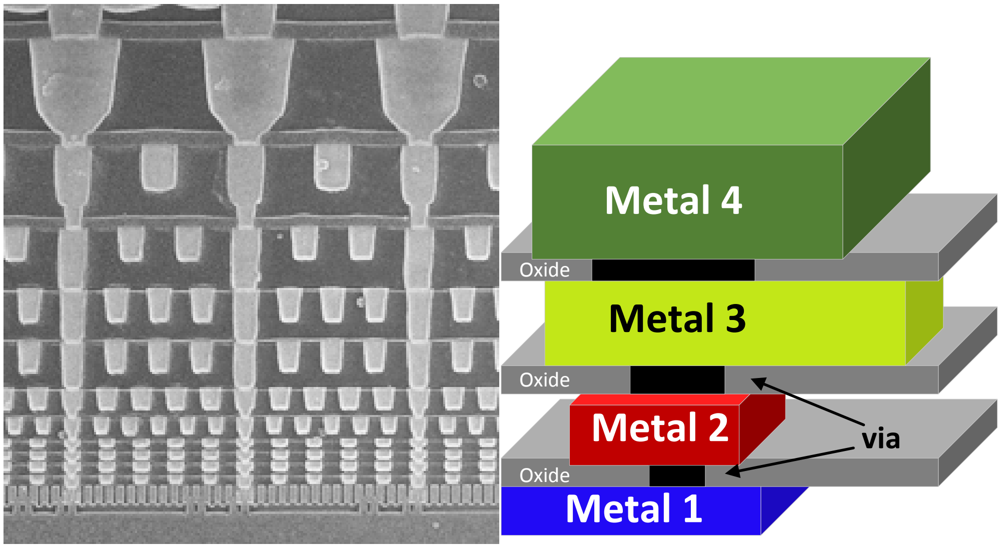
In this chapter, we propose a new design methodology that can effectively prevent reverse engineering of the chip functionality and further prevent hardware Trojan insertion with split fabrication process. Our technique allows FEOL and BEOL to be separated at higher layers ($\geq$ M4) to reduce cost. In order to enhance effectiveness of obfuscation, all unused spaces in layout will be filled with additional functional cells or circuitry called obfuscated built-in self-authentication (OBISA) instead of non-functional filler cells during layout design [116]. If any of the OBISA cells are replaced by a Trojan cell during FEOL, OBISA will be able to detect the modification. We propose to make connections between OBISA added into unused spaces and the original circuit, especially in its critical parts that are to be protected. The OBISA circuit not only makes it extremely difficult for adversary to identify the original design, but also thwarts hardware Trojan insertion by filling unused spaces in layout. In addition, several design-for-security methods are proposed to minimize timing and power overhead introduced by OBISA circuit while maintaining a high test coverage for OBISA.

## 6.2 Low-layer Split vs. High-layer Split

### 6.2.1 Cost Issues

Reverse engineering requires analysis of local standard-cell types and their interconnections in order to identify structures or some targeted logics within a circuit. Split fabrication can prevent reverse-engineering the complete design or macro in a layout design by hiding a portion of wires. However, an adversary still could identify some sub-circuits, such as adder, decoder, cryptographic logic, etc., based on FEOL mask-layer information. The strength of obfuscation depends on the split layer, i.e., the layer that ends at the FEOL, since it determines how much layout information will be exposed to untrusted foundries. Figure 6.1 (a) shows a cross-section of a $14nm$ Intel chip [117]. A high-layer split leaks more interconnections while low-layer split leaks much less. Split after M1 provides exceptional circuit obfuscation, because adversaries at untrusted foundry only see unconnected gates in layout and no inter-cell connections at all [109] [110]. However, such a low-layer split also brings challenges in split manufacturing process.

First, high metal layers are thicker and have a larger pitch than low metal layers, as shown in Figure 6.1 (a) and (b). Table 6.1 shows the pitches of different metal layers for a 45nm technology [114] [93]. The integration process of FEOL and BEOL wafers requires precise alignment using either electrical, mechanical, or optical alignment



(a) Intel's 14nm Chip [117]

(b) Metal layers and vias

**Fig. 6.1:** A cross-section of an IC.

techniques. A via requires a certain amount of space surrounding in order to satisfy design rules. If alignment is not perfect, the misalignment defects could influence circuit performance or even produce malfunction. However, the tolerance for misalignment improves greatly if the split occurs at higher layers. Unfortunately, misalignment has a higher probability to occur in split fabrication due to different process technologies or facilities at two different foundries (FEOL and BEOL), the yield for low-layer split could be relatively low.

**Table 6.1:** Pitch length of different metal layers in 45nm CMOS technology

| Layer | Poly | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 | M9 |
|---|---|---|---|---|---|---|---|---|---|---|
| Pitch(nm) | 125 | 130 | 140 | 140 | 280 | 280 | 280 | 800 | 800 | 1600 |

Second, higher layer split leads to fewer and less dense connections between the trusted and untrusted tiers, and can mitigate the challenges of alignment for a large number of connections. Therefore, high-layer splits would reduce complexity of integrating FEOL and BEOL and result in a high integration yield.

Third, lower layer splits also requires a closer technology match between foundries. One major reason for choosing split fabrication is that trusted foundry (BEOL) has less advanced technology and cannot meet the specification for a particular design. A split at a higher level can allow BEOL fabricated with older process technologies, making split manufacturing more widely acceptable and further reducing its costs.

### 6.2.2 Security Issues

Split at a higher layer has many pros as described above, but it also results in significant interconnections for circuit blocks information leakage. Although adversaries cannot reverse engineer the entire design due to lack of connections in BEOL, it still provides adversaries opportunities to identify some sub-circuits (such as adder, decoder, and FSM) and tamper them with hardware Trojans [118]. [109] and [111] proposed to insert additional "dummy" cells as spare cells for obfuscating the composition of a circuit, but the inserted cells have no interconnections between themselves or between them and the main design. Hence, they are easy to identify.

## 6.3 Split Fabrication with OBISA

### 6.3.1 Proposed Approach

Based on the previous techniques, there is a tradeoff between cost and security for split manufacturing. However, our objective is to develop a methodology that allows high-layer splits, M3 or higher, while lowering the cost and at the same time providing a high level of security. We propose to insert *functional standard cells* into unused spaces of layout instead of filler cells or dummy cells. The inserted cells are connected together to form a circuitry, called obfuscated built-in self-authentication (OBISA). Note that this is different from the previously proposed built-in self-authentication (BISA) [75] [76] for preventing Trojan insertion both from design and objective standpoints. As shown in Figure 6.2 and 6.3, OBISA is connected to the original circuit it is trying to protect, which makes it extremely difficult for adversaries at untrusted foundry to separate the OBISA design from the original design. However, OBISA circuit would result in dynamic power and timing overhead when the original circuit is operating. Therefore, a gating mechanism and a net selection method are proposed to minimize the negative impact on the original design. Additionally, we attempt to maintain a high test coverage for OBISA circuit with tree-structure circuits as in [75] [76]. Its built-in self-test (BIST) like structure can detect potential malicious modifications by test patterns that target stuck-at faults. However, the tree-structure circuit and test structure components could become a target for the adversary to identify them. In this paper, an approach is presented that can create fan-outs to further obfuscate tree-structure circuit in OBISA without impacting its original high test coverage. Moreover, lifting critical wires to the trusted tier (BEOL) can prevent identifying the test structure within OBISA.

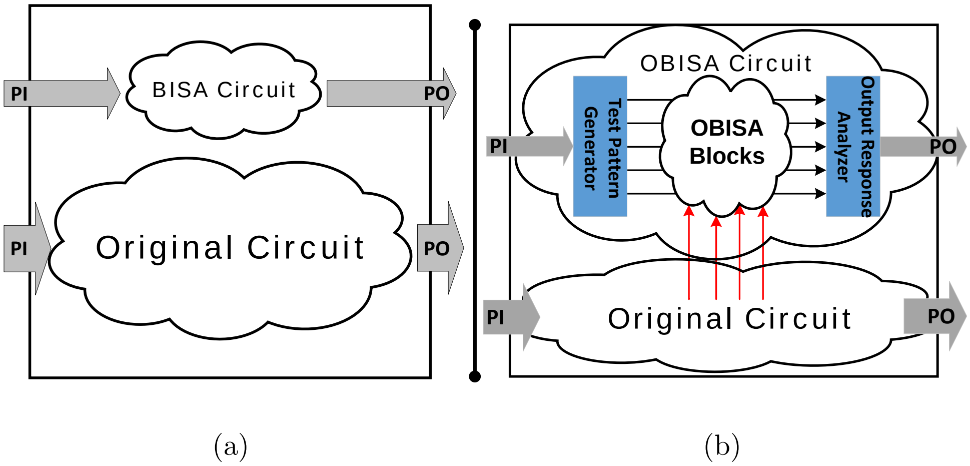


**Fig. 6.2:** (a) BISA structure and (b) the proposed OBISA structure for split manufacturing in this paper.

### 6.3.2 OBISA Structure

Figure 6.2(b) shows the structure of the proposed OBISA. It consists of a test pattern generator (TPG), an output response analyzer (ORA) and OBISA blocks under test. The entire OBISA circuit is implemented in unused spaces of the layout. In this paper, we use a linear feedback shift register (LFSR) as TPG and multiple input shift register (MISR) as ORA. They are used to generate random test patterns and compress responses to generate a final signature, respectively [99].

Two operating modes are associated with the proposed technique. In *functional mode*, the original circuit is operating normally, but clocks for BISA circuits are disabled and LFSR and MISR are reset to their pre-defined state. A gating mechanism presented in Section 6.4.3 will block signals from activating original circuits to OBISA circuit. Therefore, OBISA circuit stays quiet and does not consume any dynamic power. In addition, these idle OBISA cells can fulfill the role of decap cells [75] [76]. Note that

OBISA circuitry will, however, consume leakage power. The other mode, *authentication mode*, is used to authenticate a fabricated chip in the field. In this mode, LFSR generates a random test pattern at every clock cycle and the test pattern is shared by all OBISA blocks. At the same time, MISR collects responses from OBISA blocks and eventually produces a signature, as shown in Figure 6.2(b). The test phase ends when a sufficient number of test patterns have been applied. Since inputs to OBISA come from LFSR and original circuits, thus test patterns for OBISA are generated depending on the LFSR and the state of the original circuit. We can keep the state of original circuit and run a large number of clock cycles on LFSR to test the inserted circuit as one iteration. In the next iteration, we change state of the original circuit and perform the tests once again. Test coverage will increase as more iterations are performed. Functional simulation at the OBISA design phase could help us find an efficient combination of the circuit state and the seed in the LFSR.

Note that clock is provided externally either in normal mode or in authentication mode. Typically, authentication test clock is much slower than functional clock [99]. The authentication clock frequency is dependent on the number of gates in an OBISA block and the gate types. The timing constraint can be obtained using post-layout timing analysis tool to find the longest path after OBISA inserted circuit. In the authentication mode, a very slow test clock can be used to ensure no path fails in OBISA circuits.

### 6.3.3 Security Analysis

Split manufacturing obfuscates the design by preventing an untrusted foundry from gaining a full view of the layout. Our proposed OBISA structure can improve obfusca-

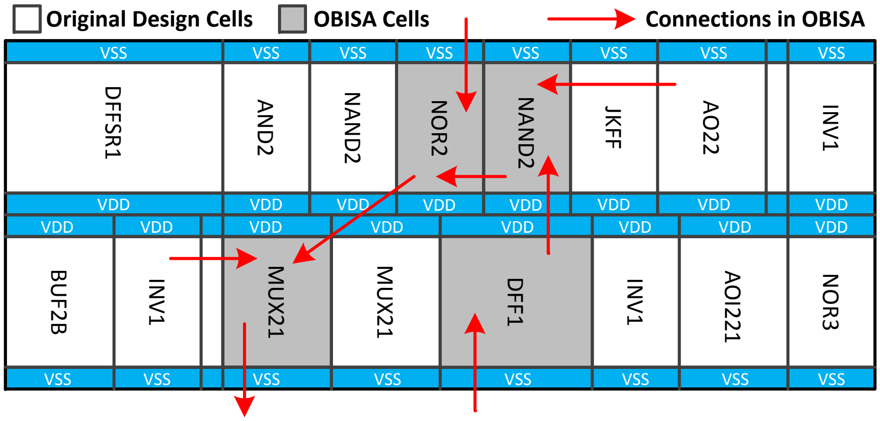


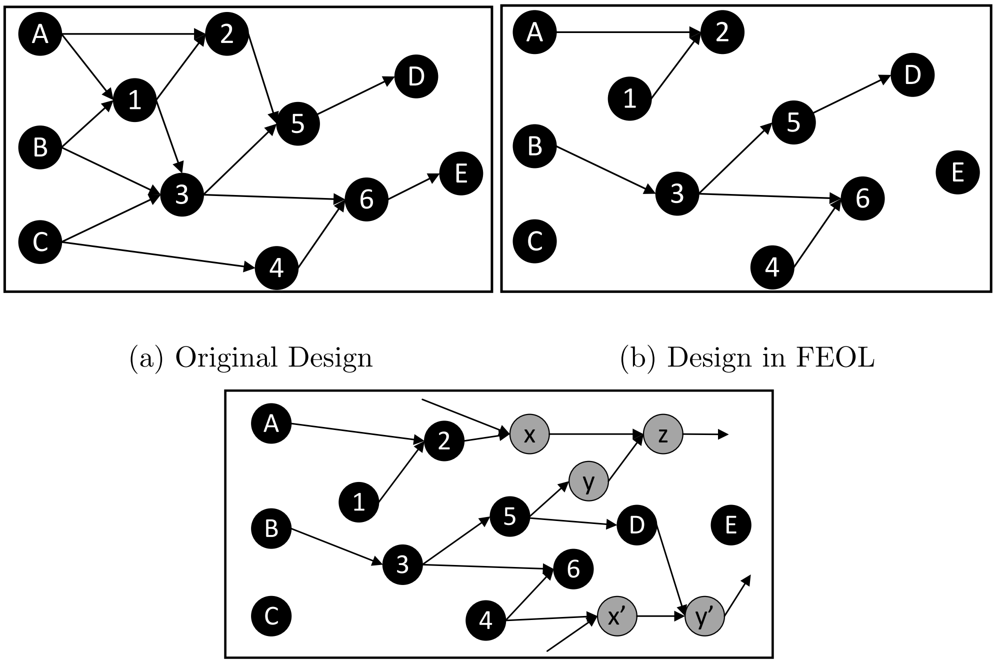
**Fig. 6.3:** Layout after OBISA insertion.

tion through the following features:

- Functional filler cells are inserted into unused spaces during layout design, as shown in Figure 6.3. Original cells and additional OBISA cells are all standard cells from the same technology library.

- Additional cells are connected together to construct combinational circuits. There is no difference in routing between original circuits and inserted OBISA circuits. While tree-structure circuit (no fan-out) is required for a BISA block in [75] [76] in order to ensure a high test coverage, fan-out is allowed in this new OBISA technique. An approach for fan-out creation in OBISA blocks is proposed in Section 6.4.3.

- Interconnections between OBISA circuits and original circuits (obfuscation connections) are allowed. The additional logics can further obfuscate the original design, especially for the security-critical parts. For example, Figure 6.4(a) shows a design. Split fabrication can hide some wires in FEOL, but some sub-circuits



(a) Original Design

(b) Design in FEOL

(c) Design with inserted OBISA circuit (gray) in FEOL

**Fig. 6.4:** Circuit graphs: OBISA circuit is used to obfuscate the original design.

(cell 3, 4, 5 and 6) could be identified. In Figure 6.4(c), the proposed technique can add OBISA cells to protect this sub-circuit from reverse-engineering.

- The inserted OBISA includes LFSR and MISR. We understand that LFSR and MISR have a unique structure and are controlled by mode select port, so they could be a target by an adversary to identify the additional OBISA circuit. Split manufacturing can hide critical wires in LFSR and MISR and effectively obfuscate LFSR and MISR.

- Additional local/global connections and various gates introduced by OBISA circuits can hinder neighbor connectedness analysis and standard-cell composition bias analysis [111]. OBISA blocks have not only local connections to connect cells nearby, but also long connections to other OBISA blocks, LFSR and MISR. A measure of spatial connectedness will be influenced by OBISA circuits. Similarly, additional cells with different types can change the types and proportions of cells of design presented in a small region in layout. OBISA cells can obfuscate the cell composition analysis.

- The proximity attack is based on the heuristic that floorplanning and placement (F&P) tools place the partitions close by and orient the partitions so as to reduce the wiring (delay) between the pins to be connected. [114] shows that the proximity attack could be a threat of connecting the missing nets correctly. However, OBISA circuit is able to produce additional tier-to-tier connections which can mitigate the threats of proximity attacks.

## 6.4 Implementation Strategy

### 6.4.1 Implementation Flow

Figure 6.5 presents our proposed OBISA implementation flow. The flow fits within the conventional ASIC design flow and is computable with current commercial physical design tools. OBISA insertion procedure begins after clock tree synthesis. At that point, the whole original circuit has been placed and no more cells will be added in conventional flow (the most left column in Figure 6.5). The unused spaces would be identified in DEF file and various standard cells are inserted depending on size of each unused space. More information regarding OBISA cell insertion will be described in Section 6.4.2. Once all unused spaces are filled with OBISA cells, we will begin to connect all OBISA cells in a region to construct a number of tree-structure combinational circuits (referred to as OBISA blocks). These steps as shown in the middle column were developed in [75] [76]. The steps in the third column are proposed to strengthen obfuscation, including fanout creation in tree-structure OBISA blocks, adding obfuscation connections, and lifting secure-critical paths within OBISA. Detailed connection strategies for each of them will be presented in Section 6.4.3. After the OBISA process, the flow resumes the procedure in conventional design flow. The physical design tool will perform routing for the entire design including original circuit and OBISA circuit. All constraints for the original design can be taken care of by the physical design tool during routing process. Once the timing and sign-off of the design are successful, the last step involves the generation of a GDSII format of the design for final tape-out.

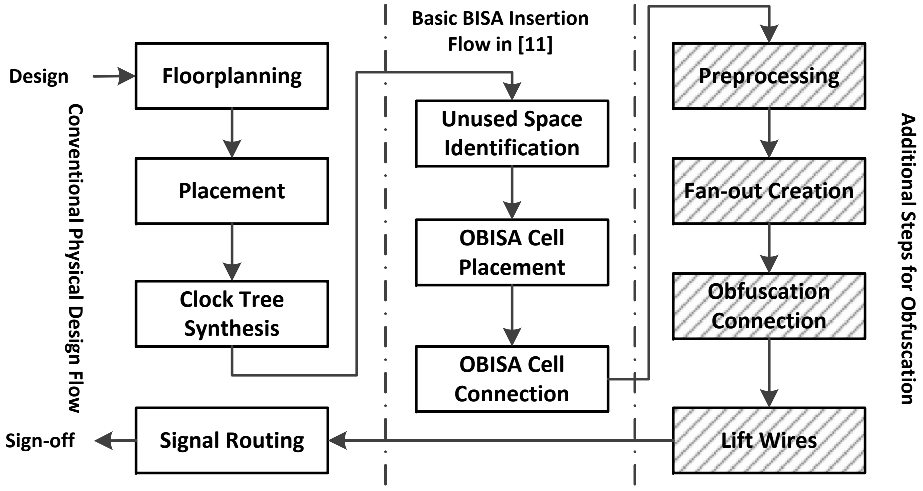


**Fig. 6.5:** The OBISA implementation flow.

### 6.4.2 OBISA Cell Insertion Strategy

Several filling algorithms are proposed to fill every unused space as much as possible and no more cells could be added without changing the layout [75] [76]. Cells with different types and sizes are inserted to ensure a variety of OBISA cells due to the following reasons:

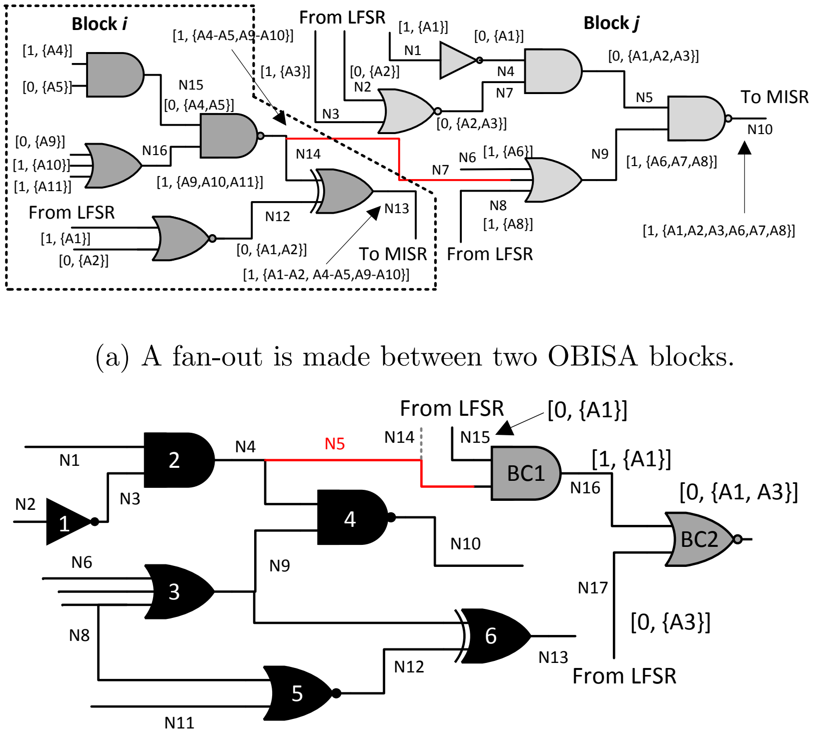
- A greater variety of OBISA cells can effectively thwart the cell composition analysis [111]. This is analogous to “dummy” cells in prior work.
- Our OBISA also supports either pre-mask or post-mask Engineering Change Order (ECO). Unused spaces are filled with a variety of standard cells and all these cells can be treated as spare logic gates. When an OBISA cell is selected for ECO, a few modifications are needed to bypass this cell in OBISA block. Since OBISA circuits do not have a certain timing constraint, routing for OBISA could be very flexible.

### 6.4.3 OBISA Cell Connection Strategy

The OBISA structure allows adding fan-out and obfuscation connections. There are two points we focus on: testability of the OBISA circuit and effect on obfuscation strength. Our connection strategies are presented as follows:

#### Preprocessing

Two pieces of information are required to collect for each net in OBISA blocks: idle state (IS) and related inputs (RI) in LFSR. As we described earlier, OBISA circuits



(a) A fan-out is made between two OBISA blocks.

(b) An obfuscation connection is made.

**Fig. 6.6:** Additional connections for improving obfuscation.

are working in authentication mode only, so LFSR will be set to a specific state if the chip is in the other mode. We propose to set the state of LFSR to a vector that has alternative ‘1’ and ‘0’ to avoid biasing values in OBISA circuit. If an input is connected to the original design because of the obfuscation connection, its IS is undetermined (‘X’). Therefore, the IS for each net can be ‘1’, ‘0’, or ‘X’, based on input values and their interconnections. IS can determine if a cell could be a gated cell (see Section 6.4.3). Another required information is the RI for each net. Taking the OBISA circuit in Figure 6.6(a) for example, the RI for the net N7 are A2 (N2) and A3 (N3), because nets N2 and N3 determine the value on net N7, while the related inputs for net N5 are

A1, A2, and A3. Information regarding RI can help us decide how to create fan-out without losing any test coverage (more in Section 6.4.3). In this paper, we use a symbol, [x, {A,B,C}], to represent IS (x) and RI (A,B,C) of a net. One can see the IS and RI for the nets in OBISA circuits are labeled in Figure 6.6.

### Fan-out Creation

Adding fan-out could potentially produce redundant gates and thereby lower controllability of gates. Fan-outs can be created by following the rules:

- The fan-out is created between a net in one block $i$ and an input pin of another block $j$ ($i \neq j$).
- The net in one block $i$ and the root output in block $j$ have no common RIs.

If these two conditions are satisfied, the net and the input pin can be a candidate pair for a fan-out. During the fan-out creation, the original connection on the input pin in block $j$ from LFSR should be removed, so the pin is available for the new fan-out connection. These two tree-structure OBISA blocks $i$ and $j$ share a sub-tree circuit so that they are still tree-structure blocks in nature. Therefore, the added fan-out will not result in any extra redundant gates in OBISA blocks. Each net is still controllable, so a high test coverage of OBISA circuit will be achieved. Figure 6.6(a) shows an example of the fan-out creation. The net N14 in OBISA block $i$ has completely different RI from the root output N10 in OBISA block $j$, so N14 can have a fan-out to connect any input in OBISA block $j$. In Figure 6.6(a), the pin for the net N7 is selected. Note that a fan-out cannot be made on the net N13, because the net N13 and N10 have shared RIs, A1 and A2.

### Gated Cells

For logic gates, the dominant value on one input can result in a deterministic output regardless of logic values on other inputs, so those gates are gated by their dominant values. For example, ‘1’ and ‘0’ are used to gate OR and AND cells, respectively. We will take advantage of this for preventing dynamic power consumption within OBISA circuitry. An obfuscation connection must connect to a gated cell in OBISA circuit. For a gated cell that acts as an interface, at least one input’s IS should be its dominant value during the normal operation. Other inputs of the gated cell can connect any net in the original circuit. In order to minimize modifications caused by the obfuscation connections, we prefer to choose a leaf cell for gated cell, because its input connections from LFSR can be removed without changing existing OBISA blocks. In Figure 6.6(b), cells BC1 and BC2 are both gated in idle state. BC1 is selected since it is a leaf cell in the tree-structure OBISA block. All gated cells will be identified at this step, and they could be selected for obfuscation connections, as described in the next subsection.

### Obfuscation Connection

Obfuscation connection also introduces additional interconnections between OBISA circuits and original circuits. Figure 6.6(b) shows an example of an obfuscation connection. It will cause inevitable increased capacitance on connected nets. Although we do not worry about the timing in OBISA circuits because of the slower frequency used during the authentication for OBISA, the added capacitance could potentially fail paths in the original design. Thus, we must select connection nets very carefully to avoid timing violations. Here, we propose an approach to select nets in the original circuit for obfuscation connection based on delay estimation by the static timing analysis (STA) tool.

- We define a parameter $C_0$, which is a threshold for dividing paths into critical paths and non-critical paths, as shown in Figure 6.7.
- Given the $C_0$, the STA is able to find all critical paths in original design. All nets on critical paths will be excluded from obfuscation connection. All remaining nets will be assigned a virtual path delay $C_0$. We call it virtual path delay, because it is not a real delay. Many non-critical paths have much smaller delay than $C_0$, but we treat them as the same length to simplify the problem.
- Another parameter $C_1$ is defined as a threshold to select net for obfuscation connection. The $C_1$ should be smaller than functional clock period $C_2$. The difference of $C_1$ and $C_2$ is a safe margin that ensures no timing violation is produced due to our rough estimation.
- If an obfuscation connection is made on a net, its virtual path delay will add an increased delay $D$. The increased value $D$ will vary depending on the technology libraries. It can be estimated by averaging some samples in simulations. A large enough safe margin $(C_2 - C_1)$ can tolerate such a rough calculation.
- A net can be considered for an obfuscation connection if its virtual path delay plus $D$ is still smaller than the threshold value $C_1$. For example, in Figure 6.7,

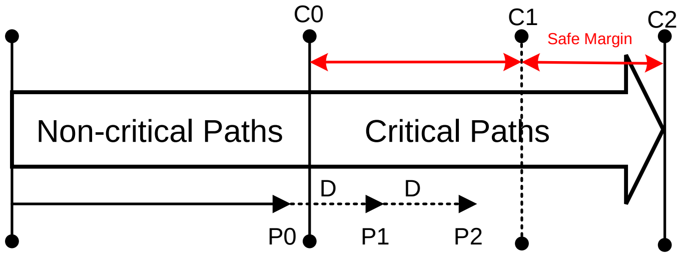


**Fig. 6.7:** Net selection for obfuscation connection.

path P0s added $D$ is still smaller than $C_1$. In the example of Figure 6.6(b), the net N4 is selected for the obfuscation connection at this step.

- If there is an available gated cell nearby (BC1 in Figure 6.6(b)), an obfuscation connection can be made from the net (N4) in original design to one input (N14) of the gated cell.

- The increased capacitance not only influences one net, but also affects all nets on paths that pass this net. Therefore, the virtual delay of these relevent nets will be updated by adding $D$. For the circuit in Figure 6.6(b), the net N1, N2, N3, N5, and N10 need to be updated. For the net in Figure 6.7, there are two obfuscation connections that push its virtual delay from P0 to P2.

The value of increased delay $D$ depends on the length of added wire. [38] shows that adding a short connection can bring about 30ps increased delay with 90nm technology. Since the obfuscation connection is made between nets not far away from each other, the increased delay $D$ will not be very large. We can obtain a conservative $D$ value in simulations.

**Hide LFSR/MISR with Lifted Wires**

Since there is one primary input that selects either function mode or test mode, an adversary could trace from this port to identify LFSR/MISR and further find OBISA cells. In order to avoid this threat, the mode select net, the feedback nets in LFSR/MISR, and some nets connecting flip-flops in LFSR/MISR should be lifted to trust tier, so attackers cannot have any opportunity to identify them.

## 6.5 Experimental Results

### 6.5.1 OBISA Implementation

The OBISA technique was evaluated using Opencore benchmark circuits. Each circuit was synthesized with 90nm CMOS technology using Synopsys Design Compiler. Physical design, including floorplanning, placement, and routing, were conducted using Synopsys IC Compiler. Scripts were developed to analyze unused spaces in layout, insert OBISA cells, and connect them using our proposed methodologies, including fan-outs and obfuscation connections.

Table 6.2 shows the implementation results of five Opencore benchmark circuits with various scales. The number of OBISA cells for each circuit is listed in the third row of the table. Those OBISA cells do not introduce any area overhead because they are placed in the unused spaces in layout. Fan-outs and obfuscation connections are created between OBISA blocks to obfuscate their tree structures and increase the difficulty for adversaries to identify them in layout. An average 5% of nets go through trusted tier that is above M3 layer, as shown in the seventh row of the table. However, a small number of obfuscation connections are sufficient to make the inserted OBISA circuits look like a part of original design, to enhance obfuscation in FEOL. Since the test vector from LFSR has alternative '1' and '0', almost all leaf cells could be a gated cell. In our implementations, we use percentage to quantify how many inputs are altered for obfuscation connection in an OBISA circuit. Table 6.2 shows results with around 5% obfuscation connections (OCs). Thus, for an OBISA block with 80 inputs, there are 4 inputs connected to original circuit as obfuscation connections. Since all the OCs are made on short paths in the original circuit, no timing violations are introduced by OCs.

**Table 6.2:** Implementation results on different benchmark circuits.

| Benchmark | DES3 | USB | AES | Ethernet | DES_perf |
|---|---|---|---|---|---|
| Total Cell # | 1,559 | 6,445 | 26,447 | 29,153 | 49,517 |
| OBISA Cell # | 158 | 439 | 2,950 | 1,169 | 2,090 |
| OBISA Cell Pct (%) | 10.1% | 5.8% | 11.1% | 4% | 4% |
| Total Net # | 1,799 | 7,709 | 28,505 | 29,981 | 49,951 |
| Secure Net # (≥M4) | 137 | 306 | 2,138 | 705 | 1,353 |
| Secure Net Pct (%) | 7.6% | 4% | 9.5% | 2.4% | 2.7% |
| Fan-out # | 30 | 52 | 134 | 84 | 106 |
| 5% OC # | 15 | 40 | 256 | 106 | 189 |

### 6.5.2 Authentication Test Coverage Analysis

The authentication test coverage is a metric to assess the security level of the circuit. A higher stuck-at test coverage for OBISA circuits indicates that more OBISA cells could be verified by structural test patterns. The target coverage is 100%. According to our proposed flow in Section 6.4.1, all inserted OBISA cells will be connected in a tree-structure manner first. Then fan-outs and obfuscation connections are added based on the existed OBISA blocks. We take the OBISA circuitry in the Ethernet benchmark circuit as a running example to compare the test coverage across different test patterns in five scenarios: *only tree-structure OBISA blocks*, *OBISA blocks with fan-out*, *BISA blocks with fan-out and three different proportions (5%, 15%, and 25%) of obfuscation connections*. For each scenario, four kinds of test patterns, 50,000, 100,000, 10,000 random patterns with different iterations, and ATPG (automatic test pattern generation) patterns, are applied to the OBISA circuits separately, and the results are shown in Figure 6.8. From the two left columns, we can see tree-structure OBISA circuit with and without fan-outs have the same test coverage using either random patterns or ATPG patterns. It demonstrates that the proposed fan-out creation method will not affect test coverage. The ATPG patterns can achieve almost 100% test coverage. For the random patterns from LFSR, the test coverage goes up as more patterns are applied. Their test coverage for 50,000 and 100,000 random patterns are 99.81% and 99.97%, respectively. The remaining three columns illustrate that test coverage will be impacted by obfuscation connections. This is because signals on obfuscation connections are tied to a constant value in one iteration and thus result in the controllability loss of gated cells. The last column shows an extreme scenario that 25% of inputs in OBISA blocks are connected to original design. The test coverage within 1-iteration are not high enough for the authentication test. However, as we described in Section 6.3.2, if we change the status of original circuits with multiple iterations, i.e. the values on obfuscation connections could change, the controllability of gated cells can be compensated to some degree depending on how many iterations can perform. Results in Figure 6.8 show that multiple iteration offers much better test coverage than the 1-iteration with the same number of test patterns, while using 10-iteration test leads to a higher test coverage than that with 5-iteration tests. The test coverage can improve further if more random patterns are applied or more iterations are performed.

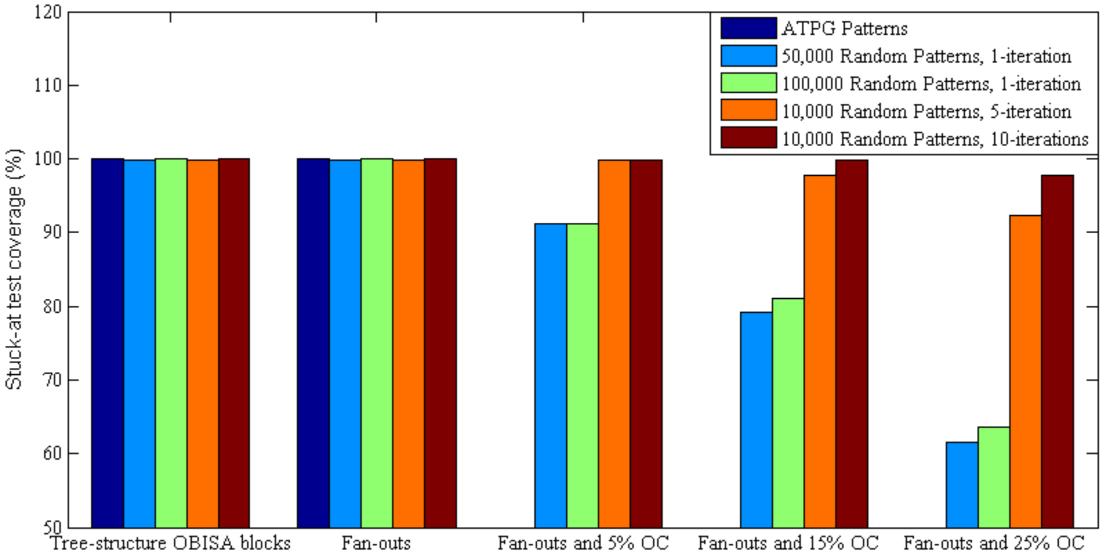


**Fig. 6.8:** Authentication test coverage for an OBISA circuit.

## 6.6 Conclusion

In this chapter, we present a low-cost and secure OBISA technique. Split fabrication and OBISA circuit can effectively protect layout design from reverse engineering, IP piracy, and tampering. The OBISA technique fills unused spaces in layout with functional circuitry while it is also connected to the original design, in order to make the FEOL extremely difficult for an untrusted foundry to identify the added cells. The dynamic power and timing overhead is small because of net selection and gated cell selection for the obfuscation connections. The OBISA circuit can be tested in the field for hardware Trojan detection. A fan-out creation approach is proposed to obfuscate OBISA blocks further and will not affect its original high test coverage.

# Chapter 7

## Security Primitive Enhancement: Bit Selection Algorithm Suitable for High-Volume Production of SRAM-PUF

With the rapid development of information technology, electronic devices (smart phone, GPS, etc.) are playing a larger role in our daily lives. Physical Unclonable Functions (PUFs) have been proposed as a more secure alternative to conventional ID storage because of their unclonability and built-in tamper evidence [28]. These different PUFs have various pros and cons, but among them, SRAM PUFs [119] [120] [121] [122] are quite popular for several reasons: (i) SRAM is a standard component available in many technology nodes; (ii) SRAM is already utilized in many systems as a volatile memory so designers do not need to add more dedicated hardware for the PUF; (iii) Typical SRAMs have large amount of space and can easily provide sufficient PUF outputs to generate ID/keys long enough for devices produced at high-volume.

Despite its advantages, however, the stability of SRAM PUF output is heavily impacted by environmental variations and aging effects. To ensure the reliability of the SRAM PUF response, the output needs to either be error corrected in post-processing or analyzed in such a way that only stable cells are used [123]. ECC uses specialized encoding and decoding processes to ameliorate data instability. However, such processes create considerable overhead for implementation. For example, [133] shows ECC requires additional $\sim 87k$ gates for 128-bit reliable key. In addition, ECC leaks information and results in high design costs that ultimately limit the use of PUFs in resource-constrained products. To reduce the overheads, an alternative is to reduce sensitivity to temporal variations. For instance, a reliability enhancement [121] was developed to reinforce the preferred value of SRAM cell by inducing accelerated aging. But performing burn-in stress for 120 hours for one SRAM PUF is prohibitively expensive, time-consuming, and in turn degrades the SRAM. Another proposed approach modifies the VDD ramp-up time to make cells more reliable [120], but this approach requires special circuitry not present in standard SRAM.

In this chapter, we take a new approach that selects the most stable bits for PUF IDs/keys with the goal of reducing or possibly removing the need for ECC altogether. A random approach might use exhaustive testing at all environmental conditions/corners and after aging to choose the bits that are stable, but this is ill-suited for mass-produced devices where test time and cost are major concerns [131]. We make the following contributions in this Chapter:

- We conduct preliminary experiments to gather large amounts of real data from an SRAM. We analyze the impact of environmental variations, temporal variations (aging by burn-in), and spatial correlation (neighborhood analysis) on the stability of PUF bits. Our neighborhood analysis reveals that the most stable bits from enrollment depend on the stability of their neighbors, i.e. those surrounded by other stable bits flip less over time.

- Based on our observations, we develop low-cost enrollment tests that only require two corner conditions (high-temperature low-voltage and low-temperature low-voltage) and no burn-in tests. It is more efficient for stable cell selection and is relatively low-cost comparing to exhaustive tests at all conditions.
- We compare the uniqueness, reliability, etc. of our PUF IDs/keys for fresh and aged SRAM chips with IDs/keys from typical approaches. Our results show that our IDs have less bit errors over time, thereby requiring much less ECC overheads.

## 7.1 Background and Preliminaries

### 7.1.1 A Typical Protocol for Systems with Intrinsic SRAM PUFs

Figure 1 illustrates a typical authentication protocol for devices with SRAM PUF. The protocol consists of two phases: (a) *enrollment* and (b) *reconstruction* [124]. An SRAM chip is integrated in the device (on-board or in package). Before the device is delivered to market, the device is “enrolled”. Specifically, the SRAM PUF output (or a portion of it) is read by the system owner and then stored in a database. If only a portion of it is stored, then the database will also store the SRAM addresses corresponding to that portion.

In the reconstruction phase, the device starts the protocol by sending its static ID (publicly available) to the database. The database then searches for the corresponding challenges (addresses) associated with the static ID and sends them back to the device. The device collects the SRAM outputs (which may be noisy) and sends them to the database for authentication/identification. The database compares them with the genuine output stored in its database. Compared to the conventional approaches that use bar-codes or embedded IDs/keys into a device, the PUF approach is much more secure [28] [129] [130]. As long as each PUF generates a unique ID/key, each device will be authenticated properly.

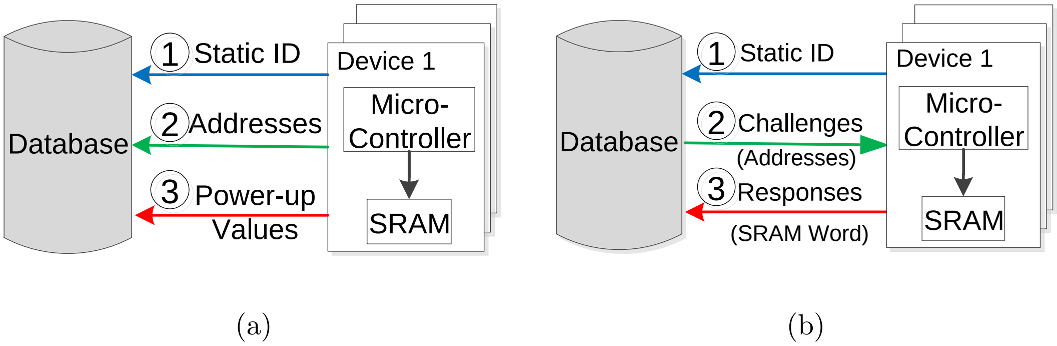


**Fig. 7.1:** SRAM PUF (a) enrollment and (b) reconstruction

### 7.1.2 Operation and Salient Features of SRAM PUF

For a standard six-transistor SRAM, every memory cell is composed of six transistors that are two cross-coupled CMOS inverters and two access transistors [94]. Each of the inverters drives one of the two state nodes, labeled “A” and “B” in Figure 7.2. When the circuit is unpowered, both state nodes are discharged low (AB=00). When power is applied, this unstable state will transition to one of the two stable states, either “0” (AB=01) or “1” (AB=10); the AB=11 state is unstable and unreachable [119]. The tendency to transition to one state or the other depends on the threshold voltages of transistors and noise. Those two transistors are designed to be symmetric, match in size, etc., but random variations incurred during manufacturing will result in random mismatches. SRAM PUFs exploit the parameter mismatch which results in each SRAM cell being biased (or skewed) toward a zero or one at power-up. It has been observed that certain cells have a strong “preference” to power-up to a ‘1’ or ‘0’ state. Cells that have no “preference” are deemed neutral and power-up at random depending upon the influences of environmental noise [132]. The more useful cells for PUF output are the ones that strongly prefer ‘0’ or ‘1’.

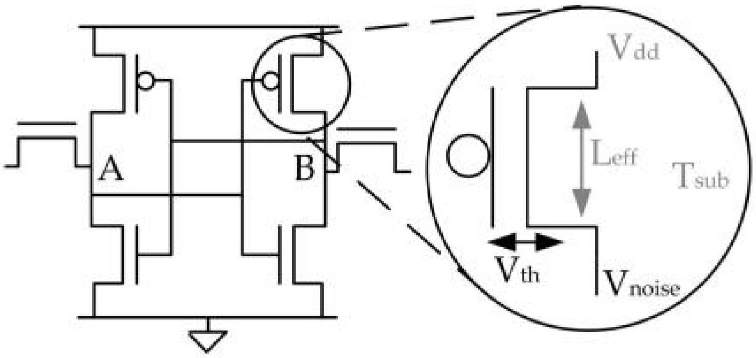


**Fig. 7.2:** A Six-transistor SRAM cell with relevant process variation and noise.

In addition to process variations, the power-up value can also be influenced by temporal variations, such as voltage supply variation, operating temperature, aging effects (such as negative bias temperature instability (NBTI) and hot-carrier injection (HCI)), etc. The temporal variations can cause cells to start-up in a different state, causing the ID generated by the SRAM PUF to change over time. For illustrative purposes, both process variation and noise are considered as impacting the *skew* of a cell. The skew of a cell is a continuous quantity used to represent the power-up tendency of that cell. Skew at a given power-up is influenced by noise, so the skew of each cell across many power-ups can be described by a probability distribution function, as shown in Figure 7.3.

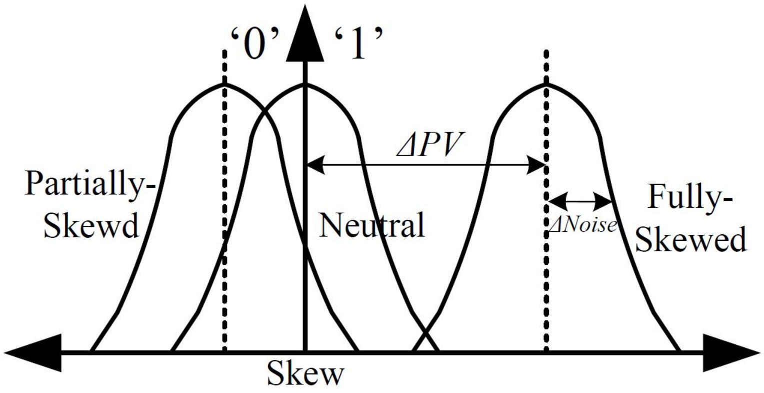


**Fig. 7.3:** The influence of process variation and noise on cell's skew.

Ideally, the SRAM PUF output would be stable and unique if only process variation exists. However, in reality, the ID/keys will change over time because of temporal noise and aging degradation over time. Therefore, if the magnitude of process variation of one cell is large enough to safely offset temporal noise, this cell would be able to produce a reliable power-up value despite environmental noise and even degradation due to aging, like the fully-skewed cell shown in Figure 7.3. Since SRAM has a large number of memory cells to use as PUF, there exist sufficient fully-biased cells (large process variation) that could potentially produce a very stable key bit. Our objective then becomes development of techniques that are low-cost in terms of design and test overhead for identifying the most stable SRAM PUF bits.

## 7.2 Preliminary Experimentation and Observations

In this section, we will explore the impact of supply voltage, ambient temperature and aging effects on power-up values (i.e., reliability of the ID/key) using real SRAM measurements. We also analyze the dependence of location on the stability of SRAM cell power-up values. From the observations we make from this data, we will develop low-cost techniques in latter sections for choosing the most stable bits for PUF ID/key.

Experiments are performed on a Xilinx Spartan-3 FPGA board with a 1MB on-board ISSI SRAM. Here, we focus on 6KB of the SRAM, because a small portion of SRAM can provide enough bits to construct a PUF. Additionally, since every SRAM cell is designed to be completely identical, the characteristics of a portion of SRAM cells can be extended to other memory cells in this SRAM as well as others.

### 7.2.1 Stability Under Power Voltage Variation

First, the PUF output is collected under supply voltage variations. Since the operation voltage range of the on-board SRAM is ±10% of the nominal voltage 3.3 volt, many measurements are taken at different core voltages from 3.0v to 3.6v. In order to illustrate the changes of power-up values under voltage variations, 20 measurements at low voltage (3.0v) and high voltage (3.6v) are taken (10 from each voltage). For every memory cell, the frequency of producing a ‘1’ is calculated based on 10 measurements at the same voltage. A frequency of 1 (0) indicates these cells are producing stable ‘1’ (‘0’) in all ten measurements, while other values between 0 and 1 indicate that cells produce some ones and some zeroes in the 10 measurements (i.e., were not stable). Then, we analyzed the change of distribution of cell skew as voltage varied from low to high.

Table 7.1 depicts the distribution. The first row (column) in Table 7.1 indicate the frequency of producing ‘1’ at high (low) voltage. The fractions in the table represent the percentage of the cells that have a particular combination of frequencies producing ‘1’ at low and high voltages. For example, the number in the fourth row and third column says that about 0.31% cells produced a ‘1’ at HV and LV with frequencies between 0-0.2 and 0.4-0.6 respectively. The table shows that 66.40% (34.06% + 32.34%) of cells generate either a stable ‘0’ or ‘1’ across the measurements. The top right and bottom left elements in Table 7.1 are both 0 which indicates that none of the bits that were stable ‘0’ (‘1’) at HV became stable one (zero) at LV and vice versa.

**Table 7.1:** Distribution of cell skew at high and low voltages.

| **LV/HV** | **0** | **0∼0.2** | **0.2∼0.4** | **0.4∼0.6** | **0.6∼0.8** | **0.8∼1.0** | **1** |
|---|---|---|---|---|---|---|---|
| **0** | 34.06 | 2.57 | 0.17 | 0.02 | 0 | 0 | 0 |
| **0∼0.2** | 1.96 | 3.93 | 1.49 | 0.27 | 0.04 | 0.01 | 0 |
| **0.2∼0.4** | 0.16 | 1.36 | 2.09 | 1.15 | 0.29 | 0.01 | 0.01 |
| **0.4∼0.6** | 0.01 | 0.31 | 1.12 | 1.79 | 1.08 | 0.17 | 0.04 |
| **0.6∼0.8** | 0 | 0.05 | 0.30 | 1.37 | 2.68 | 0.90 | 0.50 |
| **0.8∼1.0** | 0 | 0 | 0.03 | 0.21 | 1.52 | 0.41 | 1.42 |
| **1** | 0 | 0 | 0 | 0.03 | 0.65 | 2.46 | 32.34 |

### 7.2.2 Stability Under Temperature Variation

Similar experiments were conducted by sweeping temperature from 0°C to 80°C using our Themostream system. We use the same approach as above to analyze the distribution changes introduced by temperature variations (see Table 7.2). Our data indicates that 60.33% (30.79%+29.54%) of cells are stable regardless of temperature, and 14.06% of memory cells change to the opposite skew state due to temperature change. Among them, 4.79% (2.24%+2.55%) memory cells change from stable zero to stable one or stable one to stable zero). Our experiments demonstrate that temperature variations have much greater impact on bit stability than supply voltage variations for SRAM PUF.

**Table 7.2:** Distribution of cell skew across temperature.

| **LT/HT** | **0** | **0∼0.2** | **0.2∼0.4** | **0.4∼0.6** | **0.6∼0.8** | **0.8∼1.0** | **1** |
|---|---|---|---|---|---|---|---|
| **0** | 30.79 | 1.12 | 0.65 | 0.47 | 0.55 | 0.39 | 2.24 |
| **0∼0.2** | 4.22 | 0.53 | 0.36 | 0.28 | 0.35 | 0.23 | 2.03 |
| **0.2∼0.4** | 2.30 | 0.29 | 0.20 | 0.16 | 0.20 | 0.15 | 1.56 |
| **0.4∼0.6** | 1.77 | 0.30 | 0.15 | 0.18 | 0.21 | 0.20 | 1.75 |
| **0.6∼0.8** | 2.06 | 0.34 | 0.22 | 0.23 | 0.27 | 0.27 | 2.43 |
| **0.8∼1.0** | 1.46 | 0.34 | 0.17 | 0.17 | 0.23 | 0.19 | 2.43 |
| **1** | 2.55 | 0.67 | 0.55 | 0.59 | 0.81 | 0.80 | 29.54 |

### 7.2.3 Stability Under Aging Effects

In this section, we look at the stability of cells as the SRAM ages. To accelerate the aging, we perform burn-in of the SRAM using Thermostream burn-in system. Specifically, we execute write ‘0’, write ‘1’ ,and read operations alternately to/from every memory cell under high temperature (80°C) and high power voltage (3.6) conditions. In Table 7.3, we compare two measurements performed under nominal conditions (room temperature + nominal voltage) on the fresh SRAM and the SRAM after 2 hours of accelerated aging. Table 7.3 shows that 59.26% (29.82%+29.44%) of the bits are stable and approximately 1.89% (0.88%+1.01%) of cells change from one stable state to the opposite stable state because of aging effects.

### 7.2.4 Stability of Neighboring Cells

Prior work has shown that neighboring chips on the same wafer undergo similar processes and correlate to each other. Hence, chips have the same fault-free (or conversely faulty) properties as their neighbors in the same wafer [125]. The spatial correlation of

**Table 7.3:** Distribution of cell skew for fresh and aged SRAM.

| **A/F** | **0** | **0∼0.2** | **0.2∼0.4** | **0.4∼0.6** | **0.6∼0.8** | **0.8∼1.0** | **1** |
|---|---|---|---|---|---|---|---|
| **0** | 29.82 | 2.14 | 0.84 | 0.63 | 0.62 | 0.38 | 0.88 |
| **0∼0.2** | 3.94 | 1.27 | 0.622 | 0.56 | 0.58 | 0.40 | 1.03 |
| **0.2∼0.4** | 1.80 | 0.72 | 0.50 | 0.40 | 0.47 | 0.35 | 1.08 |
| **0.4∼0.6** | 1.18 | 0.62 | 0.41 | 0.42 | 0.49 | 0.37 | 1.24 |
| **0.6∼0.8** | 1.08 | 0.66 | 0.47 | 0.49 | 0.64 | 0.51 | 2.05 |
| **0.8∼1.0** | 0.55 | 0.37 | 0.26 | 0.34 | 0.54 | 0.48 | 2.37 |
| **1** | 1.01 | 0.80 | 0.64 | 0.81 | 1.28 | 1.52 | 29.44 |

parameters among cells within a wafer has been investigated in [127] [128].

In this section, we investigate whether the same relationship holds for predicting the reliability of SRAM cells for PUFs by performing the following test. We took 10 measurements at random environmental conditions for the entire SRAM. Based on the output, we labeled the cells as "stable" if they produced the same value for all 10 measurements or as "unstable" otherwise. For each stable and unstable cell, we looked at a 20 cell "window" around the cells (the 9 cells before and 9 cells after it) and counted the number of unstable cells in the window. Figure 7.4 shows the distribution of the count for stable and unstable cell cases fitted with Gaussian distributions. The distribution for stable cells is closer to zero meaning they have less unstable bits surrounding them. This test shows that the stability of each cell/bit *is* correlated with its neighborhood: the more stable cells/bits in the neighborhood, the more stable/reliable the cell will be as a PUF.

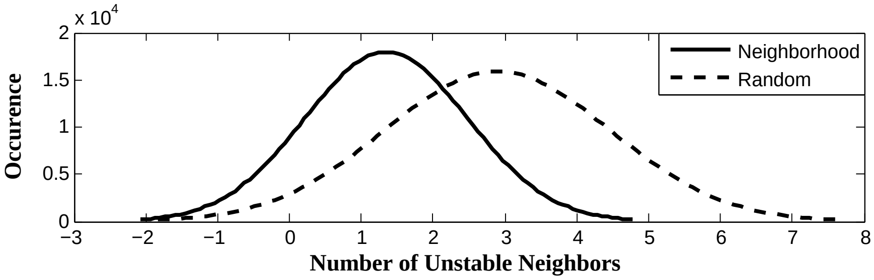


**Fig. 7.4:** Stable neighbors provides better reliability than random selection.

## 7.3 Proposed Enrollment and Bit Selection Techniques

There are two primary observations we have obtained from Section 7.2: (i) temperature variation can introduce more flipped bits than voltage variation and aging; (ii) the power-up stability of a cell is related to its neighboring cells. We take advantage of these characteristics to identify very reliable cells with a reduced set of enrollment tests and a neighborhood-based bit selection.

### 7.3.1 Critical Test Conditions for Enrollment

Due to instability of power-up values under environment variations, we want to identify stable cells and exclude two types of unstable cells, partial-skewed cells and neutral-skewed cells. Partial-skewed cells are stable in most of power-up operations and have a small possibility to flip. By changing environment condition to make the probability distribution shift to neutral location, some partial-skewed cells will be much easier to flip. Power voltage and temperature are two major environmental factors we can change in the enrollment tests with relatively low cost. According to the experiment results in Section 7.2, we know that temperature can contribute more to the change of cell's skew.

The impact of temperature on MOSFET devices is well studied in literature. An increase in temperature decreases device threshold voltages:

$$V_{th}(T) = V_{th}(T_0) - \kappa \Delta T$$

Therefore, $V_{th}$ has a linear relationship to temperature. Additionally, an increase in temperature increases the magnitude of thermal noise as well [119]:

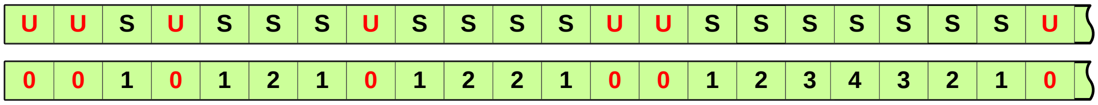
$$\sigma^2_{NOISE}(T) = \frac{2K_B T}{C}$$

which could lead to more random power-up values. In order to let both 0-skew and 1-skew cells move closer to the neutral location, we select the maximum and minimum temperatures. Moreover, low supply voltage leave a cell susceptible to noise-induced state changes, while higher voltage makes a cell stable and immune to noise [119]. Thus, for the PUF enrollment tests, high-temperature/low voltage (HTLV) and low-temperature/low voltage (LTLV) are the best condition combination for low cost enrollment tests, which can find unstable bits more efficiently. Additionally, for neutral-skewed cells, the high and low temperature conditions can shift their neutral position left and right, so the added environmental noises make them much easier to flip in a small number of tests. Changing temperature during enrollment is a much slower process than changing power supply voltage; thus using HTLV and LTLV makes the enrollment process fast and low-cost. We will evaluate the effectiveness of using HTLV and LTLV for enrollment in Section 7.4.

### 7.3.2 Neighborhood-based Bit Selection

In this section, we propose a heuristic bit selection algorithm that takes the spatial correlation we discovered in Section 7.2.4 into account. By doing so, we expect to reduce the errors due to environmental and temporal noise (aging) in the SRAM PUF. Since aging has not been covered by the above critical enrollment tests, this bit selection is quite important.

The details of our algorithm are as follows. We develop a metric to assign value (weight) to the stability of SRAM cells. Basically, we have observed that the cells that are “most stable” across environmental conditions are surrounded by more stable cells during enrollment. A stable cell surrounded by more stable cells has a tendency to become more stable because its neighboring cells are likely to experience similar aging stress and operating conditions. In our metric, we give such cells a higher weight. After determining the weight of each cell, we use a heuristic algorithm that greedily chooses cells for the PUF ID/key with weight greater than a threshold.



| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| U | U | S | U | S | S | S | U | S | S | S | S | U | U | S | S | S | S | S | S | S | U |
| 0 | 0 | 1 | 0 | 1 | 2 | 1 | 0 | 1 | 2 | 2 | 1 | 0 | 0 | 1 | 2 | 3 | 4 | 3 | 2 | 1 | 0 |

**Fig. 7.5:** Weight assignment in the bit selection algorithm.

Since we do not know the physical locations of memory cells, the heuristic bit selection algorithms have been proposed under the assumption that a cell is affected by adjacent cells only (however, we expect better results if the physical 2D structure is known, i.e. vertical and adjacent neighbors of each cell). Every word (16 bits) from each memory address is concatenated in a line, as shown in Figure 7.5. The cells have been weighted based on the adjacent cells along the line. First, all stable and unstable cells are marked as “stable” or “unstable” according to the enrollment tests, as shown in Figure 7.5. Then all unstable cells are replaced with zeroes, and they will not be selected as ID/key. Next, we focus on some stable blocks which are composed of continuous stable cells. For example, a stable block with $2p-1$ cells, the middle cell is weighted to $p$. Then, the closest two neighbors, one from each side, are weighted to $p-1$ and so on. Finally, the last two cells, from both sides are weighted to 1, because one of its neighbors is unstable bit too. The pseudo-code of the bit selection algorithm is shown as Algorithm 1. An array $W(i)$ is used to record the weight of each bit in the bit selection algorithm, and a k-bit ID/key will be generated in an array $FinalCells$. Taking the 15th to 21st bits in Figure 7.5 as another example, in a stable block with length of 7, the middle cell is weighted to 4 as that cell is surrounded by six stable cells, three on each side. The other six cells are weighted according to distance to unstable neighbor(s).

Our algorithm uses a threshold to choose the most stable bits for use as the PUF ID/key. The higher the threshold, the larger number of stable cells a cell must have to be selected for the ID/key. However, by increasing the threshold, the number of qualified bits reduces exponentially. Hence, an appropriate threshold should be decided based on key size, number of stable cells, neighborhood characteristics, etc. A lower threshold is required for a higher-sized key from the same number of stable cells.

**Algorithm 2** Neighborhood-based Bit Selection

```
procedure BitSelection ($M, T, k$)
    $M \leftarrow$ Total number of bits
    $T \leftarrow$ Threshold
    $k \leftarrow$ A k-bit ID/key needs to be generated
    for $i = 1$ to $M$ do
        if (The cell $i$ is unstable during enrollment) then
            $W(i) = 0$
        else
            Weighting the cell $i$
            if ($W(i) \geq T$) then
                The cell $i$ is a qualified bit
            end if
        end if
    end for
    $Q \leftarrow$ Sort weight of qualified bits in descending order
    $FinalCells \leftarrow Q[1:k]$ First $k$ bits are selected
    return $FinalCells[1...k]$
end procedure
```

## 7.4 Experimental Results and Discussion

### 7.4.1 Experiment Setup

In this section, our proposed approaches are evaluated on an experiment platform, the Xilinx Spartan-3 FPGA board. Its 1MB on-board SRAM is explored as the SRAM PUF. The whole platform is an intrinsic SRAM PUF system. The FPGA chip acts a microcontroller, which reads the SRAM data and sends it to a PC via UART port. A power supply and Themostream system are employed to adjust the power voltage (±10%) and temperature (0°C-80°C) respectively. The experiment setup is shown in Figure 7.6.

We collected 100 measurements of the entire SRAM memory for fresh and aged SRAMs under different environment conditions: 20 at nominal conditions, 10 at each of following conditions: low voltage (LV), high voltage (HV), low temperature (LT), high temperature (HT), and four corner conditions (LTLV, LTHV, HTLV, and HTHV). Each measurement has 1MB (8,388,608 bits) power-up values of the SRAM. Two phases of accelerated aging sessions were performed on the same SRAM. Two phases of 5-hour aging have been applied to the SRAM chip. After every burn-in, another 100 measurements were performed from the aged SRAM at the same environmental conditions mentioned above.

### 7.4.2 Effectiveness of Critical Enrollment Tests

In this section, we compare enrollment using the typical nominal conditions (case 1) with our proposed approach from Section 7.3 that uses HTLV and LTLV conditions

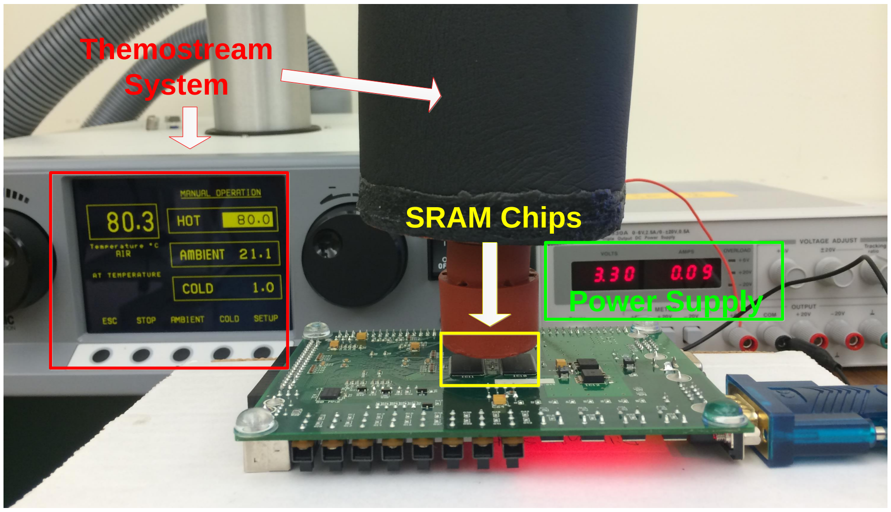
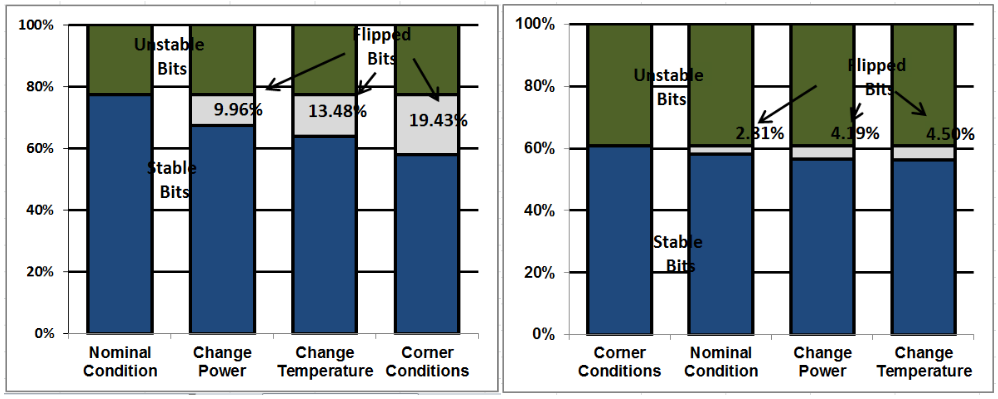


**Fig. 7.6:** Experiment setup.

(referred to as case 2). Note that case 1 would be the typical approach in prior work. In both approaches, we determine stable bits as those that do not change for all the corresponding enrollment measurements. Then we calculate how many of these bits change at the other measurement conditions (those not used during enrollment).

The results are shown in Figure 7.7 where (a) and (b) correspond to case 1 and 2 respectively. From left to right, each new condition is applied resulting in more of the original stable bits determined by enrollment flipping. In Figure 7.7(a), the original stable bits reduce by about 19.43% as the key is constructed at corner conditions. In Figure 7.7(b), our proposed approach does a better job of determining the stable bits by focusing on measurements only at critical conditions. The percentage of flipped bits is only 4.5%. By using the stable bits from enrollment using case 2, there will lesser need to correct errors in the ID/key in the reconstruction phase with lesser tests. Therefore, the above experiments illustrate that the choice of conditions used at PUF enrollment is quite significant and the proposed approach is more appropriate than the typical approach.



(a) Case 1

(b) Case 2

**Fig. 7.7:** (a) Case 1: nominal conditions and (b) Case 2: corner conditions for the enrollment tests.

### 7.4.3 Reliability of SRAM cells

In this experiment, we investigated the whole SRAM using critical condition tests for enrollment as discussed in the previous section. From the reliable bits determined by enrollment, we chose stable bits for analysis according to two approaches: (i) a naive approach that randomly selects the desired number of bits from stable bits obtained during enrollment (called “random” in this chapter); (ii) the proposed bit selection approach discussed in Section 7.3. For the proposed approach, we also varied the threshold to determine its impact. 80 measurements from various environmental conditions (nominal, HTLV, etc.) for each chip state (fresh and aged 10 hours) were used to determine which approach produced fewer bit flips during reconstruction. We calculated the percentage of bits in the 1MB that changed from enrollment in each of the 80 measurements we collected.

The distributions containing all 80 measurements are shown in Figure 7.8. Figure 7.8 (a) and (b) correspond to fresh and 10 hours aging respectively. The first, second, and third columns in Figure 7.8 compare the distribution for threshold equal to 20 to those with threshold equal to 1, 8, and 18. Note that threshold equal to 1 corresponds to the random approach. Looking at the top left plot, one can see that the bit selection approach with threshold equal 20 has more occurrences of 100% than the random approach. As we increase the threshold (more reliable neighbors required for selection as key bit), the occurrences of 100% also tend to improve. For the fresh SRAM, the bit selection algorithm with threshold of 20 has an improvement of 38% than the random selection (Threshold=1). Note that there are some outliers (low values of percentage) for threshold at 20, but these would probably be less significant if we gathered more measurements. The trends clearly show that there is better stability with bit selection and higher threshold values.

Examining the impact of aging (Figure 7.8 (b)), we see that the stability of the keys are lower than the fresh chips. This is because the enrollment conditions do not contain aging measurements. Without these measurements, we cannot determine which bits will flip as the chip ages. Comparing the two algorithms (leftmost column of Figure 7.8), we can see that the percentage of stable measurements at or near 100% is still more for the proposed bit selection algorithm than the random approach. The same trend of improvement as we increase the threshold also still holds for the aged chips. Although the percentage of correct bit reconstruction goes down due to the aging effects, the improvement of bit selection algorithm (at threshold equal 20) compared to the random approach increases to 45%. To some degree, one also can say that the bit selection algorithms compensate well for the lack of aging measurements used in enrollment. This is important because it removes the need for burn-in tests which would have significant costs and device reliability issues for high volume production.

### 7.4.4 Reliability and Uniqueness of PUF Key

In these experiments, we evaluate the reliability and uniqueness of keys generated by the proposed bit selection algorithm. We divided the whole 1MB SRAM into 8 PUFs. Each PUF has 128KB and used to generate a 128-bit key. This 128-bit key can uniquely identify $3.4e38$ devices, so it can apply to most current applications in the market. 20 critical condition tests (HTLV and LTLV) and 20 nominal condition tests were used for enrollment. Another 80 measurements from various environmental conditions for each chip state (fresh and aged 10 hours) were used to reconstruct the key. The proposed bit selection approach and the random bit selection approach that was used in the previous section are investigated and compared in this section.

#### Bit Error

To analyze the impact on error correction code (ECC) overheads, we also look at the maximum number of bit flips in each of the 80 measurements for fresh and aged chips for random and neighborhood-based bit selection (threshold equal 18). The number of bits that flip will determine the number of bits we need to correct via ECC and thus is

**Table 7.4:** Bit flips in the two bit selection methods.

| **Reconstruction** | | **Fresh** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Enroll** | **Error Bits** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **NC** | **Random Selection** | 338 | 154 | 78 | 39 | 25 | 9 | 5 | 1 | 0 | 0 |
| **CC** | **Random Selection** | 550 | 82 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **Proposed Selection** | 559 | 72 | 7 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Reconstruction** | | **10 hours burn-in** | | | | | | | | | |
| **Enroll** | **Error Bits** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **NC** | **Random Selection** | 118 | 214 | 148 | 91 | 39 | 13 | 11 | 5 | 1 | 2 |
| **CC** | **Random Selection** | 347 | 198 | 63 | 24 | 6 | 0 | 1 | 1 | 0 | 0 |
| | **Proposed Selection** | 438 | 162 | 34 | 5 | 1 | 0 | 0 | 0 | 0 | 0 |

NC: Nominal condition enrollment tests; CC: Critical condition enrollment tests;

directly related to the ECC overheads. The results are shown in Table 7.4. Note that for random selection, we consider two different enrollment conditions: nominal conditions (NC) and corner conditions (CC).

Table 7.4 clearly shows that critical enrollment tests (CC) is more effective than nominal condition tests (NC) for selecting stable bits in the enrollment phase. For the fresh chip, NC case has up to 7 simultaneous bit errors. For critical enrollment tests, the maximum number of error bits is 3 for both algorithms so there is improvement in ECC overhead. For fresh chips, we also observe that the average number of errors is also similar between random selection (CC case) and the bit selection algorithm. As the chips age, however, the maximum number of error bits increases at a faster rate for the random selection (CC case) compared to the bit selection algorithm. For 10 hours burn-in, the maximum number of error bits is 9 for the random approach (NC case), 7 for random approach (CC case) but only 4 for bit selection approach. There is about a 22% and 55% decrease in bit flips for the proposed approaches compared to the typical approach (random selection at NC). These improvements represent a nontrivial decrease in overhead since the cost of ECC increases significantly with each bit error.

#### Uniqueness

The bit selection algorithm would be less useful if it resulted in lower uniqueness of keys/IDs in the population. In this section, we calculated the PUF uniqueness for the random (CC case) and bit selection techniques based on the Hamming distance (HD) between the IDs of different PUFs. For the bit selection, different thresholds are investigated as well (specifically 5, 8, and 12). The occurrence rate of each of 2016 ($\binom{64}{2}$) HDs across the 128 PUF outputs is shown in Table 7.5. Table 7.5 contains the mean and variance of the HD for each approach (note that the HD distributions were Gaussian). The average inter-die HD using random bit selection is about 63.78 bits out of 128 or 49.8%. The average inter-die HD of different thresholds are 63.99 (49.99%), 64.03 (50.02%), 63.81(49.85%), which are closer to ideal value 64 out of 128 or 50% than the random bit selection. We conclude that the uniqueness of IDs/keys from our bit selection algorithms is at least as good as the random approach, if not better.

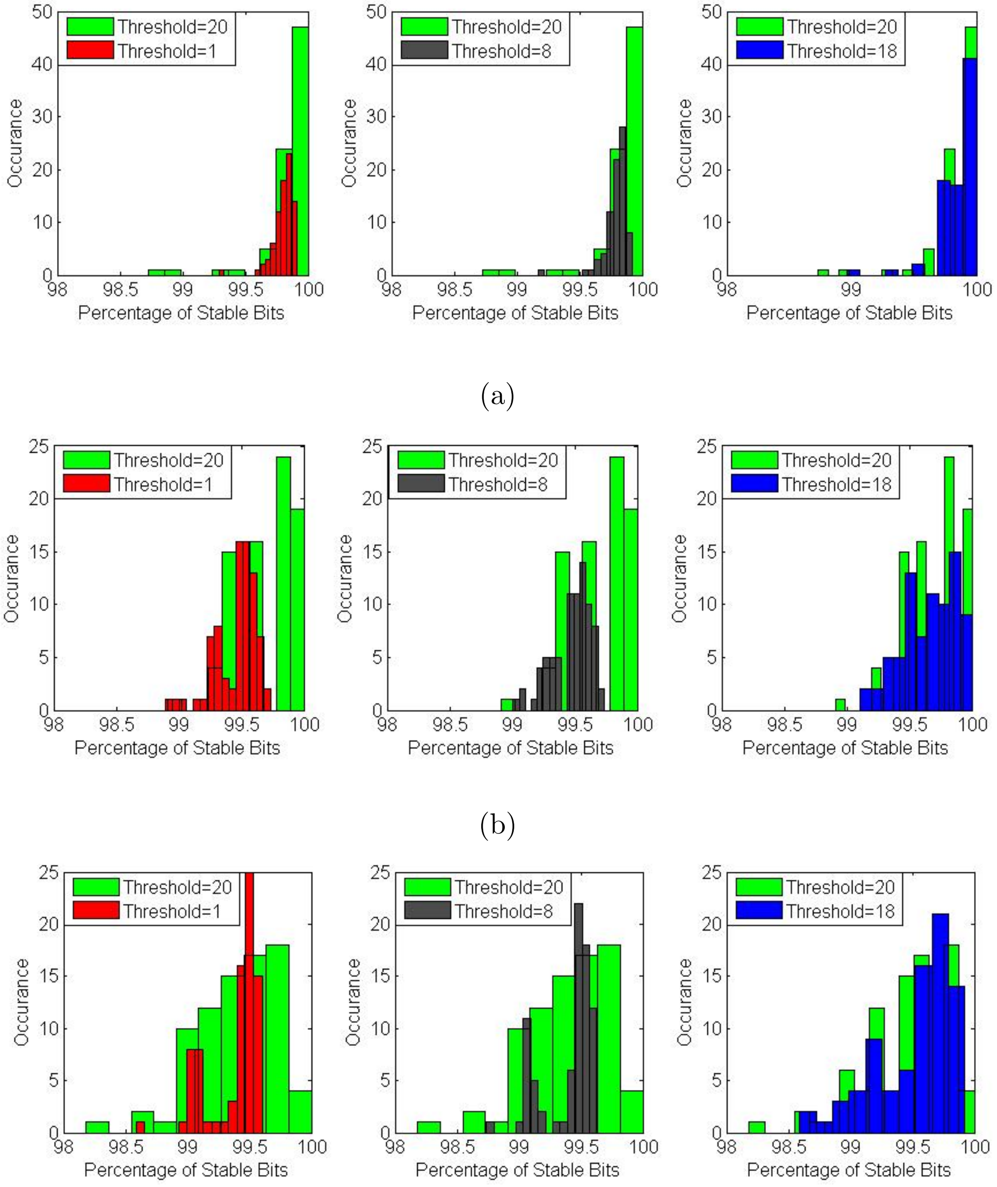
### 7.5 Conclusion

In this chapter, we have presented our discovery that critical conditions for enrollment tests can more efficiently identify unreliable bits. We have also developed an effective bit

**Table 7.5:** HD in 128-bit outputs from 64 PUFs.

|  | **Random** | **Neigborhood-based Bit Selection** |  |  |
|---|---|---|---|---|
|  |  | T=5 | T=8 | T=12 |
| **Mean** | 63.78 | 63.99 | 64.03 | 63.81 |
| **Variance** | 31.96 | 31.35 | 31.94 | 32.07 |

selection algorithm that can assist with key selection (i.e., determine "most stable bits) from the reliable bits obtained from enrollment tests. Our experiments demonstrate the effectiveness of our approaches to reduce ECC overhead with low test cost, thereby making SRAM PUFs more suitable for high-volume production. One limitation in the current paper was that without knowledge of the full physical SRAM structure, we could not develop a neighborhood-based metric/algorithm that includes 2D information. We shall investigate this in future work.



(a)

(b)

(c)

**Fig. 7.8:** Distributions of percentage of correct bits using different thresholds for SRAM (a) without aging, (b) with 5 hours aging, and (c) with 10 hours aging.

# Chapter 8

# Conclusions

Today's integrated circuit (IC) development demands a large capital investment. Many third-parties are involved in IC design and manufacturing process, in order to reduce costs. Therefore, the semiconductor supply chain becomes more vulnerable to a wide range of attacks than ever before. To improve security and trustworthiness of ICs, we presented a series of design and test methodologies to deal with four challenging hardware security problems. The major contributions of the thesis will be presented in this chapter.

## 8.1 Hardware Trojan Detection and Prevention

Hardware Trojan is one of the biggest concerns since they can be inserted into a circuit at any stage of development and manufacturing, and could be very difficult (if not impossible) to detect. Hardware Trojans can be inserted at any phase from circuit design to fabrication. However, nowadays since almost all design houses have become "fabless" – their parts are fabricated in third-party foundries offshore to lower costs – one of the best opportunities to introduce malicious inclusions and make alterations occurs at the fabrication stage of the supply chain. Therefore, we focus on hardware

Trojans injected at fabrication stage in this thesis. A post-silicon test method (chapter 2) and a design-for-trust approach (chapter 5) have been developed to effectively detect Trojan-inserted ICs and prevent Trojan insertion, respectively.

### 8.1.1 Post Silicon Test Method for Trojan Detection

Chapter 2 proposes a delay-based hardware Trojan detection methodology. A clock sweeping technique is used to obtain path delay information without any additional hardware. Conventional transition and path delay fault (TDF and PDF) patterns can be used to achieve high coverage on the nodes of critical and non-critical paths. Once the data has been collected by clock sweeping, we generate a series of delay signatures for ICs, and analyze whether ICs contain Trojans or not. Statistical analysis, such as MDS, is used to distinguish the effects of hardware Trojans from process variations. In addition, the proposed post-silicon detection technique has been evaluated in simulations with s38417 benchmark circuit and FPGAs. Different types and sizes of hardware Trojans were inserted. Both simulation and silicon results demonstrate that even in the presence of uncontrollable process variations, measurement noise, and environment variation, Trojan-inserted ICs could still be identified using our proposed technique.

### 8.1.2 Design-for-Trust Method for Trojan Prevention

Chapter 5 proposes the first design-for-trust technique that prevents the insertion of additional Trojan gates in design layout and mask. The proposed BISA technique is able to fill all unused spaces with functional standard cells instead of non-functional filler cells during layout design. The inserted standard cells are then connected to form a circuitry called built-in self-authentication (BISA) circuit, which is independent of the original circuit. The BISA structure can be used to test the functionality of its inserted cells. If any of the inserted cells are replaced or otherwise modified, the BISA test procedure will be able to detect it. The main advantage of BISA is that it has no golden chip requirement. In contrast, most detection approaches need golden chips either fabricated by a trusted foundry or verified to be Trojan-free through reverse-engineering, both of which are prohibitively expensive. Since BISA relies on logic testing, process variation is not a factor either, as compared to Trojan detection techniques based on side-channel analysis. As an additional advantage, BISA has a negligible impact on original design in terms of area and power.

In Chapter 5, we also present a new procedure that automatically inserts the entire BISA circuitry (including linear feedback shift register (LFSR), multiple-input signature register (MISR) and BISA cells) and can guarantee 100% test coverage, even for SOCs with large unused spaces. We propose a new filling algorithm based on dynamic programming that can produce an optimal solution. Results show that it fills unused spaces more efficiently and obtains higher test coverage than the greedy algorithm in [75]. To demonstrate the feasibility of BISA, the proposed technique has been implemented on 15 benchmarks from different benchmark suites including SOCs and single-module designs. We demonstrate that the proposed BISA structure is compatible to any hierarchical design. In addition, we analyze potential impacts introduced by BISA. We compare timing (critical path delays) for designs with and without BISA. We also analyze decoupling capacitance in BISA cells and regular filler cells from two standard cell libraries. We find that the added BISA circuits have very small impact on the timing and decoupling capacitors of the original design without BISA. Ten different attacks were performed on BISA circuitry and the results show that BISA can detect every one of them with a high level of confidence.

### 8.2 Recycled IC Detection

In this thesis, we discussed the recycled IC problem and proposed a path-delay fingerprinting flow to detect recycled ICs. Because of silicon aging, the path delay would be larger and thus change its original path-delay fingerprint. With no additional hardware circuitry required, the post-silicon test method provides no overhead in term of area and power consumption. The simulation results of different benchmarks with a variety of workload and operating environment demonstrated the effectiveness of the proposed technique.

### 8.3 Design Obfuscation for Anti-Reverse Engineering

The Chapter 6 proposes a new design methodology that can effectively prevent reverse engineering of the chip functionality and further prevent hardware Trojan insertion with split fabrication process. Our technique allows FEOL and BEOL to be separated at higher layers ($\geq$ M4) to reduce cost. In order to enhance effectiveness of obfuscation, all unused spaces in layout will be filled with additional functional cells or circuitry called obfuscated built-in self-authentication (OBISA) instead of non-functional filler cells during layout design. If any of the OBISA cells are replaced by a Trojan cell during

FEOL, OBISA will be able to detect the modification. We propose to make connections between OBISA added into unused spaces and the original circuit, especially in its critical parts that are to be protected. The OBISA circuit not only makes it extremely difficult for adversary to identify the original design, but also thwarts hardware Trojan insertion by filling unused spaces in layout. In addition, several optimization approaches are proposed to minimize timing and power overhead introduced by OBISA circuit while maintaining a high test coverage for OBISA.

## 8.4 Reliable On-Chip ID Generation

Secure and reliable on-chip ID/key generation becomes crucial for Internet of Things (IoT) devices. Physical unclonable function is an emerging security block for generating volatile secret ID/key for different applications in authentication, identification, counterfeit detection and cryptographic. In this thesis, a post-silicon test technique and a bit selection method for SRAM-PUF are proposed to generate more reliable on-chip ID/key.

### 8.4.1 Post-Silicon Test Method for ID Generation

We will use our proposed clock sweeping technique to uniquely identify ICs. We develop a methodology to use the path delay information to create a unique binary identifier for each IC. In addition, we will analyze our ability to accurately identify ICs under measurement and environmental noise. Simulation results from 90nm technology and experimental results from 90nm FPGAs demonstrate the effectiveness of our technique.

### 8.4.2 Security Primitive Design for ID Generation

A current challenge in SRAM PUFs is their sensitivity to temperature and voltage variations as well as aging. Chapter 7 has identified that neighboring SRAM cells can used to determine the more reliable SRAM PUF cells. Based on this observation, we found the most critical enrollment conditions and developed a low-cost bit selection algorithm for identifying the most reliable bits which is suitable for massive production of SRAM PUFs. The experiment results from SRAM chips show that the SRAM-PUF reliability improves dramtically compared to random bit selection.

## 8.5 Summary of Conclusions

This thesis is devoted to developing a series of design and test methodologies to tackle four major malicious activities, including hardware Trojan insertion, recycled IC counterfeiting, reverse engineering, and on-chip ID/key stolen, in order to improve the security and trustworthiness of ICs. Three post-silicon test methodologies, two design-for-security methodologies, and one hardware security primitive enhancement technique are proposed to target these four security problems in this thesis. However, there are still future research directions to improve the security and trustworthiness of ICs. Researchers can bring IC security to the next level by developing new solutions.

# Bibliography

[1] S. Adee,“The hunt for the kill switch,” IEEE Spectrum, http://spectrum.ieee.org/semiconductors/design/the-hunt-for-the-kill-switch

[2] S. Skorobogatov and C. Woods, “Breakthrough silicon scanning discovers backdoor in military chip,”

[3] DIGITIMES Research, “Trends in the global IC design service market.” [Online]. Available: http://www.digitimes.com/news/a20120313RS400.html?chid=2.

[4] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, “Trustworthy Hardware: Identifying and Classifying Hardware Trojans,” IEEE Computer Mag., vol. 43, pp. 39-46, 2010.

[5] M. Tehranipoor and C. Wang, “Introduction to Hardware Security and Trust,” Springer, Augest 2011.

[6] Y. Jin and Y. Makris, “Hardware Trojan Detection Using Path Delay Fringerprint,” IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2008.

[7] C. Dunbar and G. Qu, “Designing Trusted Embedded Systems from Finite State Machines,” ACM Transactions on Embedded Computing Systems (TECS), Vol. 13, Issue 5s, Nov. 2014.

[8] Y. Shiyanovskii, et al, “Process Reliability based Trojans Through NBTI and HCI Effects,” NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 215-222, June 2010.

[9] X. Zhang, et al, “A Study on the effectiveness of Trojan detection techniques using a red team blue team approach,” 2013 IEEE 31st VLSI Test Symposium (VTS), 2013.

[10] L. Lin, W. Burleson, and C. Paar, “MOLES: Malicious Off-chip Leakage Enabled by Side-channels,” IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), pp. 117-122, Nov. 2009.

[11] B. Cha and S. Gupta, “A Resizing Method to Minimize Effects of Hardware Trojans,” 2014 IEEE 23rd Asian Test Symposium (ATS), 2014.

[12] N. Tsoutsos and M. Maniatakos, “Fabrication Attacks: Zero-overhead malicious modifications enabling modern microprocessor privilege,” Emerging Topics in Computing, IEEE Transactions on, Vol. 2 , Issue 1, pp. 81-93, 2013.

[13] J. Stradley and D. Karraker, “The electronic part supply chain and risks of counterfeit parts in defense applications,” Components and Packaging Technologies, IEEE Transactions on, vol. 29, pp. 703-705, sept. 2006.

[14] J. Cassell, “Reports of Counterfeit Parts Quadruple Since 2009,” Challenging US Defence Industry and National Security.

[15] U. Guin, D. DiMase, and M. Tehranipoor, “A Comprehensive Framework for Counterfeit Defect Coverage Analysis and Detection Assessment,” Journal of Electronic Testing: Theory and Applications (JETTA), 2014.

[16] X. Zhang and M. Tehranipoor, ”Design of On-chip Light-weight Sensors for Effective Detection of Recycled ICs,,” IEEE Transactions on Very Large Scale Integrated Circuit (TVLSI), 2013.

[17] Bureau of Industry and Security, U.S. Department of Commerce, “Defense Industrial base Assessment: Counterfeit Electronics,” Jan. 2010.

[18] U.Guin, M.Tehranipoor, D.DiMase, and M.Megrdichian, “Counterfeit IC Detection and Challenges Ahead,” *ACM/SIGDA E-NEWSLETTER*, vol.43, no.3, March 2013.

[19] U.Guin, D.DiMase, and M.Tehranipoor, “Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead,” *Journal of Electronic Testing*, vol.30, no.1, pp. 9-23, 2014.

[20] U.Guin, D.DiMase, and M.Tehranipoor, “A comprehensive framework for counterfeit defect coverage analysis and detection assessment,” *Journal of Electronic Testing*, vol.30, no.1, pp. 25-40, 2014.

[21] U.Guin, K.Huang, D.DiMase, J.Carulli, M.Tehranipoor, and Y.Makris, “Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain,” *Proceedings of the IEEE*, vol. 102, no.8, pp. 1207-1228, Aug 2014.

[22] U.Guin, D.Forte, and M.Tehranipoor, “Anti-Counterfeit Techniques: From Design to Resign,” in *Microprocessor Test and Verification (MTV)*, 2013.

[23] M.M. Tehranipoor, U.Guin, and D.Forte, *Counterfeit Integrated Circuits: Detection and Avoidance.* Springer, 2015.

[24] X.Zhang, N.Tuzzio, and M.Tehranipoor, “Identification of recovered ics using fingerprints from a light-weight on-chip sensor,” in *Proc. IEEE-ACM Design Automation Conference*, June 2012, pp. 703-708.

[25] J. Smith, "Non-destructive State Machine Reverse Engineering," 2013 6th International Symposium on Resilient Control Systems (ISRCS), pp. 120-124, Aug., 2013.

[26] P. Subramanyan, et al., "Reverse Engineering Digital Circuits Using Structural and Functional Analyses," IEEE Transactions on Emerging Topics in Computing, Volume 2, Issue 1, pp. 63-80, Dec., 2013.

[27] U. Rurmair *et al*, "Modeling Attacks on Physical Unclonable Functions," the 17th ACM conference on Computer and communications security, October 04-08, 2010.

[28] G. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," ACM/IEEE Design Automation (DAC), pp. 9-14, June 2007.

[29] N. Tuzzio, K. Xiao, X. Zhang, and M. Tehranipoor, "A Zero-Overhead IC Identification Technique using Clock Sweeping and Path Delay Analysis," IEEE GLSVLSI, 2012.

[30] B. Gassend *et al*, "Silicon Physical Random Functions," in Proceedings of the 9th ACM conference on Computer and communications security, pp. 148-160, 2002.

[31] J. Lee, D. Lim, B. Gassend, G. Suh, M. van Dijk, S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication application," In Proceedings of the Symposium on VLSI Circuits, pp. 176-179, 2004.

[32] J. Guajardo, S. Kumar, G. Schrijen, P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," Cryptographic Hardware and Embedded Systems Workshop, LNCS, vol. 4727, pp. 63-80, 2007.

[33] C. Bao, D. Forte, and A. Srivastava, "On Application of One-Class SVM to Reverse Engineering-based Hardware Trojan Detection," 15th International Symposium on Quality Electronic Design (ISQED), pp. 47-54, March 2014.

[34] S. Bhunia, M. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan Attacks Threat Analysis and Countermeasures," Proceedings of the IEEE, Vol. 102, Issue 8, pp. 1229-1247, 2014.

[35] M. Banga and M. Hsiao, "A Novel Sustained Vector Technique for the Detection of Hardware Trojans," International Conference on VLSI Design, pp. 327-332, Jan. 2009.

[36] R. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "MERO: A statistical approach for hardware Trojan detection," Workshop on Cryptographic Hardware and Embedded Systems, pp. 396-410, 2009.

[37] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting Malicious Inclusions in Secure Hareware: Challenges and Solutions," IEEE Int. Symp. Hardware-Oriented Security and Trust (HOST), pp. 15-19, 2008.

[38] K. Xiao, X. Zhang, and M. Tehranipoor, "A Clock Sweeping Technique for Detecting Hardware Trojans Impacting Circuits Delay," IEEE Design & Test, Vol. 30, Issue 2, pp. 26-34. April, 2013.

[39] D. Agrawal, S. Baktir, and D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan Detection using IC Fingerprinting," IEEE Symp. on Security and Privacy (SP), pp. 296-310, 2007.

[40] J.Aarestad, D. Acharyya, R. Rad, and J. Plusquellic, "Detecting Trojans through Leakage Current Analysis Multiple Supply Pad IDDQs," IEEE Transactions on Information Forensics and Security, vol. 5, issue 4, pp. 893-904, 2010.

[41] D. Forte, et al., "Temperature tracking An innovative run-time approach for hardware Trojan detection," International Conference On Computer Aided Design (ICCAD), 2013.

[42] F. Stellari, et al., "Verification of Untrusted Chips Using Trusted Layout and Emission Measurements," 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 19-24, 2014.

[43] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware Trojans in third-party digital IP cores," 2011 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 67-70, 2011.

[44] M. Hicks, et al., "Overcoming an Untrusted Computing Base Detecting and Removing Malicious Hardware Automatically," 2010 IEEE Symposium on Security and Privacy (SP), pp. 159-172, 2010.

[45] H. Salmani and M. Tehranipoor, "Analyzing Circuit Vulnerability to Hardware Trojan Insertion At the Behavioral Level," 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 190-195, 2013.

[46] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: identification of stealthy malicious logic using boolean functional analysis," in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 697-708, 2013.

[47] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, "A Score-Based Classification Method for Identifying Hardware-Trojans At Gate-Level Netlists," in Proceedings of the 2015 Design, Automation & Test in Europe Conference (DATE), pp. 465-470, 2015.

[48] M. Rathmair, F. Schupfer, and C. Krieg, "Applied Formal Methods for Hardware Trojan Detection," 2014 IEEE International Symposium on Circuits and Systems (ISCAS), 2014.

[49] E. Love, Y. Jin, and Y. Makris, "Enhancing Security via Provably Trustworthy Hardware Intellectual Property," 2011 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 12-17, 2011.

[50] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time," IEEE Transactions on VLSI, 2011.

[51] B. Zhou, et al., "A Low Cost Acceleration Method for Hardware Trojan Detection based on Fan-out Cone Analysis," Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis, 2014.

[52] H. Salmani and M. Tehranipoor, "Layout-Aware Switching Activity Localization to Enhance Hardware Trojan Detection," IEEE Transactions on Information Forensics and Security (TIFS), Vol. 7, Issue. 3, pp. 76-87, 2012.

[53] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri, "Design and Analysis of Ring Oscillator based Design-for-Trust Technique," VLSI Test Symposium (VTS), pp. 105-110, 2011.

[54] J. Li and J. Lach, "At-Speed Delay Characterization for IC Authentication and Trojan Horse Detection," IEEE Int. Symposium on Hardware-Oriented Security and Trust (HOST), pp. 8-14, 2008.

[55] A. Ramdas, S. Saeed, and O. Sinanoglu, "Slack Removal for Enhanced Reliability and Trust," Design & Technology of Integrated Systems In Nanoscale Era (DTIS), 2014 9th IEEE International Conference On, pp. 1-4, 2014.

[56] X. Zhang and M. Tehranipoor, "RON: An on-chip ring oscillator network for hardware Trojan detection," DATE, pp. 1-6, 2011.

[57] S. Narasimhan, et al., "Improving IC Security Against Trojan Attacks Through Integration of Security Monitors,"IEEE Design & Test of Computers, Vol. 29, Issue 5, pp. 37-60, 2012.

[58] Y. Cao, C. Chang, and S. Chen, "Cluster-based Distributed Active Current Timer for Hardware Trojan Detection," 2013 ISCAS.

[59] B. Cha and S. Gupta, "Efficient Trojan Detection via Calibration of Process Variations," IEEE 21st Asian Test Symposium (ATS), pp. 355-361, 2012.

[60] Y. Liu, K. Huang, and Y. Makris, "Hardware Trojan Detection Through Golden Chip-Free Statistical Side Channel Fingerprinting," Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE, 2014.

[61] G. Bloom, B. Narahari, and R. Simha "OS Support for Detecting Trojan Circuit Attacks," IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), 2009.

[62] J. Dubeuf, D. Hely, and R. Karri, "Run-time detection of hardware Trojans: The Processor Protection Unit," 2013 18th IEEE European Test Symposium (ETS), 2013.

[63] Y. Jin and D. Sullivan, “Real-time trust evaluation in integrated circuits,” DATE, 2014.

[64] Y.Jin, D. Maliuk, and Y. Makris, “Post-deployment trust evaluation in wireless cryptographic IC,” 2012 Design, Automation & Test in Europe Conference (DATE), pp. 965-970, 2012.

[65] J. Roy, F. Koushanfar, and I Markov, “EPIC: Ending Piracy of Integrated Circuits,” Design, Automation and Test in Europe (DATE), 2008.

[66] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, “Security Analysis of Logic Obfuscation,” 2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 83-89, 2012.

[67] R. Chakraborty and S. Bhunia, “Security against Hardware Trojan through a Novel Application of Design Obfuscation,” IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), pp. 113-116, 2009.

[68] R. Chakraborty and S. Bhunia, “HARPOON: An Obufscation-Based SoC Design Methodology for Hardware Protection,” IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, Vol. 28, No. 10, pp. 1493-1502, Oct, 2009.

[69] A. Baumgarten, A. Tyagi, and J. Zambreno, “Preventing IC Piracy Using Reconfigurable Logic Barriers,” IEEE Design & Test of Computers, Vol. 27, Issue 1, pp. 66-75, 2010.

[70] B. Liu and B. Wang, “Embedded Reconfigurable Logic for ASIC Design Obfuscation Against Supply Chain Attacks,” Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014.

[71] J. Wendt and M. Potkonjak, “Hardware Obfuscation Using PUF-based Logic,” 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 270-271, 2014.

[72] R. Cocchi, P Baukus, L. Chow, and B. Wang, “Circuit Camouflage Integration for Hardware IP Protection,” 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), 2014.

[73] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, “Security Analysis of Integrated Circuit Camouflaging,” Proceedings of the 2013 ACM SIGSAC conference on Computer & communications, pp. 709-720, 2013.

[74] Y. Bi, et al., “Leveraging Emerging Technology for Hardware Security-case study on silicon nanowire FETs and Graphene SymFETs,” 2014 IEEE 23rd Asian Test Symposium (ATS), pp. 342-347, 2014.

[75] K. Xiao and M. Tehranipoor, “BISA: Built-In Self-Authentication for Hardware Trojan Prevention,” IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2013.

[76] K.Xiao, D. Forte, and M. Tehranipoor, “A Novel Built-In Self-Authentication Technique to Prevent Inserting Hardware Trojans,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 33, Issue 12, pp. 1778-1791, Dec, 2014.

[77] K. Xiao and M. Tehranipoor, “Methods And Systems For Preventing Hardware Trojan Insertion,” US Patent App. 14/204,656, 2014.

[78] D. Mclntyre, et al., “Trustworthy Computing in a Multi-Core System Using Distributed Scheduling,” 2010 IEEE 16th Int. On-Line Testing Symposium (IOLTS), pp. 211-213, 2010.

[79] C. Liu, et al., “Shielding Heterogeneous MPSoCs from Untrustworthy 3PIPs through Security Driven Task Scheduling,” IEEE Transactions on Emerging Topics in Computing, Vol. 2, Issue 4, pp. 461-472, 2014.

[80] O. Keren, I. Levin, and M. Karpovsky, “Duplication Based One-to-Many Coding for Trojan HW Detection,” 2010 IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), 2010.

[81] J. Rajendran, et al., “High Level Synthesis for Security and Trust,” 2013 IEEE 19th International On-Line Testing Symposium (IOLTS), pp. 232-233, 2013.

[82] U. Guin, D. Forte, and M. Tehranipoor, “Low-Cost On-Chip Structures for Combating Die and IC Recycling,” Design Automation Conference (DAC), 2014.

[83] K. Huang, J. Carulli, and Y. Makris, “Parametric counterfeit IC detection via Support Vector Machines,” In Proc. International Symposium on Fault and Defect Tolerance in VLSI Systems (DFT), pp. 7-12, 2012.

[84] K. Xiao, D. Forte, and M. Tehranipoor, “Circuit Timing Signature (CTS) for Detection of Counterfeit Integrated Circuits,” Secure System Design and Trustable Computing, pp. 211-239, Spinger, 2016.

[85] G. Xu and A. Singh, “Achieving high transition delay fault coverage with partial DTSFF scan chains,” IEEE Int. Test Conf. (ITC), pp.1-9, 2007.

[86] I.Borg and P.Groenen, “Modern Multidimensional Scaling, Theory and Applications,” New York, Springer-Verlag, 1997.

[87] X. Zhang, K. Xiao, and M. Tehranipoor, “Path-delay fingerprinting for identification of recovered ICs,” IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 13-18, 2013.

[88] Y. Zheng, X. Wang, and S. Bhunia, “SACCI: Scan-Based Characterization Through Clock Phase Sweep for Counterfeit Chip Detection,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems (TVLSI), Volume 23, Issue 5, pp. 831-841, 2014.

[89] N. Kimizuka, T. Yamamoto, T. Mogami, K. Yamaguchi, K. Imai, and T. Horiuchi, "The impact of bias temperature instability for direct-tunneling ultra-thin gate oxide on mosfet scaling, In VLSI Technology, 1999.

[90] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of pmos nbti effect for robust nanometer design," Proc. of the 43rd Annual Conference on Design Automation (DAC06), pp. 10471052, 2006.

[91] W. Jiang, H. Le, J. Chung, T. Kopley, P. Marcoux, and C. Dai, "Assessing circuit-level hot-carrier reliability," IEEE International Reliability Physics Symposium Proceedings, pp. 173179, 1998.

[92] L. Wu, et. al., "Glacier: a hot carrier gate level circuit characterization and simulation system for vlsi design," In Proceedings of IEEE International Symposium on Quality Electronic Design (ISQED), 73 79, 2000.

[93] http://www.nangate.com/.

[94] J. Rabaey, A. Chandrakasan, and B. Nikolic, "Digital Integrated Circuits: A Design Perspective", Second Edition, Prentice Hall, 2003.

[95] I. T. Jolliffe, "Principal Component Analysis (2nd Edition)," Springer, pp. 150-165, 2002.

[96] G. Becker, F. Regazzoni, C. Paar, and W. Burleson, Stealthy Dopant-Level Hardware Trojans, in Cryptographic Hardware and Embedded Systems - CHES 2013, vol. 8086, pp. 197214. 2013.

[97] X Zhang, K Xiao, M Tehranipoor, J Rajendran, and R Karri, "A study on the effectiveness of trojan detection techniques using a red team blue team approach," IEEE 31st VLSI Test Symposium (VTS), 2013.

[98] R. Frohwerk, "Signature Analysis: A New Digital Field Service Method," Hewlett-Packard Journal, vol. 28, no. 9, pp. 2-8, May 1977.

[99] M. Bushnell and V. Agrawal, "Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits," Springer, 2000.

[100] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, "Introduction to Algorithms," The MIT Press (third edition), 2009.

[101] H. Chen, C. Chang, and T. Hwang, "Reconfigurable ECO Cells for Timing Closure and IR Drop Minimization," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 18, Issue 12, pp. 1686-1695, Dec., 2010.

[102] H. Xiang, K. Chao, and M. Wong, "An ECO Routing Algorithm for Eliminating Coupling-Capacitance Violations," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, issue 9, Spet. 2006.

[103] http://eda-automation.blogspot.com/2010/06/filler-cells.html

[104] C. Yeh and M. Marek-Sadowska, "Timing-aware power noise reduction in placement," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, Issue 3, pp. 527-541, 2007.

[105] N. Weste and D. Harris, "CMOS VLSI Design: A Circuits and Systems Perspective," 4th edition, Addison-Wesley, 2010.

[106] http://www.oracle.com/technetwork/systems/opensparc/index.html.

[107] R. Jarvis and M. McIntyre, "Split Manufacturing Method for Advanced Semiconductor Circuits," US Patent 10/305,670, 2007.

[108] J. Valamehr, et al, "A 3-D Split Manufacturing Approach to Trustworthy System Development," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 32, No. 4, April, 2013.

[109] K. Vaidyanathan, et al, "Building Trusted ICs using Split Fabrication," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2014.

[110] K. Vaidyanathan, et al, "Efficient and Secure Intellectual Property (IP) Design with Split Fabrication," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2014.

[111] M. Jagasivamani, et al, "Split-Fabrication Obfuscation: Metrics and Techniques," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2014.

[112] B. Hill, et al, "A Split-Foundry Asynchronous FPGA," Proceedings of the IEEE Custom Integrated Circuits Conference (CICC), September, 2013.

[113] F. Imeson, et al, "Securing Computer Hardware Using 3D integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation," in the Proceedings of the 22nd USENIX Security Symposium, Aug., 2013.

[114] J. Rahendran, et al. "Is Split Manufacturing Secure?" IEEE Design, Automation & Test in Europe, 2013.

[115] R. Jarvis and M. McIntyre, "Split Manufacturing Method for Advanced Semicondunctor Circuits," US Patent 7195931, March 2007.

[116] K. Xiao, D. Forte, and M. Tehranipoor, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 14-19, 2015.

[117] Chipworks Inc. http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/intels-14-nm-parts-are-finally-here/

[118] R. Chakraborty and S. Bhunia, "HARPOON: An Obufscation-Based SoC Design Methodology for Hardware Protection," IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, Vol. 28, No. 10, pp. 1493-1502, Oct, 2009.

[119] D. Holcomb, W. Burleson and K. Fu, "Power-Up SRAM State as An Identifying Fingerprint and Source of True Random Numbers," IEEE Transactions on Computer, 2009.

[120] M. Cortez *et al*, "Adapting Voltage Ramp-Up Time for Temperature Noise Reduction on Memory-based PUFs," IEEE intl. Symposium on Hardware-Oriented Security Trust (HOST), 2013.

[121] M. Bhargava, C. Cakir, and K. Mai, "Reliability Enhancement of Bi-Stable PUFs in 65nm Bulk CMOS," IEEE intl. Symposium on Hardware-Oriented Security Trust (HOST), 2012.

[122] Y. Zheng, S. Hashemian and S. Bhunia, "RESP: A Robust Physical Unclonable Function Retrofitted into Embedded SRAM Array," 50th ACM/IEEE Design Automation Conference (DAC), 2013.

[123] M. Yu, and S. Devadas, "Secure and Robust Error Correction for Physical Unclonable Functions," IEEE Design & Test of Computers, vol. 27, no.1, pp. 48-65, 2010.

[124] M. Cortez *et al*, "Modeling SRAM Start-Up Behavior for Physical Unclonable Functions," IEEE Symp. Defect and Fault Tolerance in VLSI (DFT), 2012.

[125] S. Sabadc and D. Walker, "Improved Wafer-Level Spatial Analysis for IDDQ Limit Setting," Proceeding of International Test Conf. (ITC), pp. 82-91, 2001.

[126] F. Tehranipoor, N. Karimina, K. Xiao, and J. Chandy, "DRAM based Intrinsic Physical Unclonable Functions for System Level Security," in Proceedings of the 25th edition on Great Lakes Symposium on VLSI, 2015.

[127] W. Daasch and K. Cota, "Variance Reduction Using Wafer Patterns in IDDQ Data," Proceeding of International Test Conf. (ITC), pp. 189-198, 2000.

[128] J. Lee and D. Walker "IC Performance Prediction for Test Cost Reduction," IEEE Int. Symp. on Semiconductor Manufacturing Conference, pp. 111-114, 1999.

[129] M.T. Rahman, K. Xiao, D. Forte, X. Zhang, J. Shi, and M. Tehranipoor, "Ti-TRNG: Technology independent true random number generator," in Proceedings of the 51st Annual Design Automation Conference (DAC), pp. 1-6, 2014.

[130] Y. Zheng, A. Krishna, and S. Bhunia, "ScanPUF: Robust ultralow-overhead PUF using scan chain," 18th Asia and South Pacific Design Automation Conference (ASP-DAC), 2013.

[131] K. Xiao *et al.*, "Bit Selection Algorithm Suitable for High Volume Production of SRAM PUF," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2014.

[132] M. Cortez et al, “Modeling SRAM start-up behavior for physical unclonable functions,” IEEE international symposium on Defect and fault tolerance in VLSI and nanotechnology systems (DFT), 2012.

[133] Md. Tauhidur Rahman et al., “ARO-PUF: An Aging-Resistant Ring-Oscillator PUF Design,” in proc. Design, Automation, and Test in Europe (DATE), 2014.