8-18-2017

Computer Aided Design Solutions for Testability and Security

Qihang Shi
University of Connecticut - Storrs, shiqihang83@gmail.com

Computer Aided Design Solutions for Testability and Security

Qihang Shi, Ph.D.

University of Connecticut, 2017

As technology down scaling continues, new technical challenges emerge for the Integrated Circuits (IC) industry. One direct impact of down-scaling in feature sizes leads to elevated process variations, which has been complicating timing closure and requiring classification of fabricated ICs according to their maximum performance. To address this challenge, speed-binning based on on-chip delay sensor measurements has been proposed as alternative to current speed-binning methods. This practice requires advanced data analysis techniques for the binning result to be accurate. Down-scaling has also increased transistor count, which puts an increased burden on IC testing. In particular, increase in area and capacity of embedded memories has led to overhead in test time and loss test coverage, which is especially true for System-on-Chip (SOC)

designs. Indeed, expected increase in logic area between logic and memory cores will likely further undermine the current solution to the problem, the hierarchical test architecture. Further, widening use of information technology led to widened security concerns. In today's threat environment, both hardware Intellectual Properties (IP) and software security sensitive information can become target of attacks, malicious tampering, and unauthorized access. Therefore, it is necessary for existing design flows to be properly improved to address these new challenges.

Among possible options, mathematical optimization and novel architectural designs have time and again proved to be most promising approaches. In this work, we focus on developing new tools for this purpose by augmenting existing design flow. Implementation results on benchmarks proves proposed solutions to be effective and efficient.

Computer Aided Design Solutions for Testability and Security

Qihang Shi

B.E., Tsinghua University, Beijing, 2001

M.S., Lund University, Lund, 2009

A Dissertation

Submitted in Partial Fullfilment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2017

APPROVAL PAGE

Doctor of Philosophy Dissertation

Computer Aided Design Solutions for Testability and Security

Presented by

Qihang Shi, M.S.

Major Advisor

Mark M. Tehranipoor

Associate Advisor

Domenic Forte

Associate Advisor

Lei Wang

Associate Advisor

Rajeev Bansal

Associate Advisor

John Chandy

University of Connecticut

2017

To My Loving Family

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Dr. Mark M. Tehranipoor, for his support and guidance during my Ph.D. study at the University of Connecticut. Pursuing Ph.D. under his advisory was an opportunity that I treasured greatly and I hope in the end I've delivered enough to match the confidence invested in me. At times, it seemed questionable whether I would ever think fast enough to keep up with his analytical and critical prowess; yet as I push myself over the edge, I eventually saw myself performing at a rate I did not consider possible for me. Dr. Tehranipoor will forever remain a shining example of excellence as a researcher, mentor, and entrepreneur in my heart.

I would like to also express my gratitude to Dr. Domenic Forte for his inspiring and patient guidance over the past two years. Many of my small achievements would not have been possible without his insightful comments and advices. I shall admire and try my best to emulate his academic vision and diligent work ethnic in my future careers. I'd like to offer my sincere thanks to Dr. John Chandy, Dr. Rajeev Bansal, and Dr. Lei Wang for joining my Advisory Committees and providing constructive feedback. Dr. Chandy and Dr. Bansal both kindly offered help when I most needed it, to which I shall remain forever grateful.

I would like to further convey my thanks to all industrial coordinators who have supervised and advised my work, Mr. LeRoy Winemberg, Dr. Ramesh Tekumalla, and

Dr. Lisa McIlrath, who extended generous aid during my training to become a competent researcher.

Special thanks to all my labmates: Shuo Wang, Hassan Salmani, Xuehui Zhang, Wei Zhao, Jifeng Chen, Fang Bao, Niranjan Kayam, Kan Xiao, Ujjwal Guin, Gustavo Contreras, Kun Yang, Md. Tauhidur Rahman, Adib Nahiyan, Miao He, Zimu Guo, 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.

Words cannot delineate my appreciation to my family for their love and support. My parents, Yexin Shi and Yueping Zhou, whose confidence in me and support afforded me through my darkest years into a brighter future. My work is indeed impossible without your support through the years. As I dedicate this humble work to you, I hope it furnishes a little consolation for all the paths we opted not to tread.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Advance in IC fabrication technology has introduced a number of benefits to the industry. For example, integration of exponentially more devices is made possible for each newer generation of systems, and decrease in cost of cryptographic hardware has supported informatization of everyday life.

On the other hand, these advances did not come without cost.

Process variations on devices manufactured with lower feature sizes has increased. Conventionally, speed-binning has been done with functional tests, while more recent techniques use structural tests [2, 3]. However, both techniques have limitations. Due to these limitations, techniques based on post-silicon measurements with on-chip delay sensors have been developed [4–11]. In particular, Ring Oscillators (RO) and path-based ROs are used with reported success. However, for F_{MAX} estimation that is closely related to the speed-binning result, using data from all sensors might not be the best idea, and there are sensor combinations that could be optimized, which could yield better speed estimation speed and accuracy.

Increase in transistor count has lead to increase in area and capacity of embedded

memories in modern system-on-chips (SOCs). In 2013, the International Technology Roadmap for Semiconductors (ITRS) [12] predicted memory and uncore logic in microprocessors and SOC designs to continue to grow in area with each technology node advance while percentage of core logic area remains almost the same in the future. In the test section of its 2013 report, ITRS reported [13] 40% of respondents considered test cost as one of their major concerns, with 85% expecting it to become the biggest concern in the future. The trend of aggressive growth in embedded memories necessitates improvement in testing methodologies and further reduction of test cost. Research to improve the test of embedded memories is one among them.

Security sensitive information in portable devices made access of information convenient, meanwhile it also opened up possibilities of physical attacks. Microprobing is one kind of physical attack that directly probes at signal wires in order to extract sensitive information [14]. In a successful microprobing attack, plaintexts such as personal data, code format intellectual property (IP) or even encryption keys can be compromised [15]. Most security critical ICs reinforced against microprobing attacks with active shield to zeroize sensitive information once a breach has been detected. However, major problems exist with this approach. Active shields are designed to cover the entirety of the die, in some designs more than one metal routing layer is required. This puts a prohibitively high cost on the design, and leaves ICs fabricated with technologies offering smaller number of available routing layers dangerously exposed to probing attacks. Furthermore, research has shown using active shields in the top metal layer of an

IC to be very ineffective against microprobing attacks [16].

Finally, advanced fabrication technology has increased in cost, forcing outsourcing of fabrication, and casts doubts on IP security in a globalized IC supply chain. Most semiconductor companies now outsource fabrication to off-shore state-of-the-art foundries, which leads to gradual loss of control of the whole supply chain. Split manufacturing has been proposed recently as an approach to enable use of state-of-the-art semiconductor foundries while minimizing the risks to IP [17,18]. This method divides a design into Front End of Line (FEOL) and Back End of Line (BEOL) parts for fabrication at different foundries. Therefore, an untrusted foundry only has access to FEOL information, which is insufficient for them to constitute a threat to IP security of the design. Naturally, the security of split manufacturing depends on this assessment being true. However, it is currently difficult to verify this quality. Attacks and protections are proposed to clarify this very issue, but as of yet no definite answer exists for security of split manufacturing.

In light of these challenges, current design flow needs to adapt to properly address them. Research already exist that seek to address these issues, which are detailed in Section 1.1.

1.1 Related Work

Considerable research interest has been invested invested on all of the four particular challenges to modern IC design. According to the design challenges they seek to address, existing literature are summarized in following sections.

1.1.1 Speed-binning with Ring Oscillators and Modeling of Process Variations

In 2004, Wang *et al.* proposed to use ring oscillator (RO) based sensors to perform binning of wafers [4]. This approach uses a simple inverter chain which might not fit well with the task of predicting speed of an IC, which is determined by the slowest timing paths. Therefore, authors in [5] proposed to improve it with a novel path-based RO sensor called Path-RO sensor. Instead of using a simple inverter chain, the Path-RO sensor is based on actual critical paths of the functional design, and is believed to yield more accurate path delay information than traditional inverter ROs. Unfortunately, modern designs contain thousands of paths that might become critical due to process variation, and the amount is impossible for the few sensors to represent. To address this, in [10] the technique is further improved by introducing techniques to design paths whose delay will be representative of a number of potential critical paths instead of only representative of itself. However, a representative sensor may still become less than representative due to intra-die variation. Therefore in addition to improved sensor designs, a technique widely used is to insert multiple sensors at a number of locations. Some techniques, such as the one reported by Liu *et al.*, use a statistical model to

perform learning from measurements collected with on-chip RO sensors so that circuit delay can be predicted [19].

The first problem with current application of process variation and RO-based on-chip speed sensors stems from difficulties in mathematical modeling of process variation. To begin with, many assumptions that are commonly made in these models lack substantiation in reality. This include assumptions that are "typically true" such as assumption of all source of influences being additive, and assumptions that are simply assumed for the sake of simplicity, for example the assumption that *systematic* variation follows a spherical spatial correlation. Indeed, it has been suggested that *with-in-die* and *die-to-die* variations might not be statistically independent [20], which makes even the "typically true" assumptions shaky. To ensure validity, many assumptions taken in these models are corroborated with silicon data, which depend heavily on the particular technology used to fabricate test chips used to gather these data. For example, *systematic* and *random* variations are assumed to be equal in [20] based on empirical data in 32-nm technology [21]. For another design fabricated with another technology, the validity of these assumptions might not hold. In some other cases where details about the technology are withheld due to limited intellectual property authorization, so are precise modeling restricted. Therefore, methods based purely on mathematical modeling of process variation such as SSTA does not account for all application scenarios or satisfy all practical requirements. Moreover, on-chip speed sensors extract post-silicon data but are designed pre-silicon. In practice, these sensors are typically placed close to

functional circuits for the purpose of improving correlation in *systematic* variation [5] with functional circuits in its vicinity, or placed at regular grid locations to better represent the overall picture on the whole die [22]. Both practices assume that *systematic* variation are correlated to distance alone and are homogeneous and isometric often used in mathematical models of process variation. However, this might not be realistic considering the nature of variation sources. For example, variation in effective channel lengths could be caused by accumulative drift during photo-lithography, which obviously will neither be homogeneous nor isometric. When real correlation does not match model assumption, sensors could be placed at on-chip locations that represent poorly on process variation information they are designed to capture. Therefore, in reality it is unavoidable for speed sensors thus placed to have outlier sensors that negatively impact binning. These problems might be solvable with more precise mathematical models of process variation; however, a newer technology might invalidate all information gained on the old one. Further, more precise modeling might require higher CPU time to solve, which is exactly the problem these work set out to alleviate by using simplified models in the first place.

1.1.2 Raising Time Cost in System-on-Chip (SoC) Memory Tests

One approach to reduce memory test time is to run logic and memory tests at the same time [23–25]. These methods are mainly concerned with loading of MBIST controller vectors, but paid very little attention to whether test coverage is lost by the bypass on

memory interface.

Another more recent development that also allows concurrent test of memory and logic is through test scheduling for individually wrapped design blocks [26–28]. Each wrapped design block, referred to as âĂIJcoreâĂİ, is tested with boundary registers whose values are shifted-in through a test control line (e.g., one that complies with IEEE 1500-2005 standard [28]) that is in turn connected to test processors and external automatic test equipment (ATE). The test processor regulates test control signals on the test control line and can be configured to allow multiple cores to conduct test at the same time. Test scheduling algorithms have been developed to optimize concurrent test among all SOC cores [29, 30], that resulted in significant reduction in test time compared to testing the blocks in sequence [29]. This approach is tailored for large SOC designs and has seen wide industry adoption [31, 32] and academic interest [33–35]. However, on a lower level the test is not conducted on paths that starts and ends in different cores.

Usually memory cores are tested individually without involvement of any logic core that may access it, and this could mean that the longest path to memory cells during functional mode may be left untested. As area of embedded memories increase in modern SOC designs, test of memory access paths also increase in importance. Due to the increased area of both uncore logic and memory, interconnect load on memory access paths increases, resulting in aggravated transition and path delay faults (TDFs and PDFs) and creating possible critical or speed paths. Since memory access remains

as one of the most critical paths, delay tests in these paths have also become critical to speed binning of the complete design [36]. Recent literature presented several methods to address this problem [37–39]. Authors in [37] showed a memory test optimization algorithm for better fault coverage in memory and peripheral logic. Another research proposed various improvements to memory interface tests in preventing X-propagation, allowing detection of memory input faults as well as test time improvement based on scan architecture modification [38]. Despite improvement on test time, this method remains a scan-based test, and does little to ensure concurrent application of logic and memory test. Another recent approach proposed to test the memory access paths by scanning in test patterns using the fan-in and fan-out registers [39]. This method solves PDF test for memory access paths, however it does not allow concurrent application of logic and memory tests.

1.1.3 Threat and Countermeasures of Circuit Microprobing Attacks

Circuit microprobing refers to techniques that allow an attacker to directly observe partial or full sensitive information, e. g. plaintexts or encryption keys. ICs designed for security-critical applications such as smartcards, microcontrollers in mobile devices and security tokens [16, 40, 41] are among the most common victims to this kind of attacks. Unfortunately, a lot of these applications also have exploitable security weaknesses [41], probably due to tight budget margins. Wires of targeted nets that the attacker wishes to reach are likely buried under multiple passivation, metal, and dielectric

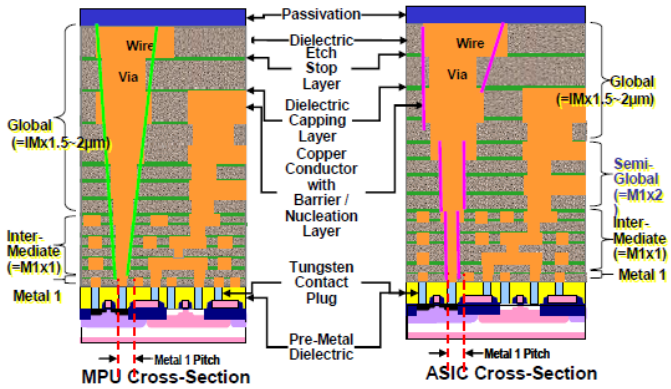


Fig. 1.1: Typical cross-sections of microprocessors (MPU) and Application-Specific Integrated Circuits (ASIC) [1]

.

layers (shown in Figure 1.1). The attacker either has to mill through these obstacles, or mill through from the silicon substrate, the so-called *back-side* attacks. [42] This is facilitated by utilizing either the phenomenon of Photon Emission (PE) or Laser Voltage Techniques (LVX) [43]. However, both methods require observation of photon emissions, therefore limiting their effectiveness as feature size scales down. Further, it can also be prevented by fabricating a *back-to-back* 3D IC to avoid leaving back-side exposed [44]. Therefore, protection methods that assume milling through metal wires from the *front-side* is still a worthy assumption for antiprobing designs.

Existing techniques designed to stop microprobing usually perform their duty by detecting and then zeroizing sensitive information. The more widely studied and attempted approach is to detect hardware tampering by building a mesh of trigger wires to cover the design [44–48]. This is called an active shield, because the trigger wires are supposed to be constantly monitored in order to detect an attack. Some shield

designs are analog: for example, the authors in [45] use capacitance measurement to detect damage done to it, and thereby detect tampering. The problem with analog shield designs is that analog sensors rely on parametric measurement, which has been shown to be weaker [40]. Therefore, digital active shields have been proposed [44, 47, 48]. These methods send digital random vectors through the trigger wires, and check whether received vectors are altered. A milling through the mesh would be reliably detected when it cuts off at least one of the trigger wires.

The digital active shield design is most commonly implemented in an industrial design, so much that published attacks has also proposed exploits to beat them. [40, 41] Conversely, this also has encouraged dedicated protections that invalidates these exploits by antiprobing designers. The authors in [44] investigated the problem of possible *prediction attack*, where an attacker could predict the next random vector to be sent if the random vector generation is not secure enough. The authors then presented a design where block ciphers in Cipher Block Chaining (CBC) mode are used to generate secure random vectors. Another research proposed to obfuscate layout routing of the active shield so that the attacker would not be able to figure out how to perform a successful rerouting attack [48].

1.1.4 Threat of Untrusted Foundries and Existing Countermeasures

Due to globalization of IC supply chain, IC fabrication has been off-shored to overseas sites, raising concerns on whether trust between IP owner and overseas IC fabricator

could be established [49]. A foundry with malicious intent could conduct a number of attacks such IP piracy [50], IC cloning and overproduction [51, 52], hardware Trojan insertion [53]. Such a threat to IP security is often referred to as an *untrusted foundry*.

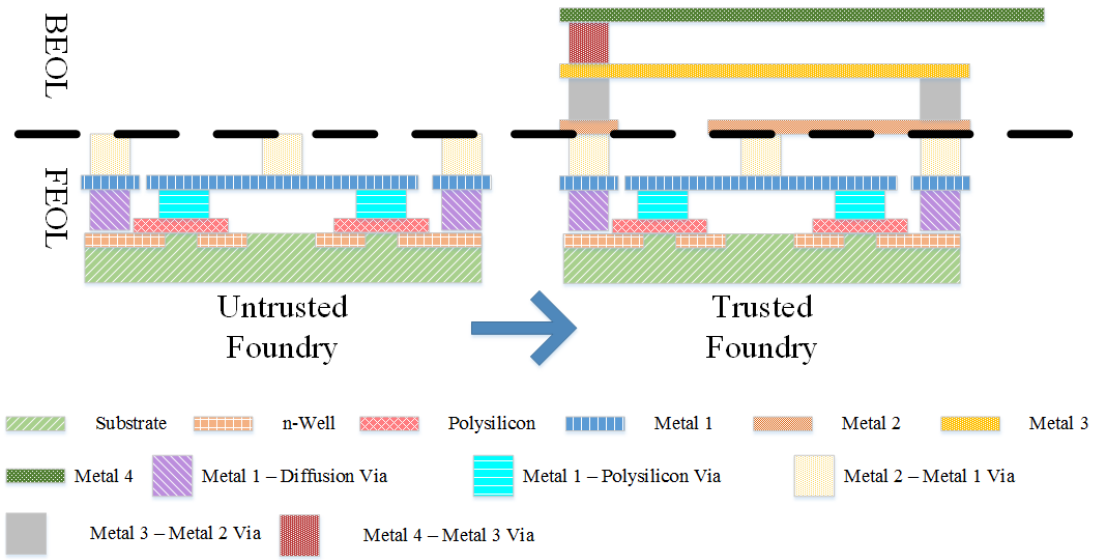


Fig. 1.2: Typical split manufacturing arrangement (assuming split made between Metal 2 and Metal 1 layers).

Research exist to address threat of all possible attacks by untrusted foundries. Split manufacturing, also known as split fabrication, is a novel technique proposed to address the threat to IP security posed by untrustworthy foundries [17]. The technique proposes to split an IC layout ready for fabrication into multiple parts, and fabricate each part at a separate foundry. One example of split manufacturing, as shown in Figure 5.1, has the untrusted foundry manufacture the front-end-of-line (FEOL) part of the IC, and then ship it to a trusted foundry to deposit back-end-of-line part onto it. By denying the untrusted foundry complete layout information, split manufacturing prevents it from stealing IP information, and deters attacks that require reverse engi-

neering of the design. Meanwhile, techniques against hardware Trojan insertion exist in two categories characterized by how they address the issue: The first category focus on detecting Trojans, either by functional verification, side-channel signal analysis, or by new front-end design techniques such as design-for-trust [54–62]. Techniques in this category detect existence of hardware Trojans by generating a signature of the circuit under test (CUT), then classify the CUT with this signature. To perform classification, they require a golden model, i.e. signature of a copy of the same circuit that is known to be free from hardware Trojans. Unfortunately, it remains doubtful whether golden models can be acquired for real world applications. On the other hand, technique focused on preventing hardware Trojans from being inserted does not have this requirement. Built-in self authentication (BISA) is the first proposed technique to prevent hardware Trojan insertion in circuit layout and mask. By occupying all available spaces for Trojan insertion and detecting malicious removal through built-in self test, BISA is able to deter hardware Trojan insertion without the requirement of golden models.

However, problems remain. To begin with, techniques against IP piracy do not usually consider the threat of hardware Trojan insertion, and neither do techniques against hardware Trojan insertion consider IP theft, despite both attacks could be committed by their intended adversary.

An untrusted foundry is characterized by two attributes: First, the service of an untrusted foundry is imperative, otherwise it would be secure enough simply by replacing it with a trusted foundry; Second, an untrusted foundry cannot be trusted

with the security of the intellectual property (IP), hence additional measures need to be taken to prevent it from jeopardizing it. Since both attributes need to be present for split manufacturing to be necessary, we may assume all untrusted foundries in the adversarial model possess both attributes.

A natural inference from this fact is, to ensure security of a fabrication with split manufacturing, we must ensure security of all possible attacks from the untrusted foundry considering its capability. Due to the first quality, it is likely that the untrusted foundry is equally aware of its own importance, so there is no disincentive for the untrusted foundry from trying all possible attacks given its technical capability. At the same time, as a result of the second quality, the IP owner has no reason to trust the untrusted foundry not to. In all likelihood, untrusted foundries will likely try all attacks in their arsenal, and IP owners will desire an overall solution that will secure his design against all of them. Therefore, a complete security solution to address the threat of untrusted foundry needs to consider all possible attacks, and both split manufacturing and BISA are limited.

Built-In Self-Authentication (BISA)

Built-In Self-Authentication (BISA) prevents hardware Trojan insertion by exhausting one resource essential to it: *white spaces*. Normally, during the placement step of the back-end design of the circuit, gates in the circuit are placed at optimized locations based on density and routability [63]. This leaves spaces in the layout that are not

filled with standard cells. For design purposes other than security against hardware Trojan insertion, these white spaces can be filled with *filler cells* to serve as decoupling capacitors and/or extension of power tracks [64]. However, unsupervised filler cells are prone to malicious removal by Trojan inserters in order to make room for hardware Trojans. If they are replaced by Trojan gates, no performance loss significant enough to raise suspicion will likely be incurred, since Trojan gates are rarely triggered as well.

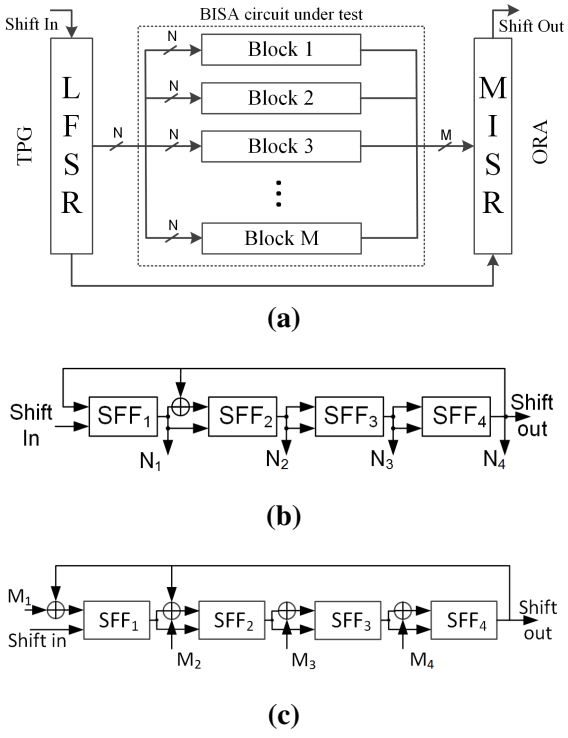


Fig. 1.3: Structure of (a) BISA, (b) four-stage LFSR, and (c) four-stage MISR.

BISA architecture BISA prevents hardware Trojan insertion by occupying white spaces with testable standard cells instead. Then, all inserted BISA cells are connected into tree-like structures to form a built-in self test (BIST) circuitry, so that they could

be tested to verify no BISA cell has been removed. Removal of its member cells will lead to a BIST failure, so that no attempt to make room for hardware Trojans will evade detection. As shown in Figure 1.3a, BISA consists of three parts: the BISA circuit under test, the test pattern generator (TPG), and the output response analyzer (ORA). 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 test coverage, the BISA circuit is divided into a number of smaller combinational logic blocks, called BISA blocks shown in Figure 1.3a. 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. TPG has been implemented with Linear Feedback Shift Register (LFSR)while ORA has been implemented with Multiple Input Signature Register (MISR) in prior work [65]. Examples of four-stage LFSR and four-stage MISR are shown in Figure 1.3b and 1.3c. They are used in the generation of random vectors and compression of responses into a signature. SFF in the figure represents scan flip flop. Other types of TPG and ORA can also be applied [66].

The main advantage of BISA over other techniques with similar objectives is that it has no golden chip requirement. Most other research on addressing the issue of Trojan insertions have focused on the development of:

1. Hardware Trojan detection techniques using functional verification and side-channel signal analysis (applied post-silicon), or

2. New design techniques to improve detection capabilities (applied to front-end design) [67].

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, if not impossible in many scenarios. 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, impact of BISA on the original design in terms of area and power is also negligible.

Attacks on BISA The attack most likely to succeed against BISA is the so-called redesign attack. This attack replaces original circuitry with smaller functionally equivalent circuitry to make room for Trojan insertion. Both BISA and original circuitry can be targeted in this attack. Redesigning the original circuit will result in significant changes of the electrical parameters, such as power and path delay. These could be detected by delay-based and power-based techniques [68–74]. Therefore, it is more likely for the attack to succeed against BISA cells. It is possible to further secure BISA cells by performing this optimization on BISA design to prevent this particular attack. However, the attacker can also choose to design a custom cell functionally equivalent to several BISA cell in order to make room for Trojan insertion. Prevention of such an attack would require anticipation of all possible custom cell designs that are functionally equivalent to any combination of BISA cells. That is not likely feasible except for

very small BISA circuitry. Therefore we believe this attack remains a possible threat to BISA security.

Further, BISA is also under a few limitations on its performance. For example, due to the existence of resizing attack, all BISA cells have to be of the smallest variant in area among standard cells of the same function, which might make it easier for the attacker to identify them.

Split Manufacturing Security and Limitations

Prior works on split manufacturing security The prior work in this field [75–77] is motivated by one major objective: to establish a sound metric of security for designs fabricated using various split manufacturing methods. Studies focused on this problem generally agree that the key to split manufacturing security is how difficult it is for an untrusted foundry to recover the hidden BEOL information. Depending on the source of information perceived crucial for the untrusted foundry to obtain BEOL information, existing study on the problem of security metrics can be classified into two categories: metrics that evaluate the layout, and metrics that evaluate the graph connectivity of FEOL part of the circuit left observable.

Most researchers attack the problem from a layout point of view. Publications in this category often examine irregularities in the layout (also known as “hints” or “clues”) and theorize how they can be used by a hypothetical attacker. Authors in [75] proposed *proximity attack*, which simulates an attacker who makes educated guesses

on BEOL connections of open input/output pins in FEOL. [78] proposed to also consider load capacitance limitations as well as the direction of dangling wires, while [79] discussed more definitions of proximity based on known router behaviors. Another study [77] also uses layout information, but instead of performing hypothetical attacks, it seeks to define objective measures of the layout that might become useful to exploit. This leads to the following metrics being proposed:

1. *Neighborhood Connectedness (NC)* is defined as function of count of connections per neighbor cell within range R, i.e. $C(R) = \frac{\sum \text{connections to neighbors in } R}{\sum \text{neighbors in } R}$;

2. *Standard-cell composition bias* is designed to reflect imbalance in number of instances of specific cells considered representative of design function*;

3. *Cell-level obfuscation* measures percentage of standard cells that has functionality hidden with obfuscation;

4. *Entropy* measures disorder in FEOL information, and is given by $E = -\sum_{i=1}^{N} p_i \log(p_i)$, where N is total number of cell types and p_i is the percentage of each cell type among all cells; and

5. Overheads in area and delay are used to measure the cost of the obfuscation technique.

Security metrics based on layout information has the advantage of being readily available to designers through layout editor tools, and have seen adoption in a number

*e.g. XOR gates for their correlation with cryptographic circuits, flip-flops for their correlation with state machines, adders for their correlation with arithmetic cores, etc.

of recently published techniques intended to strengthen security for split manufactured designs [80, 81]. Unfortunately, problems remain to plague this approach. First, since no attack has been reported to have successfully reconstructed BEOL connection from FEOL clues, all hypothetical attacks and objective metrics remain as convincing as another. Further, proposed metrics themselves don't scale linearly with difficulty in any conceivable attack, which makes it difficult for designers to estimate how much protection is enough. For example, in a proximity attack, the percentage of correct guesses means nothing without the knowledge of which connections crucial to design functionality are correctly guessed. And finally, when multiple metric values are measured, it is very difficult to find a way to estimate the relative importance among them. Without this, the designer will have no other option than to ensure arbitrarily high (again, it is very hard for him to know how high) security on all of them. Hence, trading-off between security metrics will be virtually impossible. In contrast, another research [76] evaluates split manufacturing security based on graph connectivity of FEOL layout and seeks to define difficulty by solving for the dimensions of the solution space from which the attacker must pick one correct solution. The proposed metric operates on Directional Acyclic Graphs (DAG) abstracted from both the complete layout (G) and FEOL layout (H) (see Figure 1.4a for an example). In the resulting DAG, gates are abstracted into vertices (represented with colored circles in Figure 1.4a, whose color represents models of each gate), and nets are abstracted into sets of directional edges, each of which corresponds to a driving gate/vertex and driven gate/vertex pair (repre-

sented with arrows in Figure 1.4a). Then it computes the number of legal mappings that maps each gate u_i in complete layout graph G to a distinct gate v_j in FEOL layout graph H. This number k is defined as the security of that gate, and the security of the complete layout is defined as the lowest k of all gates. In the example shown in Figure 1.4, XOR gates have a security $k = 2$ but all other gates have $k = 1$, therefore the overall security of the circuit remains at $k = 1$. This security metric is often referred to using its letter of choice k as *k-security*. A greedy algorithm is then presented to find a minimal subset of wires in the layout to uplift to BEOL while satisfying minimum security k, an optimization of split manufacturing security also known as *wire lifting*.

Unlike layout based approaches, k-security is defined based on number of possible mappings between cells in the layout and cells in the netlist. This is because by definition, a specific attack against split manufacturing will be aiming at recovering the BEOL connections; any attack that does not have this ultimate goal is not an attack specifically against split manufacturing, and any attack that succeeds in this goal will have completely compromised split manufacturing. This definition has two advantages. One, it is quantifiable, meaning an optimization algorithm can be designed with k-security as its objective function to improve the absolute security of the BEOL connections, instead of simply preventing every known loopholes. Two, its definition is not dependent on specific layout, which makes it compatible with most layout based approaches, and secure even when netlist of the design is compromised [76].

It is also effective against a wide spectrum of threats, owing to the fact that few

attack is possible without making sense of what function gates in the layout serves.

However, it is not without weaknesses. Its first disadvantage is its difficulty to compute. k-security definition checks whether any given FEOL netlist has a security level of k by checking a property called subgraph isomorphism, which makes computation of the security level of any wire lifting solution NP-hard [76]. The proposed wire lifting algorithm in [76] functions by procedurally checking if lifting another wire lowers the security level, which makes it exponentially more complex. This makes it extremely computationally costly, and even less scalable than typical NP-hard algorithms. In addition, the fact that k-security is defined using graph connectivity also results in lack of consideration on any leakage of information from FEOL layout. Indeed, the authors acknowledged this weakness, and proposed to remove it by performing layout design *after* generation of FEOL netlist. This understandably deteriorated the performance of the design in every possible way, as it relies on layout *not* being optimized for performance to keep it from leaking security information. Another example of its restrictions on performance optimization is its requirement to restrict the number of standard cell models. Optimization during netlist synthesis otherwise unrestricted often instantiates many less common standard cell models, and this long tail distribution will seriously restrict highest possible k any wire lifting can ever hope to reach. Therefore, k-security based wire lifting will have to restrict number of standard cell models synthesizer is allowed to use, which in turn restricts a number of design performance parameters. To summarize, k-security and wire lifting technique based on it offers a

better defined metric, but is too restrictive and too slow for industrial-level application.

Limitation of split manufacturing

Split manufacturing prevents all attacks that require complete knowledge of the whole layout, which also includes attacks against BISA such as identification of BISA cells. However, not all attacks require complete knowledge of the whole layout. One example is the untargeted Trojan insertion [82], a threat discussed in details in [83]. Untargeted hardware Trojans do not target specific functions of the original circuitry, and therefore cannot commit attacks that require knowledge of such functions. Nevertheless, untargeted hardware Trojans are still capable of degrading the performance and/or reliability of manufactured ICs, or trigger a denial-of-service (DoS) in critical control systems [84]. This constitutes a threat that still needs to be addressed.

1.2 Motivations and Contributions

To improve upon existing proposals to better address the challenges in modern IC design, we believe computer aided design techniques could contribute the following improvements to the current practices:

- ***To better address impact of elevated process variation upon speed binning:*** As was discussed in Section 1.1.1, problems with current methods on speed-binning and process variation modeling in general exist in primarily two aspects: the process

variation and design complexity cause RO-based speed sensors to deviate from IC speed (i.e. F_{MAX}), and this problem is difficult for current models of process variations to accurately model. Considering these implications, we propose to improve on-chip speed sensor based binning with optimized sensor selection with Genetic Algorithm [85], a method insensitive to parameter change and doesn't require very powerful hardware computing power. By finding and removing from consideration sensors whose information prove detrimental to speed binning with RO-based on-chip speed sensor information, this method could potentially offer improvement in binning precision. In this work we claim the following contributions:

1. A genetic algorithm to find an optimized set of sensors that will yield better estimation accuracy;

2. Evaluation of proposed algorithm using path-delay data from SPICE-based timing simulations;

3. Evaluation of proposed algorithm using silicon measurements on fabricated commercial ICs.

• ***To improve embedded memory test application and facilitate test coverage for uncore logic:*** We propose a novel design-for-test architecture to provide a capability to both concurrent test of logic and memory blocks, and delay fault test on functional memory access paths.

1. We develop a gate-level test architecture to enable concurrent application of

memory and logic tests by bypassing memory test access at fan-in registers of the memory core, instead of conventional bypass at memory interface.

2. We also configure the proposed test architecture to allow detection of delay faults in functional memory access paths, by reconfiguring registers on target paths into test pattern generators/response compactors.

3. We also present using at-speed memory test to target delay faults on these paths.

• ***To address the problem of microprobing attacks and improve antiprobing designs:*** We propose a framework to examine protection designs in terms of their performance against known exploits, and evaluate them based on how much trouble it would take the attacker to circumvent their defenses. Specifically, in this field, our work will provide the following contributions:

1. A layout-driven framework to assess designs against microprobing attacks considering known attacks and exploits.

2. A thorough mathematical analysis on bypassing attack at any angle.

3. A verification algorithm based on a mainstream layout editor (Synopsys IC compiler) to quantitatively evaluate a post-place-and-route design in terms of exposed area vulnerable to microprobing by security-critical nets; To our knowledge, the proposed algorithm is the first to provide this capability.

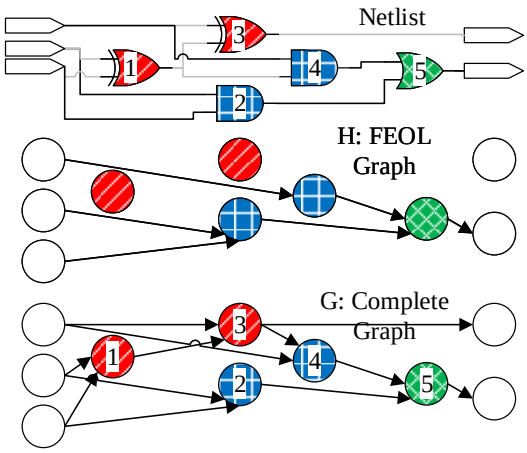
• ***To address the problem of untrusted foundry:*** We propose a new method based on existing built-in self-authentication (BISA) technique that is enhanced by split

manufacturing technique with back-end-of-line (BEOL) wire lifting optimization, so that the new technique, known as obfuscated BISA (OBISA) can be secure against both hardware Trojan insertion and IP theft, without suffering from the same vulnerabilities of either BISA or split manufacturing. Our contributions in this problem include the following:

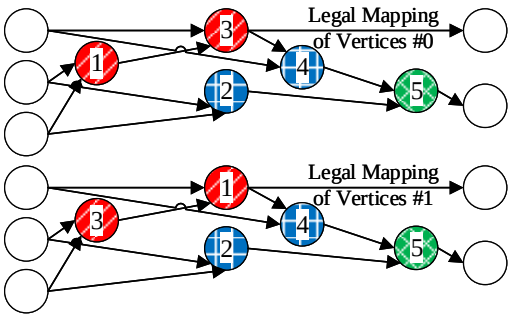
1. An OBISA technique that is secure against untargeted hardware Trojan insertion and secure against IP piracy and IC cloning at the same time, both of which are primary strengths of BISA and split manufacturing.

2. In addition to inheriting strengths from both parent techniques, the combined OBISA technique will also utilize both techniques to strengthen the other. This includes making the resulting OBISA technique secure against redesign attack, making OBISA more independent from golden models by further reducing reliance on detection-based anti-Trojan techniques, utilizing additional OBISA cells to improve wire-lifting security and relaxing restrictions on original circuitry, and reduce threat of proximity attacks, etc.

3. In addition to producing a secure technique, the proposed OBISA technique is also going to improve wire-lifting by reducing design overheads, hardware costs, and computational costs.



(a) A split manufactured full adder, its FEOL graph H, and its complete graph G.



(b) Legal mappings of vertices in FEOL graph H to vertices in complete graph G.

Fig. 1.4: Principle of k-security applied to a full adder circuit as an example.

Chapter 2

OPTIMIZED SENSOR SELECTION WITH GENETIC ALGORITHM

In this chapter, we are concerned with accuracy of speed-binning of manufactured ICs with measured results from on-chip speed sensors. We assume that limited numbers of sensor blocks are placed across different locations on the layout. In each sensor block, we consider two kinds of delay sensors, namely RO sensors (Figure 2.1) and Path-RO sensors (Figure 2.2). RO sensors are ring oscillators whose rings consist of inverter chains only. When measured, RO sensors provide number of oscillation cycles in a given period of time. Since ring oscillators have the same type of gate (inverters) and the same fan-out values among these gates, their measured values are highly homogeneous, and could serve as comparison standards. The other kind of sensor we use is

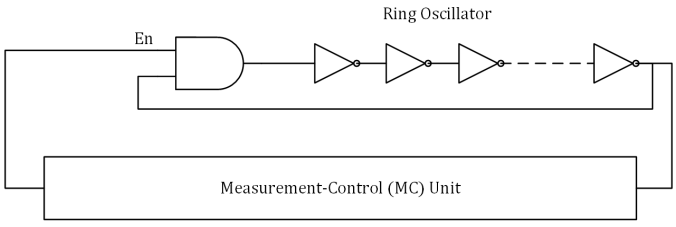


Fig. 2.1: Block diagram of an RO sensor.

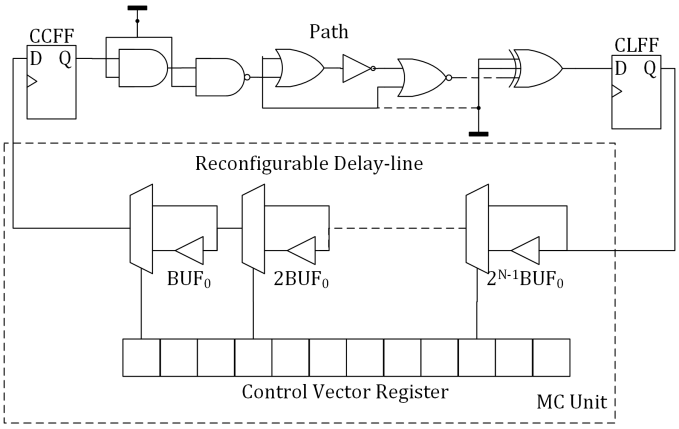


Fig. 2.2: Block diagram of an Path-RO sensor.

the Path-RO sensor. It is a path delay sensor proposed and extensively described by Wang *et al* [86]. The purpose of using actual delay paths to construct sensors is to better represent impact of process variation upon path delay. Since different gate models react differently to the same process variation, RO sensors consisted of only inverters can only represent inverter gate delay. Therefore, it is only possible to study impact of process variation upon path delay with sensors that measure path delay itself.

This chapter will first present the proposed optimization algorithm that finds a set of sensors which will yield better estimation accuracy in Section 2.1. Then in Section 2.2, path-delay data from SPICE-based timing simulation is presented to evaluate the performance of the proposed algorithm. In section 2.6, results using silicon measurements are presented and analyzed. Finally, the chapter is concluded in Section 2.7.

2.1 Optimization Algorithm

In this section we present an optimization algorithm to find a set of sensors which could yield improved speed-binning results. The speed binning in this study is done with Support Vector Machine classification [87]. SVM is a machine learning based classification technique whose efficiency is irrelevant to the particular technology, design or prior knowledge. In reality, the sensor-based classification problem is likely not linearly separable; however, it is beyond the scope of this work to find a most suitable kernel function for SVM. To simplify the issue, the SVM classification method is implemented with a simple linear kernel function for a binary classification, assuming that only two bins are required. This blind classification technique is chosen because it could perform equally well whether statistical information about the process variation is available or not. We have performed experiments in both of these situations in Sections 2.2 and 2.6.

For the speed binning in this study, we assume a total number of J dies are to be binned. Of those, J' dies are used to train the prediction model by supplying the solver with their sensor measurements as vector and their F_{MAX} as classification. After the learning is completed, sensor measurements from the other $J - J'$ dies are supplied for classification. This classification result is then compared against their given F_{MAX} classification to evaluate classification accuracy.

The proposed genetic algorithm is shown in Figure 2.3. This algorithm is suitable for our purpose for a couple of reasons: Firstly, a genetic algorithm does not require

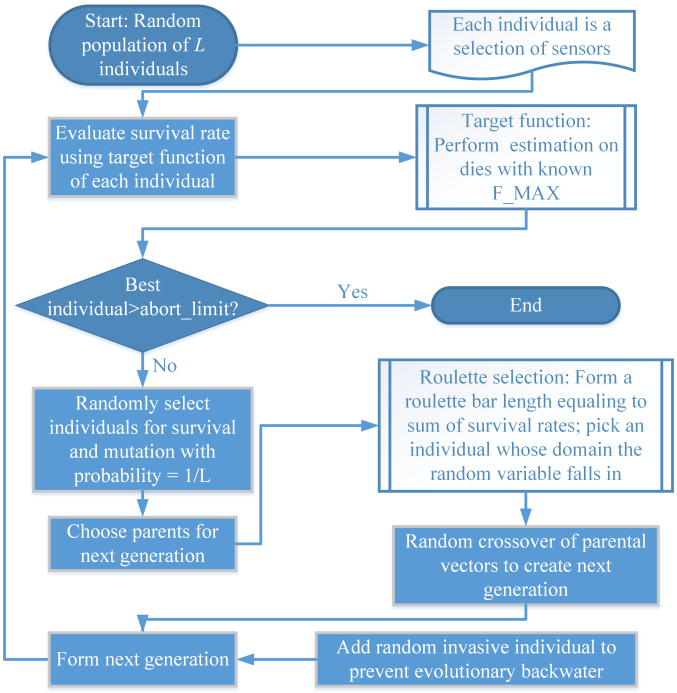


Fig. 2.3: Flowchart of the proposed optimization algorithm.

any knowledge of underlying mechanism, which makes it universal for all processes; in addition, its improvement can be guaranteed. The algorithm is designed favoring high suitability individuals to expedite evolution, and the drawback being covered with additional random individuals. As shown in the figure, the algorithm starts with a random population, evaluates their survival rate, randomly choose survival population, perform mutation and reproduction, repeat until a desired survival rate is reached ("abort_limit" in the flowchart). Since the algorithm intends to optimize sensor selection, we use a total number of S 1-dimensional binary vectors to represent each individual of the population. Here, each individual represents a possible solution, and S is the size of the gene pool. Each individual's representing binary vector is consisted of L binary values, each value represents whether measurement from a corresponding sensor should be

used or not. Here, L is the total number of sensors on each die. The target function is a function that evaluates the accuracy of the speed-binning technique of choice, e. g. linear Support Vector Machine (SVM), by supplying it with training sequences and then compare its classifications with reliable F_{MAX}. This entails a number of G dies in addition to training dies dedicated to this purpose. For each individual, the target function returns a prediction accuracy of the technique assuming the choice of sensors specified by the individual's binary vector is used. In terms of SVM classification, the prediction accuracy is the test error of the classification. Since prediction accuracy is a natural number $a_s \in [0, 1], s \in \{1, \ldots, S\}$, it is used as the survival rate of each individual in the following phase where individuals are selected for survival. In the following survival stage, S uniformly distributed random numbers $\mathbb{P} \in [0, 1]$ are compared against the survival rate of every individual to find those that survived this generation. These survived individuals are randomly subjected to mutation: each of their binary values is subjected to probability $P = \frac{1}{L}$ logic inversion. Then from the resulting individual, a roulette selection is performed to find the parents of next generation of individuals. This is done by performing a roulette selection, where a uniformly distributed random test $\mathbb{R} \in [0, \sum_{s=1}^{S} a_s]$ is taken once for each individual, each random result chooses one individual by falling into its "slot" whose size equals to the individual's survival rate. To avoid falling into evolutionary dead-water, for each generation I "invading" individuals are introduced into next generation, which are generated randomly to prevent evolutionary backwater. The other $S - I$ individuals of the next generation are pro-

duced by randomly crossing $S - I$ pairs of parents drawn with roulette selection from parent generation. This concludes processing for the current generation and creates the population of the next generation. The optimization continues until a limited number of generation or a desirable survival rate is reached. Upon conclusion, the algorithm selects the best performing individual to be used for classification.

2.2 Simulated Binning using SPICE

To verify the validity and usefulness of the proposed optimization, we first implement a binning based on SPICE-based simulation data. In this section, we introduce the method and purpose of this simulation, models and assumptions made to facilitate this simulation, then demonstrate the effect of speed binning using the proposed optimization algorithm.

2.2.1 Objective and Method

This simulation is intended to study the effect of the proposed algorithm on varying process variation and sensor design scenarios that might appear in different technology and designs, which will be difficult to study with silicon data. For this purpose, IC speed and sensor data are produced with simulation. A "true" (i. e., assumed to be accurate) speed binning is then performed with IC speed using a simple two-bin speed binning scheme. A number of ICs are randomly selected to form a training batch for the speed-sensor-based binning method using SVM. The trained model is then used to perform

binning on the rest of ICs. This process will be used as an exemplary speed-sensor-based binning with *unoptimized* sensor selection. Then an optimization is performed using the proposed algorithm to try to find an *optimized* sensor selection during the training phase, whose trained model will be used to perform a second binning on the rest of ICs. This binning result, called binning with *optimized* sensor selection, is then compared with the previous result to evaluate its improvement.

In this study we obtain IC speed with simulated path delays of the longest paths of the design. In other words, IC speed used in this study is defined as the delay of the longest path on the die under the influence of process variation. The total standard deviation of the process variation we shall cover in details in the next section.

2.2.2 Model Assumptions used in this Simulation

Model of process variation used in this evaluation is taken from the VARIUS model [20]. The VARIUS model considers three sources of variation: the *die-to-die* variation, assumed to be independent; the *random* and *systematic* parts of *with-in-die* variation. This simulation also follows the assumption made in the VARIUS model that $\sigma_{\text{random}} = \sigma_{\text{systematic}} = \frac{1}{\sqrt{2}}\sigma_{\text{WID}}$, and that systematic variation in effective channel length is dependent on the systematic variation in threshold voltage ($x_{\text{systematic, } L_{\text{eff}}} = \frac{1}{2}x_{\text{systematic},V_{\text{th}}}$). Since VARIUS model is focused on modeling *with-in-die* variation, we add the assumption that $\sigma_{\text{D2D}} = \sqrt{2}\sigma_{\text{random}} = \sqrt{2}\sigma_{\text{systematic}} = \sigma_{\text{WID}}$, an assumption also made in some earlier research [88]. Further, we use correlated random variables to generate system-

atic variation [89], whose correlation values are randomly generated with a uniform distribution on $[-1, 1]$.

In order to limit number of paths necessary to cover to avoid unrealistically long simulation time, we have restricted our simulated paths to top 20% paths, and accordingly adjusted standard deviation of process variations to $\sigma_{\text{D2D}} = \sigma_{\text{WID}} = \frac{1}{3\sqrt{2}} 20\% \approx 4.71\%$.

In summary, parameters and assumptions used in the mathematical model are shown in Table 2.1.

2.2.3 Method of Implementation

Most of the variation figures used in this study were generated with MATLAB® [90] and Statistics and Machine Learning Toolbox™ Release 2012b [90]. In particular, generation of systematic variation in threshold voltage is done by *mvnrnd* tool that generates correlated Gaussian distributed random samples on specified correlations, which is in turn generated as uniformly distributed random numbers in $[-1, 1]$ with a published method [89]. The only exception to this is the random variation, which is implemented using local variation block provided by Synopsys' [91] HSPICE® version K-2015.06 [92, 93] circuit simulator, since it would be otherwise impossible to append variation parameter to the generic subckt definition of standard gates.

The benchmark design that provides paths used in this study is one single OpenSPARC™ [94] T1 core [95]. Static timing analysis with the help of Synopsys' PrimeTime® tim-

Table 2.1: Parametric assumptions used in mathematical model for path and sensor delay simulations.

Name	Role	Value
$x_{[L_{\text{eff}}\|V_{\text{th}}],i,j}$	Total relative variation in effective channel length or threshold voltage at one device of path i on die j.	$x_{\text{D2D},[L_{\text{eff}}\|V_{\text{th}}],\text{j}} + x_{\text{rand},[L_{\text{eff}}\|V_{\text{th}}]} + x_{\text{systematic},[L_{\text{eff}}\|V_{\text{th}}],\text{i,j}}$
$x_{\text{D2D},[L_{\text{eff}}\|V_{\text{th}}]}$	*Die-to-die* variation in effective channel length or threshold voltage at all devices on die j.	$N(0, \sigma_{\text{D2D},[L_{\text{eff}}\|V_{\text{th}}],\text{j}})$
$\sigma_{\text{D2D},[L_{\text{eff}}\|V_{\text{th}}],\text{j}}$	Standard deviation of *die-to-die* variation on effective channel length or threshold voltage at all devices on die j.	$\frac{1}{3\sqrt{2}} \times 20\% \approx 4.71\%$
$x_{\text{random},[L_{\text{eff}}\|V_{\text{th}}]}$	Random component of *with-in-die* variation in effective channel length or threshold voltage at a device.	$N(0, \sigma_{\text{random},[L_{\text{eff}}\|V_{\text{th}}]})$
$\sigma_{\text{random},[L_{\text{eff}}\|V_{\text{th}}]}$	Standard deviation of the random component of *with-in-die* variation in effective channel length or threshold voltage at a device.	$\frac{1}{6} \times 20\% \approx 3.33\%$
$x_{\text{systematic},V_{\text{th}},\text{i,j}}$	Systematic component of *with-in-die* variation in threshold voltage at all devices of path i on die j.	$N(0, \sigma_{\text{systematic},V_{\text{th}},\text{i,j}})$
$\sigma_{\text{systematic},V_{\text{th}},\text{i,j}}$	Standard deviation of the systematic component of *with-in-die* variation in threshold voltage at all devices of path i on die j.	$\frac{1}{6} \times 20\% \approx 3.33\%$
$x_{\text{systematic},L_{\text{eff}},\text{i,j}}$	Systematic component of *with-in-die* variation in effective channel length at all devices of path i on die j.	$\frac{1}{2} x_{\text{systematic},V_{\text{th}},\text{i,j}}$
$\rho_{\text{systematic},V_{\text{th}},i:i'}$	Correlation between systematic variations in threshold voltage at all devices of path i and path i' on all dies.	$U(-1, 1)$

ing analyzer yields 438 paths that needs to be covered, which are exported to SPICE decks for 200 dies, each using a different set of variation parameters generated with MATLAB® [90] and Statistics and Machine Learning Toolbox™ Release 2012b [90]. The Synopsys $32/28$nm Generic Library for Teaching IC Design [96] supplied by Synopsys [91] were used to synthesize the design and provide physical SPICE models.

Sensor delay are also collected using SPICE simulation. 10 sensor blocks are assumed to be placed in vicinity of the 10 longest static timing paths, giving them 90% correlation in systematic variation with said paths. This is a likely scenario in real on-chip speed sensor implementation since a real implementation is unlikely able to allow for very large number of speed sensors due to area overhead, and despite designers' effort to place sensors as close to functional paths as possible, they still would not experience exactly the same variations as the paths they try to mimic. In each sensor block, we used 8 kinds of RO sensor and 8 kinds of Path-RO sensors are inserted in each sensor block, totaling 160 sensor values per each simulated die. We have constructed RO sensors with 31, 51, 71, and 91 inverters long, and placed two instances of each in every sensor block. For Path-RO sensors, we have replicated ten longest paths by STA, and randomly chose 70 other paths from top 20% paths.

2.2.4 Simulation Results

Before our proposed optimization algorithm is tested, we first present results from SVM-only classification. As previously explained, this classification makes use of a

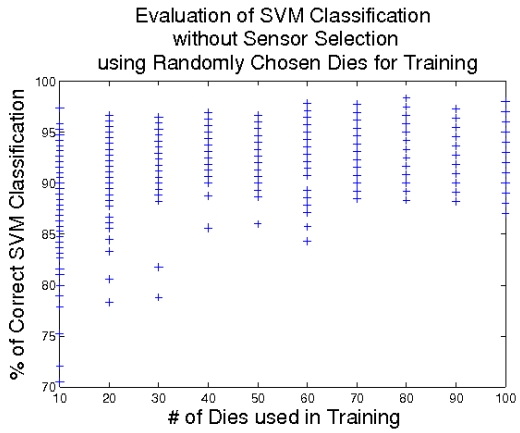


Fig. 2.4: Accuracy of SVM classification without optimizing sensor selection.

simple linear kernel function and only two bins. Obviously, number and choice of training dies leave an impact on classification accuracy. In this experiment, we randomly selected training dies and repeated classification 200 times to show distribution of classification accuracy. 5% to 50% percent training-die scenarios are investigated, and their result shown in Figure 2.4. It can be gathered from the figure that although accuracy could be quite high sometimes, it can also get quite low at some other occasions. The distribution at each given amount-of-training-die scenario is most likely due to correlation among *die-to-die* variations, which could make some dies better at being training dies. However, since inter-die correlation is not the focus of this work, optimization in training die selection is not investigated. Another observation worth mentioning is the classification accuracy doesn't increase much as more training dies are used, hinting at possibility of information saturation. Judging from the SVM method, this could probably be an indication of outlier sensors preventing better classification.

We performed proposed GA optimization to a 10-die-training SVM classifica-

tion. Figure 2.5 shows how SVM classification evolves as sensor selection is optimized by the proposed optimization algorithm. The x-axis shows the generation each data point belongs to, and the y-axis shows the classification accuracy of each data point. The blue trace represents the best classification among each generation, red trace represents the average accuracy, and green represents the worst. In this example, a population size $S = 100$ is used, and the optimization is set to run until 300 generations or 100% accuracy is reached. $I = 1$ invading individual is introduced in each generation. When the optimization starts, the SVM used all sensors and could achieve a 90.53% classification accuracy; by the time the optimization ends, the best individual could achieve 94.74% accuracy. Although improvement is quite evident, this result proved inferior to our experiment with silicon data (to be presented in next section). This could have a couple of explanations. For one, we assumed a strict 90% correlation between paths and sensors, which might not be true for all sensors, as we have discussed. Another possible reason is the small size of IC samples we were able to access probably made the silicon result more optimistic, causing a "rounding up" effect on the measured accuracy.

Now we evaluate the capability of proposed method to improve binning. We start from the same 10 training dies and randomly choose additional dies for the proposed optimization process. The optimization results are shown in Table 2.2. In the table, GA-optimized results are compared against SVM results, assuming the same dies used for GA optimization were appended to the training dies for SVM. The results substantiated

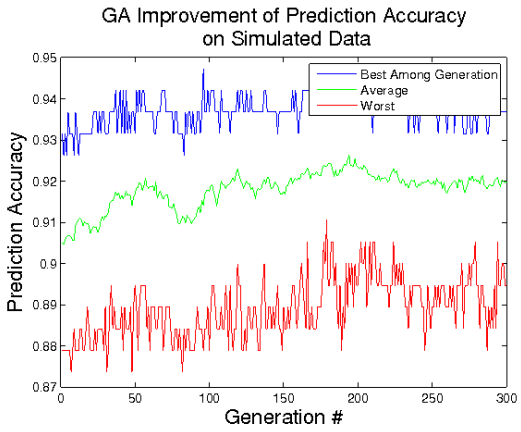


Fig. 2.5: Evolution of SVM classification accuracy under GA-based optimization of sensor selection.

Table 2.2: Comparison between optimized classification and SVM of same total training size.

Number of Training Dies	Number of Dies for Optimization	SVM Accuracy	Optimized Accuracy
10	10	92.22%	87.22%
	20	90.59%	91.18%
	30	52.50%	92.67%
	40	71.33%	92.17%
	50	91.43%	95.00%
	60	93.08%	93.08%

the improvement of the proposed method over using all sensors. Although it shows that *die-to-die* process variation does have an impact over SVM result through choice of training dies, the result also showed that with GA-optimized sensor selection the SVM could become more robust toward outlier dies as well.

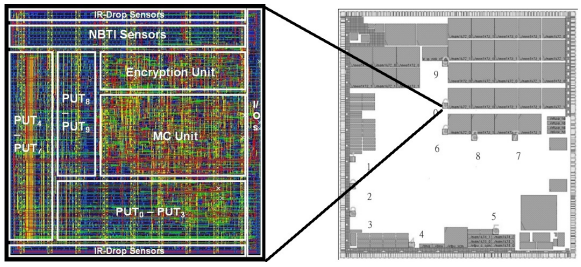


Fig. 2.6: On-chip sensor block layout (left) and their locations (right) in the fabricated SoC die. In the figure, RO sensors are denoted as “IR sensor" and Path-RO sensors as “PUT".

2.3 Experimental Result using Silicon Data

2.4 Sensor Design

The silicon data used in this section are provided by fabricated ICs as part of research work reported in [86]. These sensors were fabricated on a commercial processor IC used in automotive applications. 34 dies are measured while F_{MAX} information are available for 6 of them. Each measured die carries $J = 10$ sensor blocks at various locations, as shown in Figure 2.6. In each sensor block, $K_{\text{RO}} = 8$ different modules of RO sensors and $K_{\text{Path-RO}} = 8$ different modules of Path-RO sensors (henceforth referred to as modules) are inserted, whose respective designs are shown in Table 2.3 and Table 2.4. In total, $L = J(K_{\text{RO}} + K_{\text{Path-RO}} = 160$ sensors are available on each measured die. Data from these sensors are shown as delay figures normalized according to their individual modules, with measurement noise removed by taking average of multiple measurements.

Table 2.3: Design of 8 RO sensor modules in each sensor block.

Number	Design Description
1	300MHz Inverter Chain
2	300MHz Inverter Chain
3	300MHz Inverter Chain
4	300MHz Inverter Chain
5	350MHz Inverter Chain
6	350MHz Inverter Chain
7	400MHz Inverter Chain
8	400MHz Inverter Chain

Table 2.4: Design of 8 Path-RO sensor modules in each sensor block.

Number	Design Description
1	Path constructed with popular cells from functional design
2	Clock buffer chain with long Metal 2 wires
3	Clock buffer chain with long Metal 3 wires
4	Clock buffer chain with long Metal 5 wires
5	Clock buffer chain with wide and long Metal 4 wires
6	Clock buffer chain with wide and long Metal 5 wires
7	Small high-V_t inverter chain with alternate long and short wires
8	Large standard-V_t inverter chain with alternate long and short wires

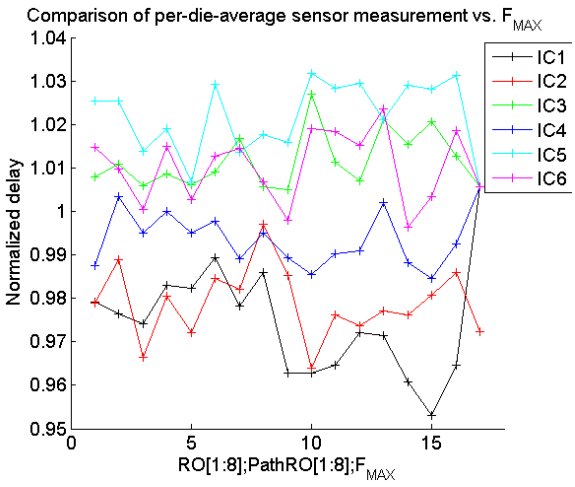


Fig. 2.7: Per-die average of normalized measured sensor delay, organized by sensor modules, compared against F_{MAX}.

2.5 Measurements

We first present a simple comparison of measurements from RO and Path-RO sensors against provided F_{MAX} classifications of the dies. As previously stated, each measured die carries $L_{\text{RO}} = JK_{\text{RO}} = 80$ RO sensors and $L_{\text{Path-RO}} = JK_{\text{Path-RO}} = 80$ Path-RO sensors. Naturally, there lies a question on how to represent each die with these data. A naive approach is to find the average of all sensors on each die and represent the die with this average. Shown in Figure 2.7 are two exemplary analyses of this approach, where average is taken of measurements from sensors in all sensor blocks of the same sensor module and compared with given F_{MAX} values. In the left figure, average is taken of measurements from sensors in all sensor blocks of the same sensor module, while in the right figure the average is taken of measurements from each sensor in a sensor block. In both figures, averages are compared with given F_{MAX} values.

Table 2.5: Classification accuracy using linear SVM.

n	Using only RO	Using only Path-RO	Using All Sensors
2	55%	45%	45%
3	83.33%	73.33%	73.33%
4	80%	80%	80%
5	80%	80%	80%

Apparently this method does not offer a reliable prediction on F_{MAX}. However, upon careful observation it is clear that some sensors offer better result than others. This naturally begs investigation on whether this can be exploited in yielding better prediction accuracy using less sensor information.

We now put the data through SVM classification as was done in Section 2.2. In this analysis, a simple linear kernel function is used. Of 6 dies whose F_{MAX} are known, a number of n dies with known F_{MAX} values are used to train the prediction model by supplying the solver with their sensor measurements as vector and their F_{MAX} as classification. The solver then finds a hyperplane that best separates vectors of one classification from those of the other. After the learning is completed, sensor measurements from other $6 - n$ dies are supplied for classification. Since only two speed bins were given for F_{MAX}, a binary classification is sufficient for the purpose of speed binning in this case. This classification result is then compared against their F_{MAX} to evaluate classification accuracy. Due to limited number of F_{MAX} values, all $\binom{6}{n}$ combinations of choices of n dies are investigated. The result of this investigation is shown in Table 2.5.

An interesting implication of the results shown in Table 2.5 is that using all sen-

sors seem to perform less well than only using some sensors, but not all of them. This can be easily understood as sensors that fell in bad neighborhood could work against the blind classification method, making it less accurate. It also verifies our earlier hypothesis that some sensors yield better result than others. Further, the observation that some sensors offer better result than using all sensors provides substantiation to the proposed technique to optimize sensor selection. Experimental evaluation of the proposed algorithm is demonstrated in the next section.

2.6 Optimization Results and Analyses on Silicon Data

We performed an experiment to find the effectiveness and validity of genetic algorithm optimization. Again, we use SVM with linear kernel function and binary classification as method of binning. Parameters of the proposed algorithms are given by $S = 100, I = 1, a_{\text{abort\_limit}} = 0.93$. Unfortunately, due to limited number of dies we are only able to present GA-optimization of classification but not improved classification. Nevertheless, optimization shown here demonstrates potential of improvement by sensor selection.

Figure 2.8 shows the evolution of the population in one exemplary run, where blue, green and red traces mark best, average and worst prediction accuracy of each generation. This run concludes at generation 232, where a best individual achieves 93.33% overall classification accuracy. From the figure we can see the improvement of best individuals has a long plateau around 85%, and appearance of individuals of high accuracy (close to 90%) is quite stochastic. Indeed, one another run has concluded

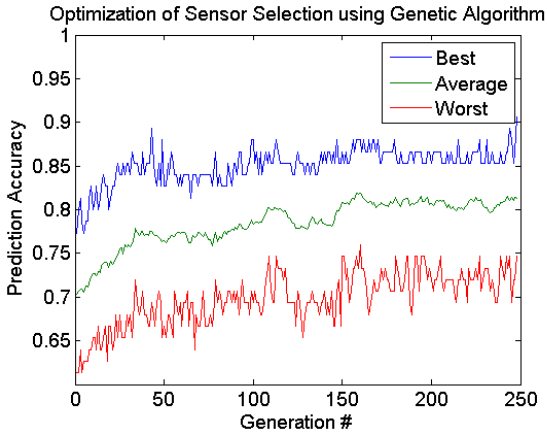


Fig. 2.8: Evolution of an exemplary process of proposed optimization algorithm.

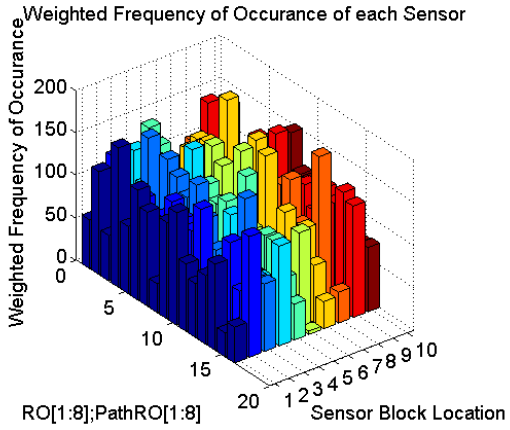


Fig. 2.9: Frequency of each sensor appearing in best accuracy sensor selection list, weighted by classification accuracy.

before generation 200. This could be explained by certain sensor choices being hard to reach from random origins, which implies possible improvement of the technique by improved invasive individual generation. Now let's look at how frequency counting would change the result. Shown in Figure 2.9 is the frequency of a sensor appearing in a best individual calculated by the sum of best individuals from all generations weighted by their respective accuracy. This figure demonstrates with statistical certainty of each sensor's suitability in performing speed-binning. From the figure we can see in 232

generations the difference of best and worst sensor in each sensor module or each sensor block location is well over 100. This result supports our previous observation that choice of sensors play a major role in speed-binning accuracy. Indeed, if we generate a sensor selection based on sensors of over 130 frequency in Figure 2.9, and then generate another selection based on sensors of less than 50 frequency, the former selection would produce 100% classification accuracy while the latter selection only giving 56%. This shows another potential approach to further improve the sensor selection process which could overcome aforementioned plateauing problem of the genetic algorithm.

2.7 Summary

We have observed that the existing mathematical-modeling based approach is not fast and accurate enough to guarantee perfect placement of on-chip speed sensors, leaving a certain amount of them detrimental to any attempt to perform speed binning with their information. Therefore, an optimization algorithm could prove useful. Based on these observations, we presented a genetic algorithm based optimization approach to improve the accuracy of on-chip speed sensor based speed binning. To verify the validity and evaluate the effectiveness of the proposed approach, we have performed an SPICE-based simulation, as well as an optimization based on silicon data collected from on-chip speed sensors fabricated in an industrial IC. In both experiments, optimized sensor selection is shown to be able to improve classification accuracy of a SVM-based blind classification. Further, the simulation result shows proposed method could improve

classification accuracy. Improved on-chip speed sensor based binning accuracy can leave less work to be done by time consuming functional test, and make sensor based speed binning a more advantageous tool during manufacture testing. The exemplary results substantiated the previous observation and demonstrated that the approach has potential of further surpassing the performance of genetic algorithm alone.

Chapter 3

CONCURRENT APPLICATION OF MEMORY AND LOGIC TESTS

Logic and memory tests are applied sequentially for different reasons (see Figure 3.1). In a sequential test, scan chain is tested first using flush patterns, then logic blocks and memory blocks are tested in series. Typically, the memory tests are run separately from logic tests. The power consumption threshold can limit the amount of simultaneous testing that can be done. Different approaches are possible to determine how much of logic or memory can be tested in a single test session.

Logic tests are usually applied using scan-based structural tests (e.g., embedded deterministic test [97]), while typical memory test algorithms such as MARCH are used for detecting faults that are dependent on the technology node, in a built-in

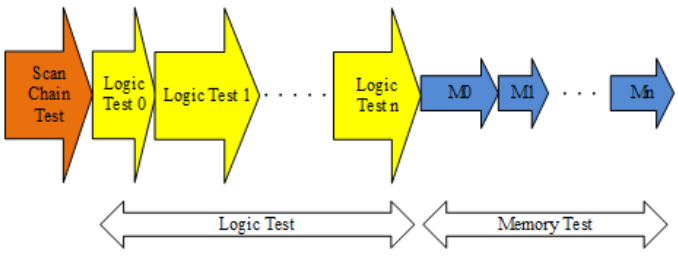


Fig. 3.1: Timing scheme of sequential application of logic and memory test.

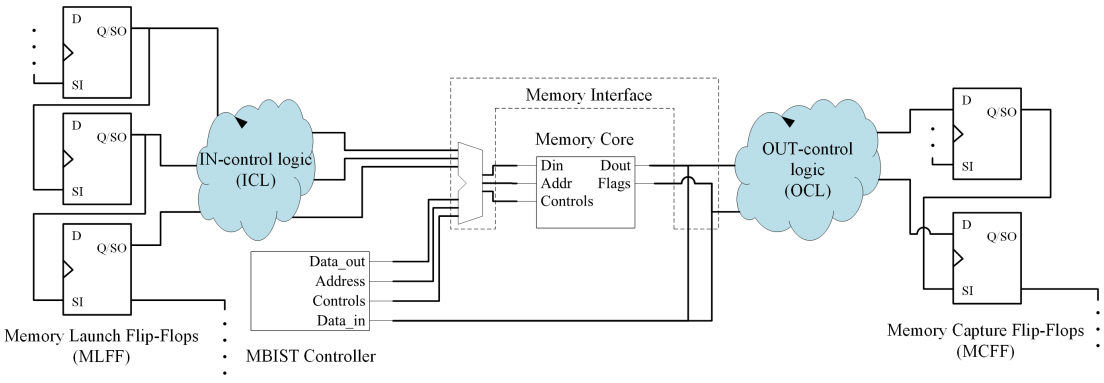


Fig. 3.2: A typical memory access structure.

self-test fashion (MBIST). The design implementation provides access to the memory using a bypass mechanism for the functional paths going into the memory, as shown in Figure 3.2. This figure shows an exemplary memory architecture and its interaction with other IP cores in a SOC. The block in the center of the figure is the memory core. It is wrapped by memory interface, where bypassing of functional and test access to the memory core is done. Further up and down stream, the functional memory access paths start and end with registers, which we term as “Memory Launch/Capture Flip-Flop", or MLFF/MCFF in the figure. They are connected to the memory interface through combinational logic, which we termed as “IN/OUT-control logic", or ICL/OCL as in the figure. A disadvantage with this current architectures is that faults on memory access paths will be left uncovered by either logic test or memory test. Since MLFFs/MCFFs will be required as launch/capture points during logic test, the functional memory access paths cannot be activated when logic test is applied since no observing/controlling flip-flops are there to capture their responses or supply them with stimulation.

Test bypass at memory interface also eliminates the possibility that these paths

be tested during memory test. These faults used to constitute a very small portion of the entire functional logic and the loss of test coverage did not constitute a serious problem. As modern designs evolve, the percentage of logic on the interface also increases [98]. As embedded memory cores become larger, so does the delay and the number of paths going through memory. For instance, in OpenSPARC T2 processor, it has been reported that the non- memory uncore components account for 51% of the total non-memory chip area [99]. Further, in a typical SOC design that involves multiple logic cores sharing access to the same memory core this situation will become even more aggravated due to additional routing prolonging the delay of paths going to the memory.

Techniques available to test these faults incur a lot of design overhead. This overhead increases drastically with increasing memory size. In a typical IEEE 1500-2005 compatible test system, logic and memory cores are wrapped and tested independently [100]. Wrapped cores could have boundary scan registers at both functional and test I/Os. For memory cores, registering functional I/Os would break up delay paths from logic core registers to memory cells. However, adding scan registers at all functional I/Os of a large core would incur significant area overhead. Moreover, registering the memory interface might not always be possible for every design.

Therefore, we propose to solve both problems with a new architecture. Instead of bypassing MBIST controller signals at memory interface, we propose to bypass them at MLFFs, and instead of collecting responses at memory core output, we propose to

collect responses at MCFFs. In doing so, we achieve the following:

1. Allow the logic test and memory test to run concurrently (Section 3.1);

2. Provide solutions to test all functional memory access paths that do not conflict with concurrent application of logic and memory test (Section 3.2);

3. Make the proposed architecture compatible to existing SOC test scheduling techniques and contribute to the overall reduction of test time without violating test power restrictions (Section 3.3).

The rest of this chapter is organized as follows. Section 3.1 discusses two alternative test schemes possible to ensure concurrent application of memory and logic tests, Section 3.2 presents three possible approaches to generate test on memory access paths, and finally Section 3.6 concludes the chapter.

3.1 Ensuring Concurrency

The key to allowing logic and memory tests to run concurrently lies in solving an inherent conflict of resources. For MLFFs, logic test requires them to fulfill the role of capture flip-flop for logic in their fan-in cone, and memory test requires it to serve as pattern launch point in order to test the path leading to memory cells. For MCFFs, logic test requires them to be launch flip-flops while memory test requires them to capture. This conflict can be resolved in two ways (see Figure 3.3). Either the conflicting flip-flops are removed from memory test path and their role in memory test is forfeited

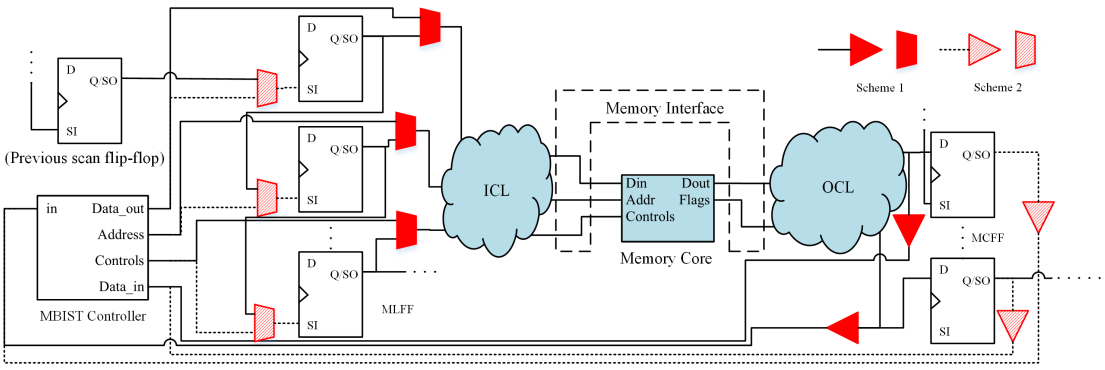


Fig. 3.3: Proposed architecture: Test Scheme 1 - Bypass to leave MLFFs and MCFFs outside of memory test path: MUX inserted at MLFF Q-pins (solid red, solid line); Test Scheme 2 - Include MLFFs and MCFFs in memory test path: MUX inserted at MLFF SI-pins (slashed red, dashed line).

(*Test Scheme 1*), or their role in logic test must be fulfilled before they can be reused in memory test (*Test Scheme 2*). It would be sub-optimal to forfeit their role in logic test since that would lead to loss in logic test coverage, hence that option is not considered. Figure 3.3 shows modified memory access structure with two possible modifications on top of the gates and blocks shown in Figure 3.2 to be able to implement schemes 1 and 2.

Test Scheme 1: Removing MLFFs and MCFFs from memory test leads to design Scheme 1, shown in Figure 3.2. As shown with solid red shapes and solid lines, multiplexers are inserted to bypass MLFFs at their Q-pins, and responses are collected at D-pins of MCFF. The advantage of this scheme is that it avoids test time overhead due to having to perform some logic test before memory test as in the other scheme, and therefore can reap the most benefit in test time reduction from concurrent application of logic and memory test. The disadvantage of this scheme is the delay induced by the

added multiplexers, buffers, and interconnects between MBIST and added multiplexers will potentially impacting memory test performance.

Test Scheme 2 The other scheme is to keep MLFFs and MCFFs in memory test path. This means logic faults that are in the fan-in cone of MLFFs and fan-out cone of MCFFs cannot be tested at the same time as memory test is conducted. Therefore two scan-test modes need to be defined: one mode is when MLFFs and MCFFs are part of the scan chain and conduct logic test, and another mode is when they are bypassed from the scan chain to take part in memory test. In the first mode, specific patterns that detect faults in fan-in cones of MLFFs and faults in fan-out cones of MCFFs are applied, therefore these flip-flops can be bypassed and used in memory test once all logic tests that require them have concluded. This two-mode scan test is implemented by inserting bypass multiplexers on the scan-in pins of MLFFs, as shown in Figure 3.2 with slashed multiplexers and dashed lines. These multiplexers feed MLFFs with MBIST controller signals when test mode changes. By bypassing at scan-in pin, this approach avoids impacting timing path, which is its advantage over the other scheme. This is also slightly more advantageous than a registered memory interface, where memory test signals are usually bypassed at D-pin, which will impact timing paths before MLFFs. The main disadvantage of Scheme 2 is it has to perform part of the logic test before memory test that can negatively impact test time reduction by concurrent logic-memory test. However, whether this happens depends on if the memory test is longer than the logic test. This is illustrated in details in Figure 3.5.

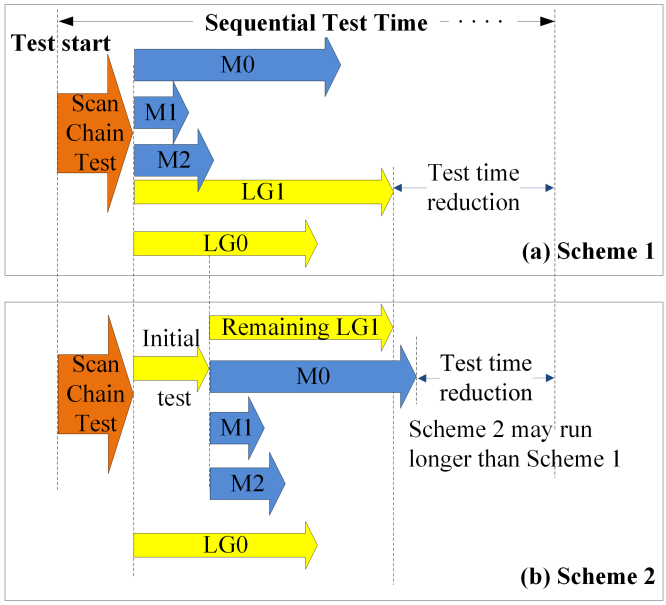


Fig. 3.4: Comparison of test time between (a) Scheme 1 (Q-pin bypass) and (b) Scheme 2 (SI-pin bypass).

Figure 3.5 shows the potential test time reduction when Schemes 1 and 2 are used, respectively. Figure 3.5(a) shows the timing diagram of Scheme 1 and the test time reduction it's able to achieve as compared to sequential application of logic and memory test. Each arrow in the figure represents time required by a test, M0 through M2 represents test time required to test each memory core, and the yellow arrows represent the test time required for logic tests LG0 and LG1 (they could be logic blocks or clock domains in a block because ATPG tools generate patterns per clock domain). Some of these tests cover circuit that interacts with the shown memories, while some do not. For this discussion, let us assume that circuit covered by LG1 interfaces with the three shown memories while circuit covered by LG0 does not. As shown in the figure, Scheme 1 starts logic and memory test at the same instant, thus making its test time re-

duction dependent on the longest time required by each of its constituting tests, in this example the LG1 test. In contrast, Figure 3.5(b) shows the timing diagram of Scheme 2. As previously discussed, unlike Scheme 1, Scheme 2 does not apply its entire logic test concurrently as the memory test. LG0 can start as soon as scan test concludes because the circuit it covers does not interact with the three shown memories. On the other hand, Part of LG1 that cannot run when Scheme 2 run the test concurrently must be applied first, which is shown in the figure as “Initial Test" for LG1 because we assumed circuit covered by LG1 interfaces with the three shown memories. Part of LG1 that can run in parallel is shown as “Remaining LG1". Each of them occupies some test time, and by pattern rearranging (elaborated in the next paragraph) the total test time of Initial Test and Remaining LG1 remains the same as the original logic test. Therefore for Scheme 2, the test time reduction depends on the sum of the test time of Initial Test and the longest among Remaining LG1 and all memory tests. As shown in Figure 3.5(b), this may result in a loss of test time reduction as compared to Scheme 1 when one of the memory tests takes longer than the Remaining LG1 as in Figure 3.5(b).

Initial Test patterns are obtained by rearranging the overall LG1 test patterns, as shown in Figure 3.6. In this shown flow, the overall LG1 patterns are rearranged using a greedy algorithm. First netlist of the logic circuit is given, as well as the faults to target during Initial test (henceforth referred to as Specified Faults). Then all detected faults by each pattern are found. After that, patterns are picked by their number of uniquely detected Specified Faults, until all Specified Faults are detected. Initial Test is then

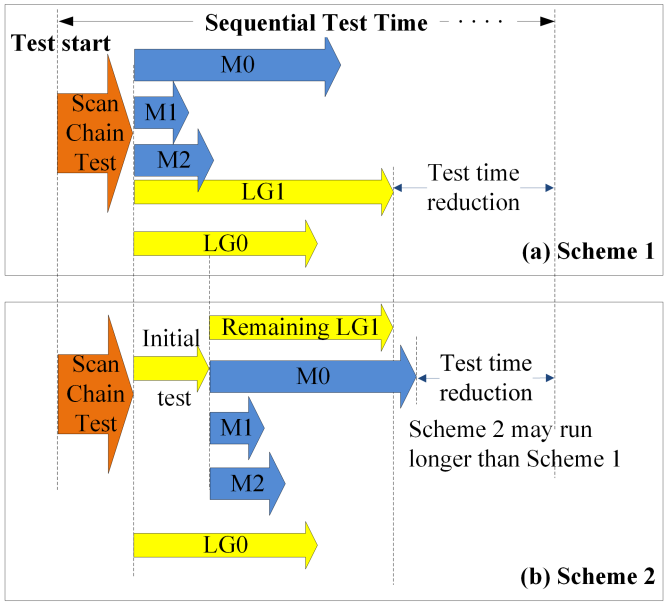


Fig. 3.5: Comparison of test time between (a) Scheme 1 (Q-pin bypass) and (b) Scheme 2 (SI-pin bypass).

formed based on these picked patterns. In addition to keeping total pattern count the same, pattern rearranging will likely result in a shorter pattern set and leave more leeway for the memory test. When memory test is much shorter than the rest of the logic test patterns, this can also be skipped or even performed on the concurrent part instead for lower computation cost or lower total pattern count. Another small disadvantage of Scheme 2 is that all participating MLFFs need to have their clock port multiplexed to run at-speed memory test clock and logic test clock when required, resulting in the need to bypass their clock pins. As shall be elaborated in the following section, application of memory test will be done by feeding address and data patterns while tying other MLFFs to constant values, so having MLFFs in different clock domains would not affect this solution much, since bits belonging to the same address or data bus also

belong to the same clock domain. If memory tests in a particular clock domain are required, bypassed clock of all affected MLFFs can be supplied with that clock instead. In both schemes, memory test responses are collected after they have passed through OUT-control logic (OCL), in order for it to be tested during memory test. This necessitates that we pass stored correct responses in MBIST controller through a replication of OUT-control logic before they are compared. Or, in the case that memory test patterns only use a limited number of data port patterns (called “data backgrounds" as in [101], where surveyed memory test sets only involve four data backgrounds: solid, checkerboard, column and row stripe), the OCL-permuted good machine response could be stored instead, likely result in a reduction in area overhead.

3.2 Testing Memory Access Paths

The schemes previously elaborated deal only with concurrent application of memory and logic test. In this section, we focus on testing memory access paths. Depending on the functional design, there are a number of possible roles for the signals registered by MLFFs. For example, in a processor design it may have been candidate signals that may be written to the memory, or base and offset address that shall pass through adder and create the final address. Even if the write signal goes through extreme permutation before being written into the memory it will have to be reversible by some other circuits present in the design, otherwise the memory will not be very useful. In this study we assume that there exists a subset of MLFFs (henceforth referred to as “essential

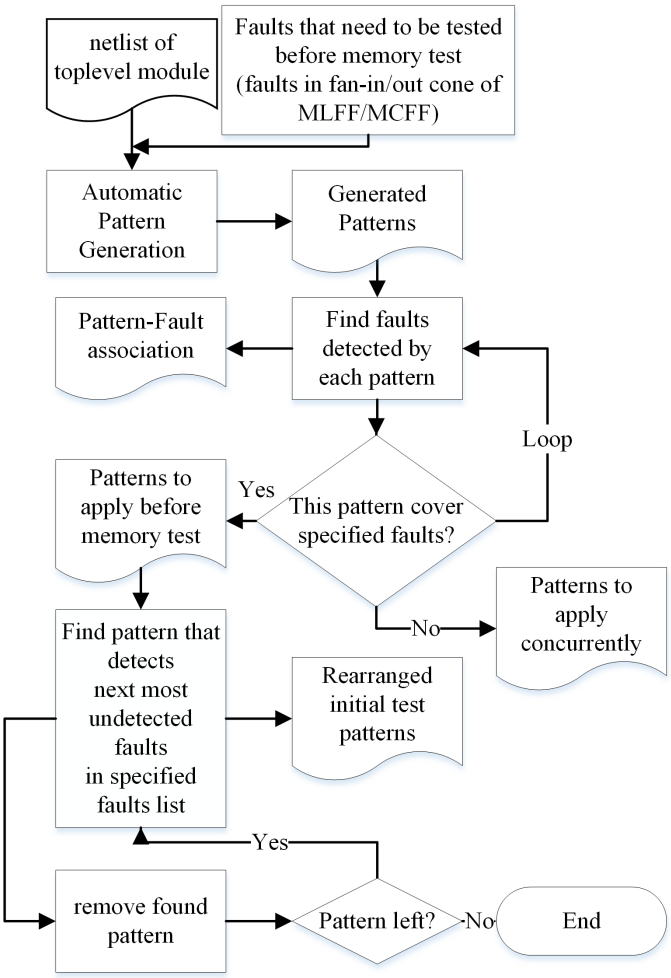


Fig. 3.6: Pattern rearranging algorithm.

MLFFs") that could allow the MBIST controller to completely cover all address and data combination of the memory by simply bypassing at these MLFFs. For example, a memory using base and offset addressing could be tested by bypassing one of the inputs to the adder with memory test address signal while tying the other with zero; and a memory driven by a memory crossbar could be tested by feeding one of the crossbar inputs with memory test data background while tying others to zero. With design information, these MLFFs required are trivial to find. Other MLFFs (henceforth referred to as "non-essential MLFFs") only need to maintain a fixed value to allow memory test stimulation to propagate through ICL, whose value can be found with automatic test pattern generation (ATPG) tools. In the case of OpenSPARC L1 and L2 caches, essential MLFFs consist of about 10% of total number of MLFFs. The number of essential MLFFs generally depends on the width and depth of the memory core, while non-essential MLFFs generally depends on how many other blocks (logic or memory) are connected to it. Testing faults in the functional memory access paths must either be performed at-speed as part of memory test, or during initial logic test. The second approach, similar to the scan-based method undertook in [102], could avoid area overhead by extra test hardware, but will likely result in longer test time. In this architecture, we investigate other methods to perform this test without shifting in patterns using the MLFF/MCFF scan chains.

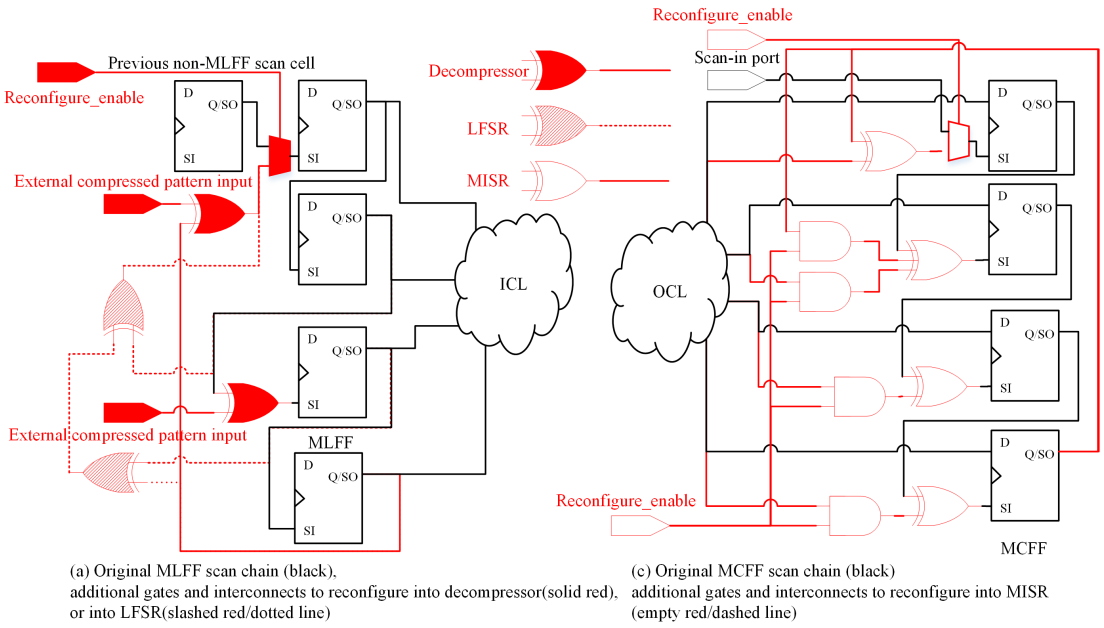
The responses of the ICL test and test patterns of the OCL test are stored with the memory cells, as was done in [102]. To achieve this, one can either use a pseudo ran-

dom test generator and bypass memory read/write control, or use deterministic pattern decompressor and organize the test patterns so that responses are read from memory after they are written to the memory. A third approach is to simply apply at-speed memory test and test functional memory access paths as a functional test.

Therefore we have three apparent approaches to test these faults: Namely pseudo random test generation, deterministic pattern decompression, and simply using memory test patterns (e.g., MARCH). The first two approaches have the obvious advantage of higher coverage, with LBIST having the potential for lower routing overhead as well as test bus occupation, and deterministic pattern decompressor is preferable for lower test time overhead.

The main disadvantage with the first two approaches is area overhead. For example, both L1 and L2 data caches of the OpenSPARC T1 processor have hundreds of MLFF registers, respectively. Constructing a dedicated LFSR and phase shifter in the conventional method will incur huge area overhead. Instead, as shown in Figure 3.7, in this evaluation we reconfigure scan chains consisted of MLFFs and MCFFs into test pattern generators (TPGs) and response generators(RGs). Figure 3.7(a) shows unaltered MLFF scan chain in black. For the purpose of reconfiguration, it is assumed that scan cells outside of MLFFs and MCFFs might exist at either end of the scan chains but not in between MLFF/MCFF scan cells. This can be achieved with scan cell constraints during design-for-test (DFT) insertion with existing commercial DFT tools. The circuits shown in solid red and slashed red/ dot red lines in Figure 3.7(a)

show modifications to reconfigure these scan chains into TPGs for deterministic and pseudo random patterns, respectively. For the pseudo random pattern approach, signals are taken from certain taps on the scan chain to provide input to the XOR gates that produces feedback as in a modular LFSR. For the deterministic approach, XORs are inserted at certain taps to reconfigure the scan chain into a ring generator. The inputs to this reconfigured ring generator are provided by SOC test bus, and when the inputs are all zero the inserted XORs will become transparent, negating the need for multiplexers to choose between ring generator and scan chain signal. In both approaches, the feedback of the TPG is fed to a multiplexer that chooses between signal from rest of scan chain (For chains purely consisted of MLFF/MCFF registers, this would be the scan-in port) and the feedback for the scan-in pin of the first scan cell in the TPG. Figure 3.7(b) shows unaltered MCFF scan chain in black, and circuitry to reconfigure it into multiple input signature register (MISR) in solid red. This figure does not show prior non-MLFF/MCFF registers exists in the chain but the assumption remains the same as in Figure 3.7(a). As shown in the figure, after adding additional XOR gates (shown in empty red) the scan chain is reconfigured into a MISR. And with the additional AND gates as well as the multiplexer at the beginning of the MISR, normal scan chain and MISR function can be selectively enabled with the “Reconfigure_enable" signal. In this method, by reusing existing scan routing, only one multiplexer, a few XOR and AND gates, and a few interconnects are required, which should not introduce a lot of area overhead. A problem might appear with the delay on interconnects between scan



(a) Original MLFF scan chain (black), additional gates and interconnects to reconfigure into decompressor(solid red), or into LFSR(slashed red/dotted line)

(c) Original MCFF scan chain (black) additional gates and interconnects to reconfigure into MISR (empty red/dashed line)

Fig. 3.7: Test generation approaches: (a) Unaltered MLFF scan chain (black) with additional circuitry to reconfigure it into decompressor for deterministic test approach (solid red/solid line) and circuitry to reconfigure it into linear feedback register (LFSR) for pseudo random pattern approach (slashed red/dotted line); (b) MCFF scan chain (black) reconfigured into response compactor (empty red/solid line) for both approaches.

cells due to distance between them. Long interconnects between scan cells may significantly impact the maximum speed of the reconfigured TPGs/RCs so much that it fails to run at the rated clock frequency. Therefore, scan chains to be modified with the proposed method must be carefully constructed to avoid this problem. This will involve proximity analyses and timing analyses for DFT insertion, as well as scan chain breaking and stitching after they are generated. Depending on the timing requirement, this will likely lead to scan chains of comparable or shorter lengths than scan chains in other parts of the logic circuits. Since scan chain lengths are usually close to typical LFSR lengths used in TPGs and MISRs, the reconfigured TPGs/RCs will have sufficiently high Hamming distance and sufficiently low aliasing probability. In the case where MLFFs/MCFFs are consisted of registers in different clock domains, registers of each clock domain will be organized into individual clock chains and tested. Test application of cores with multiple clock domains has already been covered by existing literature [103], and therefore is not elaborated in this chapter. This method also has some impact on bypass schemes. For Test Scheme 1, since this method occupies MLFFs and MCFFs when testing functional memory access paths, it cannot be applied concurrently with logic test on faults observed/controlled by MLFFs/MCFFs. In other words, pattern rearranging as in Test Scheme 2 is required for Test Scheme 1 as well to ensure maximum test time reduction. Fortunately, since usually the concurrent part of the logic test and the memory test takes much longer than test on functional memory access paths and initial test, this requirement will most likely not affect the advantageous

test time reduction of Test Scheme 1. Another topic worth discussion is the application of delay test. Since launch-off-capture (LOC) is not possible, the proposed TPG will have to use launch-off-shift schemes. This will require an scan-enable signal for The at-speed memory test approach is more a default approach than a competing one to deterministic and pseudo random tests, since the faults it covers will also be covered by the memory test of the other two approaches. In addition, the proposed TPG will have to use launch-off-shift (LOS) scheme to cover delay faults on the memory access paths. This will require a scan-enable signal able to switch at rated clock speed, which we assume can be provided by the SOC's logic test infrastructure. By skipping dedicated test on functional memory access paths, this approach could achieve better test time reduction when memory test time is crucial, as the reconfiguration method discussed earlier will occupy MLFF/MCFF as well as the memory core. Another advantage is in cases where MLFFs have unavoidable overlap with MCFFs, which precludes both TPG and compactor be constructed with reconfigured MLFF/MCFF scan chains. In this case, area overhead of the additional test hardware required by the other two approaches may be prohibitive, leaving memory test and scan test the only viable approach. The drawback of memory test approach is in fault coverage. Obviously, its fault coverage will be the lowest of all. Further, allowing memory test to cover faults means that either the memory built-in self-repair (BISR) scheme will have to tolerate faults on functional memory access paths being a possibility when a memory fault is detected, in which case the pattern count of deterministic and pseudo random test approaches can be re-

duced, reducing the difference in test time among three approach if there is any; or in a more likely scenario, unable to tell whether the fault detected by memory test is a memory fault or a fault on access paths, using memory test to test access paths leads to unsuccessful memory repair attempts and loses test time reduction.

3.3 Contribution to Test Scheduling

For SOC designs, concurrently applied test among their many cores leads to greatly appreciated test time reduction. However, test power consumption may become prohibitive for overly aggressive concurrent tests. To counter this, test scheduling algorithms [29, 104–106] are developed to manage test application schedules so that maximum possible reduction could be achieved within test power restrictions. These techniques work with the assumption that test of cores in concurrency is already in place, and provide methods to optimize their application, without concerning whether the cores it schedules tests for is logic, memory, or analog/mixed signal. In this paper, we assume the opposite: that algorithms that optimizes test scheduling of a multi-core SOC design already exists, and the proposed architecture therefore focuses on enabling the concurrent application of both logic and memory cores as well as providing solutions to test the paths in between, thereby improving both test time reduction and test coverage percentage.

3.4 Experimental Results

This section demonstrates an exemplary analysis of impact and benefit of the proposed architecture. For this purpose, OpenSPARC T1 was chosen for closeness to an industrial design, and synthesized with Synopsys 32/28nm generic library [96]. Among its many sub-modules, Level-1 and Level-2 data caches are selected to be modified with proposed method. It can be applied to either fan-in/-out registers of the memory interface or registers in the memory interface if the memory interface is registered, as the definition of MLFF and MCFF is not reliant on the existence or lack of registered memory interface; however, to demonstrate universality, we chose to base our evaluations demonstrated in this section on implementations on fan-in/-out registers of the memory interface, as this scenario is more complex and less obvious.

Area Overhead of Test Schemes Area overhead calculations for both bypass schemes on L1 and L2 Caches are shown in Table 3.1. In the table, “Original" column shows area of logic part of parental module to the data cache under investigation. “SW" column, where SW stands for scan-wrapped, has its memory interfacing ports wrapped with boundary scan chains, as is done in typical IEEE 1500-2005 compatible test systems. This column is intended to provide a reference to evaluate the result against, as scan-wrapping the memory interface is necessary for implementing concurrent test without proposed schemes. Since the proposed schemes make bypassing multiplexers on memory interface redundant, both Original and SW implementation include multiplexers from memory interface to account for area reduction as a result of the proposed

Table 3.1: Area overhead of both bypass schemes.

		L1 Data Cache			
		Original	**Scheme 1**	**Scheme 2**	**SW**
Nets		24663	27405	26976	26060
Cell Count	**Comb.**	16699	17801	17773	16699
	Seq.	5150	5150	5150	5543
	BUF/ INV	3355	3566	3566	3355
Area	**Comb.**	44863.5	49270.6	49178.1	44863.5
	Non-comb.	44212.7	44212.7	44212.7	47808.3
	Net	40296.2	41744.9	41479.1	41280.3
	Total	129372	135228	134870	133952
Overhead		-	0.098%	0.092%	0.076%
		L2 Data Cache			
		Original	**Scheme 1**	**Scheme 2**	**SW**
Nets		18845	30306	27577	36270
Cell Count	**Comb.**	10612	16244	14198	10612
	Seq.	2204	2204	2204	7387
	BUF/ INV	2388	3752	3752	2388
Area	**Comb.**	24447.4	34322.1	34133.8	24447.4
	Non-comb.	19046.1	19046.1	19046.1	66466.3
	Net	22042.0	22937.5	22813.0	38715.5
	Total	65535	76306	75993	129629
Overhead		-	0.179%	0.174%	1.068%

bypass methods. “Scheme 1" and “Scheme 2" columns show area values after application of each respective scheme. Scheme 1 bypasses MLFFs at their Q-pin, while Scheme 2 bypasses at their SI-pin. Scheme 2 also has clock pins of MLFFs/MCFFs bypassed by clock domain. In both columns, OCL is replicated, but no test generator reconfiguration is performed. Area overhead depending on different pattern generation methods is shown in Section 4. The area overhead percentage is compared against total logic area of one of eight cores in an OpenSPARC T1 processor, which is found to be 2450 by 2450 μm^2, or $6,002,500\mu$m^2.

The L2 cache is much wider than the L1 cache and this is reflected in high area

overhead. The area overhead of the implemented architecture mainly comes from the replication of OCL: if we instead assume that only limited number of data backgrounds is necessary as in [101], this can be skipped resulting in 0.052%, 0.048% for Scheme 1 and 2 on L1 cache, and -0.036%, -0.110% for Scheme 1 and 2 on L2 cache (the implemented area for L2 cache is even lower than original because the L2 data cache has fewer MLFFs than input ports, therefore nets a saving on bypass multiplexers). In addition, L1 cache has an access crossbar between MLFF and memory interface, which leads to much larger MLFF count than memory interface width. In the case of L2 cache where these two numbers are closer, the proposed methods enjoyed a much greater advantage to the alternative of scan-wrapping the memory interface.

Test Time of Test Schemes

For test time estimation, the following information is necessary: estimated memory test time and estimated time to perform the part of logic test that is conducted concurrently with the memory test. We assumed that the memory tests that are to be applied are similar to the reduced test sets described in [101]. In other words, we assumed only GalRow/GalColumn [107], Hammer [108], March RAW [109], SCAN [110], March SL [**?**], and March SS [111] tests were required. We also assume only one test voltage is required. The total test length of these test sets is 158n + 4nC [101] where n is the total number of memory entries, and C is the number of columns. On the other hand, time required to perform the concurrent part of the logic test is given by $T_{\text{logic}} = N_{\text{pat}} T_{\text{pat}} + T_{\text{shift}}$, where N_{pat} is pattern count, $T_{\text{pat}} = (T_{\text{shift}} + T_{\text{capture}})$ is the time

Table 3.2: Components for test time calculation.

		L1 Data Cache	L2 Data Cache	Execution Unit
Total Pattern Count		2104	77	6677
Initial Pattern Count		227	33	-
Estimated Initial Test Time		$228,227T$	$34,033T$	-
Estimated Remaining Logic Test Time	**Scheme 1**	$2,107,104T$	$78,077T$	$6,683,677T$
	Scheme 2	$1,878,877T$	$44,044T$	
Total Number of Entries		9×2^6	3×2^{12}	-
C		36	12	-
Estimated Memory Test Time		$173,952T$	$5,062,656T^*$	-
Estimated Total Test Time	**Scheme 1**	$2,107,104T$	$5,062,656T$	$6,683,677T$
	Scheme 2		$5,096,689T$	
Test Scheme		**Scheme 1**		**Scheme 2**
Test Time Reduction		43.10%		43.10%

required to apply each pattern, and $T_{\text{shift}} = L_{\text{chain}}T_{\text{TCK}}$ is the time required to shift-in the first pattern. In this estimation, we assume that scan chain length $L_{\text{chain}} = 100$ and scan shift clock $T_{\text{TCK}} = 10T$, where T is the functional clock period long. Assuming launch-capture cycle takes 1 functional clock period, each logic pattern would take $1001T$, and shifting in the first pattern would take $1000T$. Pattern count equals to total pattern count for Test Scheme 1, or total pattern count subtracted with number of patterns required to be applied before memory test in Test Scheme 2.

The resulting test time comparison is shown in Table 3.2. Reliable test time reduction figure of a concurrent test technique relies on having tests that are close in test time. In a more realistic scenario, the logic test time of the clock domain should be used instead, and it would be largely affected by test schedule optimization. In this

*OpenSPARC T1 L2 data cache access takes 2 clock cycles.

Table 3.3: Components for test time calculation.

		# Paths	Min Delay (ns)	Mean Delay (ns)	Max Delay (ns)
L1 Data Cache, Min Wireload	**Original**	909	0.04	0.34	0.82
	Q-pin Bypassed	931	0.09	0.40	0.88
L2 Data Cache, Wireload=7000	**Original**	1398	0.10	0.11	0.33
	Q-pin Bypassed	1398	0.24	0.26	1.12

demonstration, we take the memory test of L1 and L2 data caches as M0 and M1, and the test on logic circuits in their parental module as well as one pure logic block as LG0, LG1, and LG2. In this case LG2 dominating in test time results in the same test time reduction for both test schemes. As this is but a simple case study for demonstration of efficacy and no test scheduling algorithm is used, different test time may appear in more realistic scenarios.

Delay Analysis of Test Schemes

Bypassing with a multiplexer, inserting logic gates, and introducing extra interconnects all introduce extra delay to the timing path. This is most significant on Scheme 1, whose introduced delay impact memory access paths. Bypassing with a multiplexer introduces one same gate for each delay path start-point, so this impact largely depends on how large the delay in ICL originally was. Due to large uncore area, this would be substantially affected by routing load. During implementation, both L1 data cache and L2 data cache used multiplexer of minimum size. As shown in Table 3.3, the impact on L2 data cache is larger than L1 data cache, due to larger wireload and shorter path length.

Test time Estimation for Path Testing As elaborated earlier, there are three approaches to test ICL and OCL: decompressed deterministic patterns (DET), pseudo random number generator (PRN), or relying on memory test patterns (MT) to capture possible faults in these area. In this subsection, their impacts are investigated in terms of area overhead, test time, and fault coverage. The implementation assumes that MLFFs and MCFFs are reconfigured into test generators/response compactors, respectively. However, the L2 data cache of OpenSPARC T1 processor has a large number of MCFFs overlapping with MLFFs. Although for L2 data cache this can be solved by using another data bus instead, we implemented our test case assuming this was not possible to investigate a more complicated situation. In the case where this overlap is inevitable, only reconfiguration of MCFFs into response compactor or test pattern generator is possible, and the other role will have to be filled by additional test hardware. In this case, we assume reconfiguration is used to construct TPG instead of compactor, in order to save area required by bypass multiplexers. Test time of DET approach is done assuming that test control line can only provide N bus bits each shift clock cycle, and that compacted response is shifted out every L chain patterns, i.e. equals to compactor for logic test, which should keep aliasing chance low enough. Under these assumptions, the test time of deterministic test in this case is given by

$$T_{\text{DET,TPG}} = T_{\text{test}} + T_{\text{shiftâĹŠout}} = N_{\text{pat}}(\lceil \frac{N_{\text{bit}}}{N_{\text{bus}}} \rceil T_{\text{TCK}} + 2N|_{\text{mem,access}} T) + \lceil \frac{N_{\text{chain,MCFF}}}{N_{\text{bus}}} \rceil L_{\text{chain}} T_{\text{TCK}} \tag{3.1}$$

Where N_{pat} is the total number of patterns, N_{bit} is the width of compressed pattern, L_{chain} is the length of the scan chain, $N_{\text{chain,MCFF}}$ is number of MCFF chains, T_{CK} is the shift clock, and N_{access} is the number of functional clock cycles necessary to access the memory, here multiplied by two to account for both write and read cycles. Test time of PRNG approach is chosen to be the same as the DET approach for ease of comparison.

As a comparison, assuming scan test approach (as done in [102]) can receive test patterns from local decompressor of the core (therefore no extra test bus bit is required), its test time is given by

$$T_{\text{DET,scan}} = (N_{\text{pat}} + 1)(L_{\text{chain}} T_{\text{TCK}} + 2N_{\text{mem,access}} T) \tag{3.2}$$

Fault coverage of memory test approach is simulated by first constraining data patterns at memory interface to generate legal off- path values. Then all memory interface ports except for address bus are constrained to generated values to simulate constant off-path values during memory test. ATPG is then run with data port constrained to one of four typical memory test patterns: solid, checkerboard, column stripes and row stripes.

Table 3.4 shows fault coverage comparisons of the three approaches. Coverage shown here do not consider coverage overlap between deterministic and pseudo random test approaches with their own memory tests in order to show their own contribution; besides, it is closer to a real implementation scenario where faults on functional memory access path are to be detected as soon as possible to avoid impact on BISR

Table 3.4: Fault coverage of three test generator approaches.

Fault Coverage (%)		**DET**	**PRN**	**MT**
L1 data cache, ICL	**Stuck-at**	100	73.55	65.93
	Path-delay	100	0	0
	Transition	100	27.37	24.48
L1 data cache, OCL	**Stuck-at**	100	100	100
	Path-delay	100	0	100
	Transition	100	75.32	85.91
L2 data cache, ICL	**Stuck-at**	95.67	95.67	80.46
	Path-delay	100	0	0
	Transition	100	93.8	67.61
L2 data cache, OCL	**Stuck-at**	100	70.85	100
	Path-delay	93.8	0	63.24
	Transition	100	99.86	100

attempts. As can be seen from the table, deterministic test unsurprisingly covered most of the faults. Pseudo random test and memory tests are especially bad on detecting path delay faults, probably due to pseudo random patterns are very hard to reach the particular off-path values while memory test patterns might not have the correct off-path values constrained.

Figure 3.8 shows a comparison of test time between proposed deterministic/pseudo random test approaches and testing functional memory access paths with scanned patterns. As expected, the proposed methods greatly reduced test time required to test these paths. From the figure it can be easily seen that even if only 1 bit is spared for the TPG a test time reduction of 10X can still be expected. However, it can also be

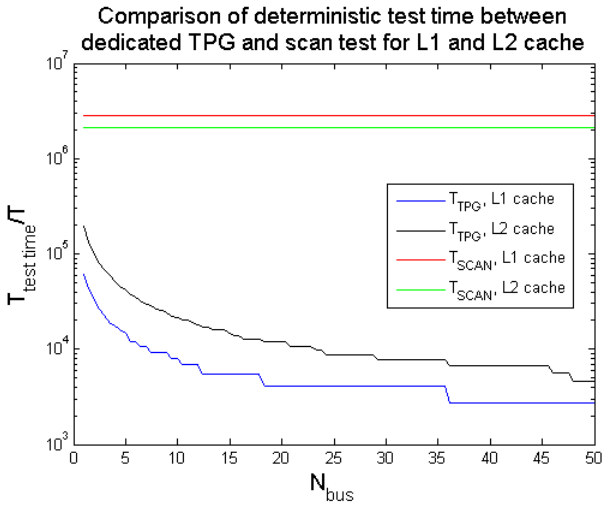


Fig. 3.8: Comparison of test time between proposed TPG approach and scan test.

seen that unlike TPG approach, the scan test approach is less sensitive to memory core scaling, making it more suitable for larger memories.

3.5 Discussion on Results

Total area overhead and test time reduction are given in Table 3.5. From the table we can see that Scheme 2 has better area overhead and comparable test time reduction. However, the area overhead difference is minimal and the test time reduction is more dependent on the actual tests lengths and test scheduling. Had this evaluation not been dominated by LG2 test time, we may see Scheme 1 achieving better test time reduction. This shows that Scheme 2 might be more advantages in cases where test times are not as close and its disadvantage of possibly impacting test time reduction can be hidden by a time consuming test.

*AO: Area overhead.
†TTR: Test time reduction.

Table 3.5: Overall area overhead and test time reduction in percentage.

Test Scheme		**1 (Q-pin bypass)**			**2 (SI-pin bypass)**		
Test Gen.		**DET**	**PRN**	**MT**	**DET**	**PRN**	**MT**
L1	**AO***	0.133	0.132	0.098	0.13	0.129	0.092
	TTR†	43.1					
	SA	100	70.2	73.59	Equals to Scheme 1		
	PDF	100	0	9.66			
	TDF	100	37.04	36.86			
L2	**AO**	0.776	0.737	0.179	0.776	0.737	0.179
	TTR	43.1					
	SA	97.81	83.38	90.14	Equals to Scheme 1		
	PDF	100	0	33.27			
	TDF	96.79	74.13	83.24			

Of the three test generation approaches, it can be seen that compressed deterministic test is better than the other two in test coverage, especially being the only approach to reliably detect path delay faults. The area difference between deterministic and pseudo random test approaches doesnâĂŹt compensate the fault coverage difference enough, although it would be preferable when the test infrastructure of the SOC design cannot afford to supply extra compressed patterns. The memory test approach is only worth considering when the other two incur prohibitive area overhead; however, due to its problem with BISR, the scan test approach as described in [102] might still be preferable if test time requirement is not too tight.

3.6 Summary

In this chapter, a novel gate-level memory test architecture is presented. This architecture supports concurrent application of both logic and memory test while allows testing to functional memory access paths at the same time. It also keeps area overhead at comparable or lower level than existing IEEE 1500-2005 compatible test architectures.

Chapter 4

A LAYOUT-DRIVEN FRAMEWORK TO EVALUATE VULNERABILITIES TO MICROPROBING ATTACKS

This chapter details our proposed framework to examine protection designs in terms of their performance against known exploits. Before presenting the framework to assess protection designs against microprobing attacks, it is essential to establish the principles of these designs. Otherwise, research could become sidetracked by objectives unnecessary or insufficient, and assessment would lack a standard for comparison.

One pitfall for the designer might be to underestimate the capability of the attacker. When considering tools available to a microprobing attack, it is important to remember that attackers capable of nano-meter scale milling is not restricted to microprobing alone. FIB itself allows circuit editing, which enables attacker to disable the whole shield by tying its detection bit to ground. Lasers can be used to inject arbitrary values to confuse protective mechanism. Indeed, both techniques have been reported successful [40]. As a result, while designs that can defeat all known attacks might not be impossible, it certainly is impractical to pursue for most devices.

Meanwhile, another myth is to underestimate the difficulty of microprobing attack. It is important to remember that attackers are likely to find a way in does not mean protection design is futile. The goal of a microprobing attack is sensitive information, and sensitivity decays with time. Information expires. Passwords are rotated. Backdoors are fixed with security updates. Even functional designs are phased out of market by new generations. Therefore, if delayed long enough, objectives of even an attacker with infinite resources can be denied.

In addition to delaying the most well-equipped attackers, it is also in the interest of the designer to deter less well-equipped attackers. This is especially true for low-cost devices such as security tokens and smartcards. This deterrence can be performed in terms of capability or information. Countermeasures vulnerable to the most cutting edge instruments might still filter out attackers that do not have access to such capabilities, and using custom designs instead of IPs reduce the risk of having a vulnerability when an IP you use is successfully attacked.

In addition to the aforementioned principles, a protection design should always be assessed with knowledge of the attack it is designed to prevent. Published microprobing attacks [40] consist of these following fundamental steps, each must be successful for the attack to succeed:

- Reverse engineer a sacrificial device to get its layout and find target wires to microprobe;
- Locate the target wires with milling tool;

- Reach the target wires without damaging target information;
- Extract target information.

Each step can have a number of alternative techniques where success with only one of them is necessary. For example, locating target wires in layout can be done by reverse engineering the design or with information from a similar design. Obfuscation can force the attacker to spend more time on this step, but if the IP is reused in another design it would allow attacker to circumvent it.

Based on principles above we propose the following framework to assess a design for vulnerability to microprobing attacks:

- For each necessary step during a microprobing attack, enumerate all known alternative techniques, the capability required by this technique, whether and how much does the design change the expected time cost on each technique;
- the protection against attackers with infinite resources is represented with the sum of techniques with the lowest time cost from each necessary step;
- the protection against less well-equipped attackers can be assessed by repeating the same process without techniques requiring unavailable capabilities.

In this framework it is possible for a particular microprobing technique to have an infinite time cost against a particular design: for example, an active shield with wires too thin for current FIB to bypass. However, the overall time cost is unlikely

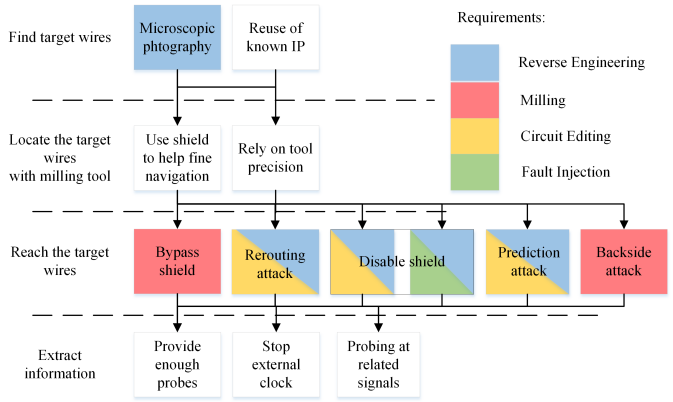


Fig. 4.1: Diagram of known microprobing techniques for assessment of design vulnerability.

to be infinite due to existence of very powerful techniques such as circuit editing: in the aforementioned example, the attacker could opt to remove the shield and disable it by fault injection or circuit editing at shield control or payload circuitry, a technique known as *disabling shield* [40]. To better illustrate this, we provide a diagram of known microprobing techniques [14, 16, 40, 41] in Figure 4.1 as an example.

Shown in Figure 4.1 is a typical flow of a microprobing attack, where each step is shown in a row and each block shows an alternative technique to complete that step. Some techniques are shaded with colors to represent the particular capability to enable that technique. *Disable shield* technique is shown with two blocks with blue triangles to show it can be completed either with circuit editing or fault injection, but in both options reverse engineering is required. Techniques in white boxes that do not have a colored alternative show possible exploits from avoidable design flaw rather than lack of protection. For example, “Use shield to help fine navigation" is possible if shield wires were not placed in 45° with regard to functional routing [40]; and if no internal

clock source is used, attacker could simply “stop external clock" to extract all information without having to use multiple probes. Based on these known microprobing techniques we may assess the protection of a few published designs, as shown in Table 4.1. An immediate observation of the shown sample attack is that bypass attack is especially expedient for the attacker and therefore most devastating for the design. It follows that this particular form of attack require more detailed attention, which is discussed in Section 4.1. From the proposed framework we can also see that layout is of central importance in both restricting the attacker’s options and increasing his time cost. If area exposed to milling can be conveniently found, it will enable designers to create antiprobing designs with better all-around resilience. For this purpose, we present an algorithm to evaluate and find *exposed area*, and present it in Section 4.2. and finally Section 4.4 concludes the chapter.

4.1 Modeling Bypass Attack

In this study, we consider a milling scenario using FIB technology as shown in Figure 4.2, where colored bars are used to represent metal wires on different routing layers. For the sake of argument, assume lowest wires in the figure are on layer n, green on

Table 4.1: Performance against known microprobing techniques of published designs

Designs	Protection against				
	Bypass Shield	Rerouting Attack	Disable Shield Backside Attack	Prediction Attack	Related Signals
Analog Shield	Weak [40]	No	No	N/A	Yes
Random Active Shield [48]	Yes	Yes	No	No	Yes
Cryptographically Secure Shield [44]	Yes	Yes	No	Yes	Yes
PAD [112]	N/A	N/A	No	N/A	No

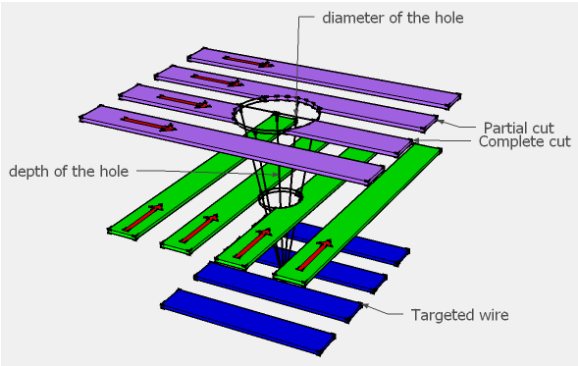


Fig. 4.2: Assumptions on FIB-based milling undertaken in this study.

layer $n + p$, top wires on layer $n + q$, and the attacker wishes to probe at one of the wires on layer n to extract sensitive information. The hollowed-out cone shown in the figure represents a hole milled with FIB equipment. In reality, a milling hole for the purpose of microprobing will probably be larger for the probe tip to maintain a reliable connection, and what Figure 4.2 showed is a best-case scenario for the attacker and worst-case scenario for the designer.

From a layout point of view, active shield designers are interested in the scenario where the attacker would make a mistake and completely cut off one metal wire at purple layer, for the purpose of detecting the attacker with a difficult-to-mistake event. It is possible that a partially cut wire may be detected by its impact on circuit timing, similar to the analog shield idea [45]; However due to reliance on afore-mentioned weakness due to reliance on parametric measurement, we have yet to see a digital active shield proposed to do this. Therefore in this study we focus on detection method based on complete cuts only.

One known exploit on active shields is to create a reroute between identified

equipotential points by circuit editing with FIB, so that the net would not become open when sections of the wires are removed [16]. This forces active shield designs to only use parallel wires of minimum spacing and widths [44]. In this case, the center of the hole least likely to result in a complete cut of a wire is in the center of the space between any two wires. Conversely, the designer need to ensure within $d_{\text{eff}} = 2W + S$ the hole is at least as deep as $T = (A/R)W$, where W and S are shield-layer metal widths and minimum wire spacing, and (A/R) is the aspect ratio of the wire. This creates a restriction of milling hole diameter d on active shield layer

$$\begin{aligned} d &\leq d_{\text{eff}} + \frac{1}{R_{\text{FIB}}} T \\ &= 2W + S + \frac{W}{R_{\text{FIB}}} \end{aligned} \tag{4.1}$$

must be satisfied or wires will be cut, where R_{FIB} is the maximum aspect ratio of FIB. If we take $W = S$ (as ITRS did [1]), Equation 4.1 further simplifies into $d \leq (3 + \frac{1}{R_{\text{FIB}}})W$.

One interesting question is whether attacker would benefit if instead of milling vertically, he mill at an angle, as shown in Figure 4.3. If we assume he was able to mill

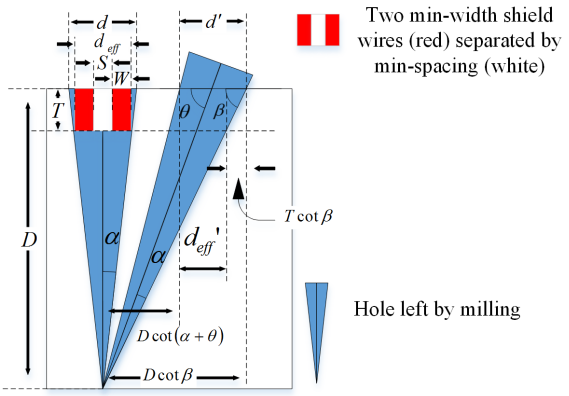


Fig. 4.3: Geometric calculations for non-perpendicular milling scenario.

at $\theta \leq \frac{1}{2}\pi$, then he will cut off wires within region d'_{eff} instead of d_{eff}

$$
\begin{aligned}
d'_{\text{eff}} &= d' - \begin{cases} T\cot\beta & , \theta \in [0, \frac{1}{2}\pi - \alpha] \\ T(\cot\beta - \cot(\theta+\alpha)) & , \theta \in [\frac{1}{2}\pi - \alpha, \frac{1}{2}\pi] \end{cases} \\
&= \frac{\sin 2\alpha}{\sin(\theta+\alpha)\sin(\theta-\alpha)} D - \\
&\quad \begin{cases} T\cot(\theta-\alpha) & , \theta \in [0, \frac{1}{2}\pi - \alpha] \\ T(\cot(\theta-\alpha) - \cot(\theta+\alpha)) & , \theta \in [\frac{1}{2}\pi - \alpha, \frac{1}{2}\pi] \end{cases}
\end{aligned} \tag{4.2}
$$

Taking derivative of $\frac{d'_{\text{eff}}}{d_{\text{eff}}}$ and letting it equal to zero yields minimum point at

$$
\begin{aligned}
\theta_0 &= \frac{1}{2}\text{acos}\left(\frac{bc - \sqrt{b^2c^2 - (a^2+b^2)(c^2-a^2)}}{a^2+b^2}\right), \text{where} \\
a &= (2(A/R)\tan\alpha + 6)\sin 2\alpha \\
b &= 2(A/R)\tan\alpha\cos 2\alpha \\
c &= 2(A/R)\tan\alpha
\end{aligned} \tag{4.3}
$$

If we further assume $(A/R) = 2.5$ as in [113] (ITRS uses 2.34 [1]), Equation 4.3 yields the following reduction in d'_{eff} over d_{eff} shown in Table 4.2. From the table we see that by milling at approximately $68^\circ - 69^\circ$ angle the attacker can effectively reduce the diameter of area by $8 - 12\%$, making it easier to bypass the shield. Since bypassing the shield is considered a convenient and preferable approach [40], this possibility makes FIB even more lethal for shields with wide top layer wires.

Table 4.2: Maximum achievable reduction of d_{eff} by milling at an angle.

R_{FIB}	5	6	7	8	9	10
$\frac{d'_{\text{eff}}}{d_{\text{eff}}}$ (%)	92.12	90.58	89.47	88.63	87.98	87.45
θ_0 (°)	68.93	68.69	68.52	68.38	68.28	68.19

4.2 Algorithm to Find Exposed Area

First, consider the wires as shown in Figure 4.2. An active shield will need a complete cut to detect milling. Using similar calculation as in Section 4.1, complete cut will happen if center of milling exists within d_{faredge} from the far edge of the wire, where

$$d_{\text{faredge}} = \frac{(2W + S)D}{2(2W + S)R_{\text{FIB}} + (A/R)W)} \tag{4.4}$$

where d is the diameter of the hole, D is the depth of the hole, (A/R) is the aspect ratio of the shield wire metal, and $R_{\text{aspect ratio}}$ is the aspect ratio given by the FIB technology the attacker is using. The aspect ratio represents the best FIB the shield will be able to defend against.

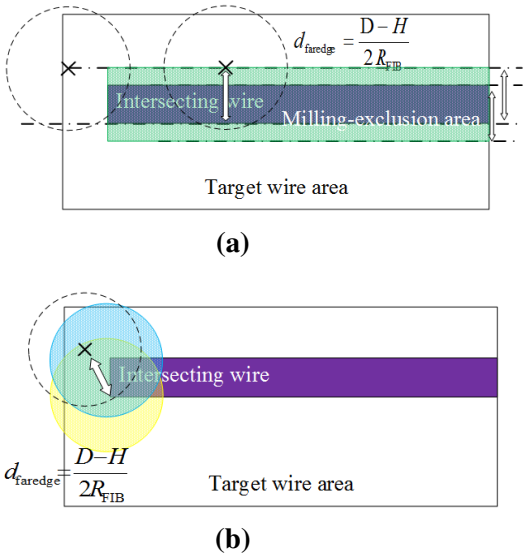


Fig. 4.4: Finding milling-exclusion area. (a) Milling-exclusion area on sides of intersecting wire; (b) Milling-exclusion area on ends of intersecting wire.

Equation 4.4 shows possibility to find the area which milling center should not fall inside. We term this area the *milling-exclusion area*. The desired *exposed area* will be its complement. Figure 4.4 shows how this area can be found for any given target wire and a wire on a higher layer capable of projecting this milling-exclusion area for it (henceforth termed as *intersecting wire*), assuming both are rectangular.

Boundaries of the milling-exclusion area can be found in two possible cases for a rectangular intersecting wire: the boundaries on the sides of the intersecting wire, and at both ends. The first kind is quite intuitive. As shown in Figure 4.4a, the center of the milling cannot fall within d_{faredge} from the farther edge of the intersecting wire, therefore boundaries of the first kind are two straight lines, each d_{faredge} away from the farther edge. The other kind of boundaries on ends are a bit more complex. Let's look at Figure 4.4b. Consider the milling hole marked by the dotted circle. For it to precisely

cut off the intersecting wire at each corner of the intersecting wire, its center must be on the edge of another circle centered at that corner, with same radius as itself. Any point within that other circle will still cut off that corner, although not necessarily the other corner. Therefore, the intersection area of both circles centered at both corners at an end constitute the complete set of milling center locations that will guarantee cut of both corners, i.e. a complete cut. Consequently, any intersecting wire rectangular in shape will project a milling-exclusion area whose shape is the union of the shape shown in Figure 4.4a and Figure 4.4b.

Now, wires in layout designs are seldom rectangular, but they are always consisted of a number of rectangular wires, usually called *shapes* by layout design tools. By iterating through each of these constituent rectangular wires, mill-exclusion areas from each intersecting wire can be projected onto each wire that may carry sensitive information and become target of microprobing attack. This process is elaborated in the pseudocode as shown in Algorithm 1.

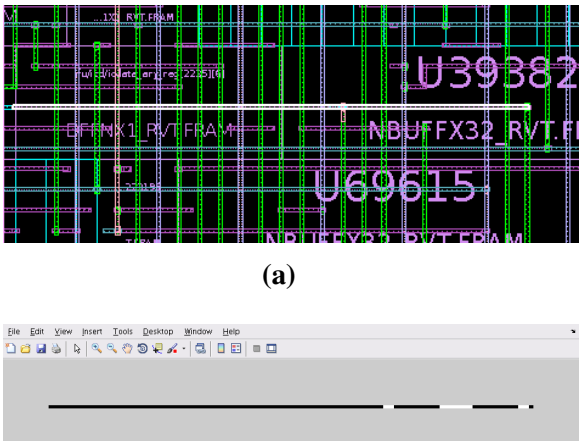
As shown in Algorithm 1, the proposed methodology starts with a set of logic nets. The algorithm first identifies their constituting wire shapes in *targeted_wire_shapes*. For each targeted wire shapes, a bitmap canvas is created, onto which mill-exclusion areas are to be projected once found. These coordinates are also given to the layout design tool to find *intersecting wire shapes* on each layer above. For each layer, a different d_{faredge} is calculated, which is then used for projections from all intersecting wire shapes on that layer. Coordinates of each intersecting wire shape are also retrieved to

```
Input: targeted_nets, precision, all_layers
Output: draw.script
begin
    targeted_wire_shapes ⇐ get_net_shapes(targeted_nets)
    N ⇐ sizeof_collection(targeted_wire_shapes)
    for (i = 1: N) do
        targeted_wire_shape ⇐ targeted_wire_shapes(i)
        canvas_size ⇐ get_sizes(get_bounding_box(targeted_wire_shape))*precision
        Print command in draw.script to create canvas in draw.script whose size equals to canvas_size
        layers_above ⇐ get_layers_above(all_layers, get_layerof(targeted_wire_shape))
        M ⇐ sizeof_collection(layers_above)
        for (j = 1: M) do
            this_layer ⇐ layers_above(j)
            d_faredge_on_thislayer ⇐ (2W+S)D / (2(2W+S)R_FIB+(A/R)W)
            intersecting_wire_shapes ⇐ get_net_shapes(targeted_nets) in
              get_bounding_box(targeted_wire_shape) on this_layer
            L ⇐ sizeof_collection(intersecting_wire_shapes)
            for (k = 1: L) do
                intersecting_wire_shape ⇐ intersecting_wire_shapes(k)
                Print command in draw.script to create projection in draw.script whose radius/widths equals
                  to d_faredge_on_thislayer
            end
        end
    end
end
```

Algorithm 1: Proposed locator algorithm for exposed area.

compute its mill-exclusion area, which is then projected to the aforementioned canvas (results shown in Figure 4.5). Projection is done by locating ends and sides of each intersecting wire shape and print the corresponding projected mill-exclusion areas. After all mill-exclusion areas are projected, running the resulting script *draw.script* can easily determine existence and area of exposed area.

For processing efficiency and adaptability, both canvas creation and projection steps are stored by the layout design tool part of the algorithm in the format of MATLAB scripts. Considerations of microprobing attacks at non-perpendicular angles can also be included with simple modifications with trigonometric functions. Another possible concern is the precision of the bitmap method. The proposed algorithm rounds toward minus infinity on borders, i.e. errs towards false positive. However, since mill-



(a)

(b)

Fig. 4.5: Exemplary results produced by proposed algorithm. (a) Exemplary targeted wire (highlighted) in layout; (b) Mill-exclusion area (black) projected on canvas of same wire.

exclusion areas are convex, overlapping of mill-exclusion areas would unlikely cause the algorithm to declare a vulnerable point when there is none either.

4.3 Evaluation Results

4.3.1 Evaluation of algorithm

In this section, we present an evaluation on the proposed algorithm. The objective is to find out how efficient can the proposed algorithm be and how much area in a typical unprotected design is exposed to microprobing attacks. For this purpose, layout of an OpenSPARC T1 core using Synopsys SAED 32nm technology library is chosen for the algorithm to inspect. For the purpose of verification, two groups of nets are selected to serve as targeted wires: first we look for long wires in the design, then we evaluate on wires on lower layers. Long wires are chosen for they resemble data buses,

which are typical targets for microprobing attacks. Wires routed in lower layers are less exposed than wires routed in higher layers and therefore forcing nets that could carry security-critical information to route on lower layers can be a sensible alternative to active shield. For this evaluation we used a resolution of 10nm, and we assume the maximum $R_{\text{FIB}} = 10$.

The long wires in this evaluation are picked based on the diagonal length of the smallest rectangle encompassing all of its shapes. All long wires thus picked have a diagonal length of at least 500μm, a number chosen to be longer than 99% of all signal route nets. On the other hand, nets routed on lower layers are restricted to not have shapes on layers higher than metal-4. This layer is picked because it is likely realistically possible if the designer tries to push his more vulnerable nets into lower layers. 5000 nets in lower-layer group of nets and 128 nets in long-wire group of nets are investigated. Their running time and exposed area are shown in Table 4.3.

Table 4.3: Evaluation results on long nets and nets on low layers

Performance	Nets on Metal-4 or Lower Layers	Long Nets
Total Number of Nets	5000	128
Total Processing Time (s)	27145	11708
Processing Time per Unit Area ($s/\mu m^2$)	5.1242	2.1297
Total Area (μm^2)	5320.58	5497.66
Exposed Area (μm^2)	4339.84	4869.21

In both cases, the proposed algorithm was able to finish processing within a few hours for a few thousand μm^2. The speed could definitely be further improved, but it seems acceptable for practical purposes, especially if we consider that the number of probes an attacker can simultaneously support is also restricted [114]. Despite only

Table 4.4: Evaluation of active shield performance

Performance \ R_{FIB}	5	6	7	8	9	10
% shield ineffective (%)	1.52	3.82	19.50	45.90	100	100
Exposed Area (μm^2)	4364.63	4507.47	4656.88	4760.98	4869.21	4869.21

having 128 nets, a much smaller number compared to 5000 nets in lower-layer group, the long-wire nets almost have the same total area. It shows a greatly reduced difficulty for the attacker to attempt microprobing at these long wires than at wires carrying signals related to it but otherwise much shorter and on lower layers. This could suggest that having those related signals might not be as mortal a sin as it first appeared. Indeed, judging from the difference in percentage of exposed area between the two groups of nets (81.57% in lower-layer group, 88.57% in long-wire group), long wires are more exposed than wires on lower layers.

4.3.2 Performance evaluation of active shield

This evaluation investigates the protection performance of active shield against FIB-based microprobing attack. Using the same layout, in this evaluation we assume that on the topmost MRDL layer, horizontal active shield wires are present. Wire width and wire spacing of this shield are both assumed to equal to 2μm, as was given by minimum wire width and minimum spacing of that layer in the technology file. The targeted nets are the same nets as the group of long nets in Subsection 4.3.1. Results are given in Table 4.4. In Table 4.4, the row “shield ineffective" indicates how many wire shapes among the total cannot benefit from the coverage of the shield at all. Note that this is based on number of shapes without regard to their area, and over 80% of

these shapes are below metal layer 4. From the results we can see even on very low R_{FIB}, the long wires are not benefiting much from the shield. This result substantiated our earlier observation that current entire-layer active shield are restricted by very wide top layer metal wire width since they cannot be placed on lower layers without making all layers above it unavailable to the design. Compared to results we see in Table 4.3, it makes sense to try using functional signal routes instead.

4.3.3 Future research directions

To our knowledge, the proposed algorithm provides the first quantifiable way to verify and evaluate microprobing vulnerabilities. This will open up a number of new opportunities in protection designs. With proposed algorithm, active shield no longer need to cover an entire layer to ensure security, therefore it can be relocated to better performing layers to improve FIB aspect ratio it can protect against. Weak links in the design, such as control and payload wires, could be buried with functional signal routes and made more resilient to attacks. Covering with multiple signal routes leads to greatly elevated requirement of reverse engineering and consequently time cost for the attacker, since he has to ensure the information gained is unspoiled and has no way to verify it. This approach can be promising if used in conjunction with anti-reverse engineering designs, as the latter greatly increase time cost in reverse engineering [115]. This could also allow protection to designs too tight in cost margin or number of layers to afford an entire layer for active shield. For this purpose, more layout-based tools could be

developed to identify security critical nets, find functional nets most suitable to serve as intersecting wire shapes, exploit faster probing assessment metrics that can integrate into existing layout optimization flow, etc.

4.4 Summary

Methods to reinforce IC in security critical applications against microprobing attacks under active research interest are plagued with high cost, weaknesses that could be exploited by attackers, and incompatibility to technologies with few layers. In this chapter, we present a layout-driven framework to assess designs for vulnerabilities to microprobing attacks. Based on design principles and assessment metrics we have established, We presented an algorithm to analyze layout designs for potential vulnerabilities to microprobing attacks. We expect it to serve as a basis for future methodologies to protect against microprobing attacks that are more effective, require lower hardware cost and applicable to a wider variety of ICs.

Chapter 5

OBFUSCATED BUILT-IN SELF-AUTHENTICATION (OBISA)

Split manufacturing is designed to stop IP piracy and IC cloning, but it fails at preventing untargeted hardware Trojan insertion and incurs significant overheads when high level of security is demanded. Built-in self-authentication (BISA) is a low cost technique for preventing and detecting hardware Trojan insertion, but is vulnerable to IP piracy, IC cloning or redesign attacks, especially on original circuitry. In this chapter, we propose an obfuscated built-in self-authentication (OBISA) technique that combines and optimizes both techniques so that they complement and improve security against both vulnerabilities, while at the same time minimizing design overheads to the extent that the proposed method does not incur prohibitive cost for designs of industrial-level sophistication. Our evaluation on AES and DES cores shows that the proposed technique can reach security levels more than two times higher, satisfy all existing layout-based security metrics, while reducing overheads from hundreds of percents to less than 13% in power, less than 5% in delay, and zero percent in area, as compared to best reported performance in existing techniques.

Changing economic trends have resulted in a globalized integrated circuit (IC)

supply chain. It is no longer economically feasible for most IC producers to own foundries and fabricate ICs in-house. For the majority of the industry, fabrication is now being performed by contracted foundries and outside the control of original intellectual property (IP) owners. IP owners enjoy reduced cost and state-of-art fabrication technologies in off-shore fabrication, at the cost of reduced control and therefore reduced trust in the manufacturing process. This has raised serious concerns on whether trust between an IP owner and such fabs can be established [49]. An untrusted foundry with malicious intent could conduct a number of attacks including IP piracy [50], IC cloning and overproduction [51, 52], and hardware Trojan insertion [53]. For off-shore fabrication to stay secure, capability of the IP owner to prevent such attacks must be thoroughly substantiated.

Split manufacturing has been proposed [17] to address the threat of IP piracy, cloning, and overproduction. This technique proposes that an untrusted foundry manufactures the front-end-of-line (FEOL) part of the IC, and then ships it to a trusted foundry to deposit back-end-of-line (BEOL) part onto it (see example in Figure 5.1). By this arrangement, the untrusted foundry is denied complete information of the layout, and therefore prevented from stealing IP information, or committing attacks that require knowledge of the complete design.

Techniques against hardware Trojan insertion exist in two categories characterized by how they address the issue: The first category focuses on detecting Trojans, either by functional verification, side-channel signal analysis, or by new front-end de-

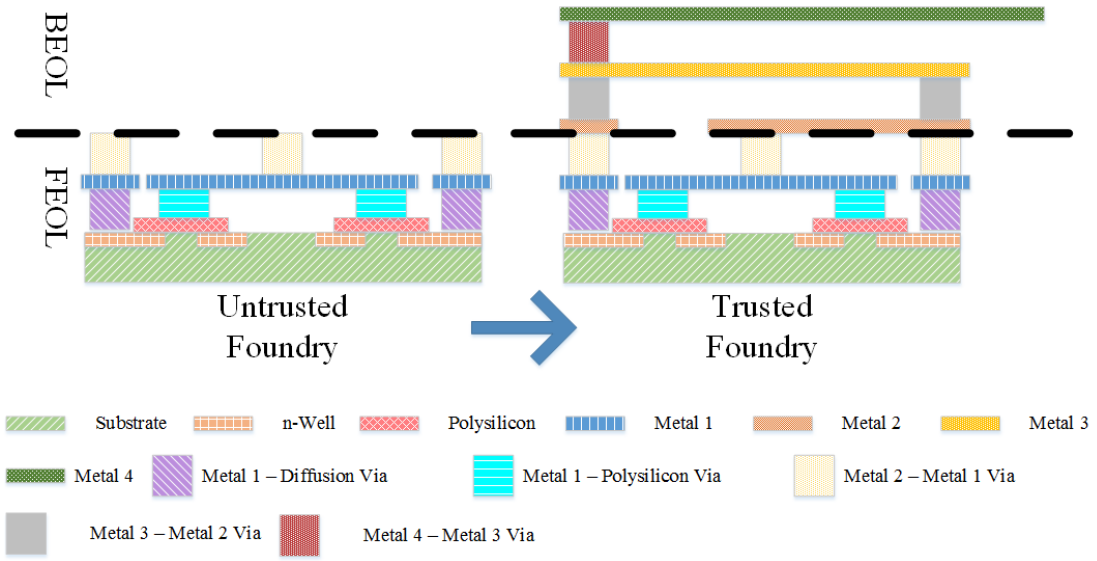


Fig. 5.1: A sample split manufacturing arrangement. It assumes split is made between Metal-2 and Metal-1 layers.

sign techniques such as design-for-trust [54–62]. Techniques in this category detect existence of hardware Trojans by generating a signature of the circuit under test (CUT), then classifying the CUT with this signature. To perform classification, they require a golden model, i.e. signature of a copy of the same circuit that is known to be free from hardware Trojans. Unfortunately, it remains doubtful whether golden models can be acquired for real world applications. In addition, process variations introduce errors in classification, especially for small, hard-to-detect Trojans. The second category, Trojan prevention techniques focuses on preventing hardware Trojans from being inserted into a design, and do not have to deal with process variation and need for golden ICs. Built-in self-authentication (BISA) is the first proposed technique to prevent hardware Trojan insertion in circuit layout and mask [65, 116]. By occupying all available spaces for Trojan insertion and detecting malicious removal through built-in self test, BISA

is able to deter hardware Trojan insertion without the requirement of golden models and free from classification errors introduced by process variation. Both split manufacturing and BISA are effective in addressing the threat they are designed to counter. However, problems remain with both techniques. To begin with, techniques against IP piracy do not usually consider the threat of hardware Trojan insertion, and neither do techniques against hardware Trojan insertion (such as BISA) consider IP theft, despite both attacks sharing the same adversary. In all likelihood, untrusted foundries will likely try all attacks in their arsenal, and IP owners will desire an overall solution that will secure his design against all of them. Therefore, a complete security solution to address the threat of untrusted foundry needs to consider all possible attacks, and both split manufacturing and BISA are limited on their own.

In this chapter, we propose a new approach to an earlier proposed technique called Obfuscated Built-In Self-Authentication (OBISA) that combines both techniques [82]. The proposed technique not only

- prevents weakness from either kind of attacks,
- is secure against attacks specific to BISA,
- is more secure in terms of proposed metrics of split manufacturing security,

but also

- has drastically reduced time complexity at generating wire lifting solutions,
- has drastically reduced design overheads of the implemented design,

- provides techniques to partition large designs to keep time complexity manageable for designs of large scale,

so that the presented technique can be expected to be implemented on industrial-size designs, while keeping overheads in both design process and design itself manageable.

The rest of the chapter is organized as follows. Section 5.2 presents the proposed OBISA technique in terms of its application flow, how it integrates with existing back-end design flow, and how each claimed contribution is realized. Section 5.3 presents experimental evaluation of the proposed OBISA technique using benchmark circuits of industrial level complexity to demonstrate its performance, as well as meeting all other published security metrics for split manufacturing in addition to the security metric it chooses to optimize. Finally, Section 5.4 concludes this chapter.

5.1 Combining BISA with Split Manufacturing Techniques

In light of respective limitations of BISA and split manufacturing techniques, it makes sense to combine them. We henceforth term the combined technique obfuscated BISA (OBISA). The most apparent advantage of the resulting technique would be the security against untargeted hardware Trojan insertion, as well as security against IP piracy and IC cloning, both of which are the primary strengths of BISA and split manufacturing. Combining with split manufacturing can also make the resulting OBISA technique secure against redesign attack, since the attacker must first identify which existing cells are connected together before designing a functionally equivalent circuit to replace

these existing cells. This will be much harder if the designer lifts the wires that connects them to BEOL, so that BISA structure becomes indistinguishable from original circuitry.

In addition, the obfuscation effect of split manufacturing can further enhance OBISA beyond protecting it against redesign attack. Combining with split manufacturing makes the threat of redesign attack by untrusted foundries much less of a problem due to security of BEOL information, and therefore reduces the necessity of using detection based anti-Trojan techniques. Since wire lifting is performed so extensive that any cell in FEOL layout is only known to connect to another cell, it makes redesign attack almost impossible and very difficult for untrusted foundry to match any single gate in the layout with the correct gate in the netlist. Split manufacturing security in OBISA could also benefit from BISA insertion, owing to improved k-security by additional cells and FEOL interconnects BISA insertion introduces to the layout. Additional cells and interconnects introduced by BISA circuitry can help to homogenize distribution of FEOL features, and proximity based attacks could also be foiled by occupying white spaces and compensating spatial distribution of gate types with BISA cells. To summarize, a combined OBISA technique could derive advantages from both split manufacturing and BISA, as well as synergy of both techniques.

Actual implementation of OBISA will depend on how it implements split manufacturing: It can be implemented simply by separating FEOL from BEOL at a certain layer without any modification to the design flow, or alternatively, wires and/or

cell placements in FEOL can be modified to avoid information leakage [75, 77]. In a previous work [82], an approach which assumes minimum computational cost is dedicated to split manufacturing (i.e., assume split manufacturing is simply implemented by separating FEOL from BEOL at a certain layer) was investigated. In the previous implementation of OBISA, obfuscation connections were added between OBISA circuits and functional circuits, and between OBISA tree-like structures, while optimization on wire lifting was kept to a minimum; In this chapter, we propose to investigate the opposite scenario, where level of security is desired (i.e., wire lifting is optimized using k-security definition). In this implementation, we dedicate more computational cost on improving split manufacturing security, as long as it does not impede industrial level of integration.

In addition to immediate advantages from combining BISA with wire lifting as was discussed in Section 5.1, the presented technique also claims the following contributions:

1. *A more efficient wire lifting algorithm*: Introduction of extra BISA cells for wire lifting to consider necessitates a more efficient wire lifting algorithm. By proposing a new set of solution constraints that are stronger than subgraph isomorphic [76], we were able to convert the wire lifting problem into a binary programming problem. In doing so, we developed a faster algorithm to find provably optimal* wire lifting solutions that has worst-case complexity of NP-

*Objective function maximizes number of edges kept unlifted, the same as the one used in [76]

complete. Experiments on Circuit432 benchmark circuits yielded 75% to 155% of edges kept at $1.74 \times 10^5 X$ to $1.06 \times 10^6 X$ speed improvements over previous wire lifting algorithm based on greedy algorithm as described in [76].

2. *A comprehensive application framework on partitioning design into manageable layout*: The previous wire lifting algorithm is limited in size of layout it can process due to its weakness in speed. The proposed fast wire lifting algorithm increased the size of layout it can realistically process by one to two orders of magnitude. In order to ensure applicability to industrial level of applications, we have investigated ways to partition designs, and proposed two approaches - one based on logic hierarchy, the other using simple geometry - that complements each other to cover all possible scenarios. By performing design partitioning, difficulty of design size on wire lifting can be divided so that it no longer scales exponentially. Implementation with said partitioning techniques proved to be successful on designs up to $385,001$ gates large.

3. *Pin-based definition of edges*: An edge in [76] was defined using its driving and driven vertices, which may not always be injective mapping, since different nets may connect the same gates in a design but through different pins. With a wire lifting algorithm with greatly reduced time complexity, we are able to define edges using their driving and driven *pins* that eliminates this problem, and at the mean time introduce additional benefit to expand applicable scenarios of wire lifting algorithm from standard-cell-only netlists to hierarchical netlists

consisted of both standard cells and logic sub-modules. This will also enable logic modules be used in addition to standard cells in wire lifting, further reduce time complexity.

4. *Cell model compensation to further improve security level*: As have been discussed earlier, the presented approach allows use of any standard cell model during OBISA insertion. This allows us to compensate the number of instantiated cells in FEOL so that wire lifting does not require restriction on standard cell models, nor does it restrict the maximum achievable security level. In doing so, we simultaneously improve security and reduce the cost in functional design timing.

5. *Secure in terms of almost all known layout-based security metrics*: In this presented approach of OBISA we perform normal performance-oriented optimizations typical of conventional design flows, and then show the resulting layout meets most known layout-based security metrics*.

5.2 Obfuscated Built-In Self-Authentication via Wire Lifting

The proposed approach to implement OBISA is characterized by a major departure from its predecessor: it uses *wire lifting* as its principal strategy to ensure split manufacturing security. A most rudimentary implementation that complies with this doctrine

*With the sole exception of *cell-level obfuscation*, which is beyond the scope of wire lifting or BISA, and can be addressed with dedicated obfuscation techniques used in unison with the presented technique.

is to simply insert BISA cells, and then solve for the optimal wire lifting solution of the resulting netlist. Such an arrangement would allow most benefits we have theorized combining both techniques would ensue. However, it is not necessarily an adequately efficient design, since it is also possible to modify BISA insertion to improve achievable security level of wire lifting. This is illustrated in an example using the same full adder circuit as was shown in Figure 1.4a.

Consider the full adder as shown in Figure 1.4a, and graph for its FEOL layout. It is apparently impossible to distinguish the two XOR gates (represented by vertices shaded in red slash) in FEOL layout. XOR gates in this full adder have a $k = 2$ security. If the same can be said for all other gate models, the FEOL layout would have $k = 2$ security. Unfortunately as we can see from the figure, it is impossible for the full adder to reach $k = 2$ no matter the lifting solution simply because it has only one OR gate. For this full adder, any further optimization of wire lifting solution will not improve its security.

There are a few ways to address this issue. In [76], only 3 to 7 gate models are allowed during design synthesis, in order to prevent rare gate models from restricting wire lifting optimization. From a designer's point of view, however, this approach seriously impacts the performance of the original circuitry in area and power. For example, restricting Circuit432 to 3 gate models almost triples total cell count (from 115 to 282) and doubles total power and area (1.7725×10^{-4} Watt, 1205.45 μm^2 to 3.4955×10^{-4} Watt, 2506.75 μm^2).

An OBISA technique that performs wire lifting does not have to submit to this restriction, because the number of instances of rare gate models can be compensated by inserting OBISA cells. An example of this advantage is shown in Figure 5.2. In this example, OBISA cells and interconnects are added, shown in dashed lines. We can see from the example how the bottleneck in the previous example - the single OR gate - is compensated with OBISA cells. In the shown wire lifting example, $k = 2$ security is reached; If we consider a more extreme solution, e.g. lifting all wires to BEOL, at maximum the layout could reach $k = 4$ security rating.

OBISA insertion does not impact timing, because they are not connected electrically to the functional circuit. It does not impact dynamic power because it is not going to be active during the design's functional mode. However, if the functional circuit is small and number of rarely-instantiated standard cell models is large, white spaces in the layout might not be sufficient. In our experiment, compensating unrestricted Circuit432 layout using OBISA insertion to at least 10 cells per model required us to lower layout utilization to 0.42, which in turn increased area to 1205.45 μm^2 and power to 1.9549×10^{-4} Watt. This becomes less of a problem when applied on larger layouts. For example, in *one_round* submodule of 256 bit AES core, utilization ratio at 0.6 would allow us to compensate to at least 209 cells per model.

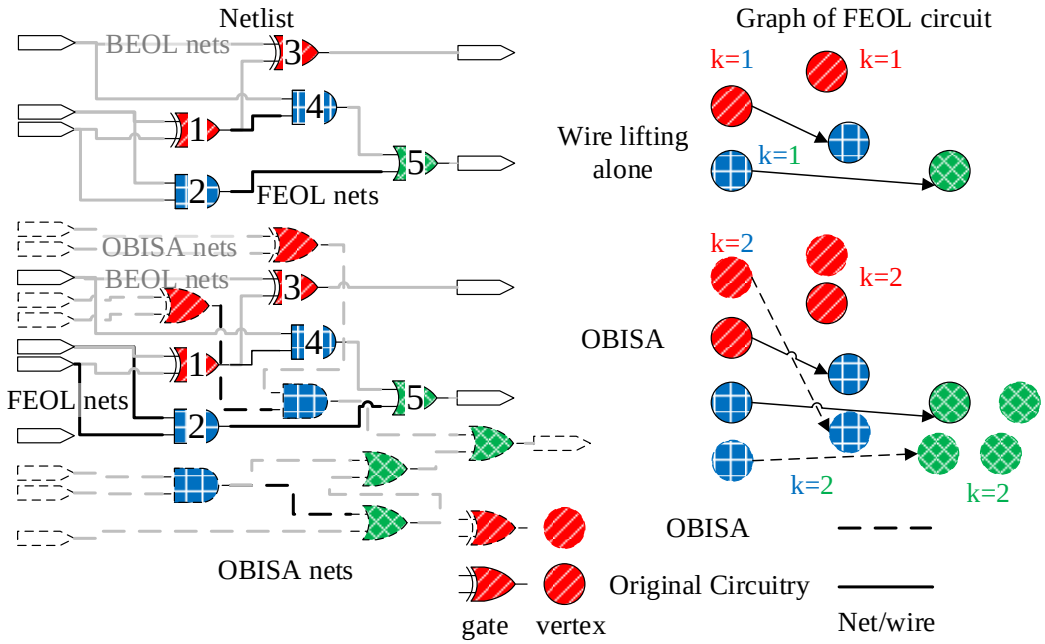


Fig. 5.2: Example: Due to OBISA insertion, wire lifting optimization on the same full adder can achieve higher k-security rating..

5.2.1 Cost of Wire Lifting

Despite the benefits of combining BISA with wire lifting, there remains a major obstacle that has rendered the approach eschewed. The obstacle is computational complexity of wire lifting algorithm, which will suffer exponentially if additional OBISA gates are added to its consideration.

We have discussed earlier in Section 1.1.4 that determining the security of any wire lifting solution has time complexity NP-hard. There are solutions that satisfy high security levels with trivial difficulty to verify: for example, it is easy to see that lifting all wires yields FEOL security level equaling to the number of the least-commonly instantiated gates; only keeping clock wires in FEOL yields security equaling to number

of least-commonly instantiated synchronous cells, etc. However, some of these easy solutions may not exist in certain designs depending on design topology; and most of these easy solutions demand a very high percentage of wires lifted to BEOL. As have been correctly observed in [76], number of wires lifted to BEOL is a cost the designer wishes to minimize, because lifting can cause overhead in timing and loss in fabrication yield due to increased difficulty of matching more vias. Therefore, a satisfactory wire lifting algorithm needs to minimize the number of wires lifted as its optimization objective.

So far, only one wire lifting algorithm based on this definition has been proposed [76], and it is based on greedy algorithm. Simply put, the algorithm (shown in Algorithm 2, henceforth referred to as "greedy wire lifting") starts from a wire lifting solution E' where all edges are assumed to have been lifted (i.e. E' equals to all edges in complete graph $E[G]$), then iteratively chooses each edge e among current E' to add back to FEOL and checks the resulting security σ of lifting solution E'. If the maximum resulting security $s = \max\{\sigma(E'\}) \geq k$ the algorithm adds its corresponding edge e_b to current solution E' and continue searching; otherwise it concludes with current solution E'. There are two problems with this approach: it is not efficient, and it is not optimal. It is not optimal because the adding one wire back to FEOL will very likely preclude at least one other wire to be added back, and therefore limiting solutions the algorithm will be able to reach. Hence, the wire lifting solution available when choosing each wire to add back will be increasingly more reliant on choices made

```
Input: All edges in graph G: E[G]
       Requested security level: k
Output: Edges to lift: E'
1  begin
2      E' ← E[G]
3      while |E'| > 0 do
4          s ← 0
5          foreach e ∈ E' do
6              E' ← E' − {e}
7              if σ(E') > s then
8                  s ← σ(E')
9                  e_b ← e
10             end
11             E' ← E' ∪ {e}
12         end
13         if s < k then
14             return E'
15         end
16         E' ← E' − {e_b}
17     end
18     return E'
19 end
```

Algorithm 2: Greedy wire lifting algorithm [76].

in earlier steps. If we choose to investigate all possible branches of the problem, the time cost will also be exponentially amplified. It is also not efficient because for each wire to be added back, security impact of adding each wire back to FEOL wires need to be determined. Even if we perform a trivial optimization by taking the first e that satisfies $\sigma(E' - \{e\}) >= k$ (since it is impossible for the algorithm to know which e that satisfies $\sigma(E' - \{e\}) >= k$ will yield better wire lifting solutions), number of trials for each wire added back is still a random variable with expectation equals to $\frac{1}{2}(n_{E'} + 1)$. If we assume the number of wires kept to be a fraction of the total number of wires - a relationship usually holds in experiments - we see the complexity of the greedy wire lifting algorithm to be exponential with regard to the total number of wires in the design.

To summarize, neither the time complexity of determining the security of each

wire lifting solution, nor that of the greedy wire lifting algorithm is polynomial to the number of edges. Unless a more efficient wire lifting algorithm is found, OBISA insertion will exponentially complicate the problem of wire lifting for exactly the same reason it aids the process.

5.2.2 Fast Wire Lifting

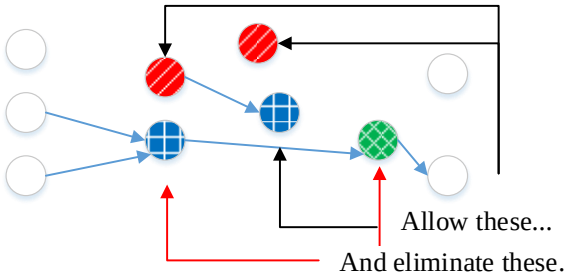
Reflecting on discussions made above we have established two facts: One, that we know with mathematical certainty that even verifying k-security of any given wire lifting solution is NP-hard; Two, that we know with almost certainty that the implementation of a wire lifting OBISA will need a more efficient wire lifting algorithm than greedy. In this subsection, we demonstrate that our proposed wire lifting OBISA technique can be efficient while satisfying those two seemingly prohibitive requirements, by providing an alternative approach to finding solutions to the wire lifting problem.

Binary Programming (BP)-based wire lifting algorithm

Since the problem of verifying k-security of *any given* wire lifting solution cannot be efficient, an efficient solution cannot consist of k-security verification of *any given* wire lifting solution. It is easy to see that the wire lifting problem can be modeled as a binary optimization problem, where each solution can be represented with a n_e-bit binary vector, where n_e is the number of edges in the complete graph. If we can find a set of constraints so that all wire lifting solutions that satisfies said constraints

(a) Uniquely identifying blue vertices in two subgraphs will need complete description of all vertices and edges in each subgraph.

(b) Two among many possible subgraphs with more than two interconnected vertices if vertices are allowed to connect to more than one edge.



(c) Constraint: only allow vertices connected to at most one edge.

Fig. 5.3: Edge types and vertex types: How to constrain the wire lifting problem to make it easier to solve.

also satisfy level k security, the problem of finding optimized wire lifting solutions becomes a Binary Programming (BP) problem, with each variable representing the choice to lift or keep one specific wire in FEOL layout, and objective function being the maximization of the sum of all variables, i.e.:

$$\begin{aligned} &\text{maximize} && \textstyle\sum_1^{n_e} x_i \\ &\text{subject to} && \boldsymbol{Ax} \leq \boldsymbol{B}, \\ &\text{where} && \boldsymbol{x} = (x_1, x_2, \ldots, x_{n_e})^\top, \\ & && \forall i \in \{1, \ldots, n_e\}, x_i \in \{0, 1\}, \end{aligned} \tag{5.1}$$

$\boldsymbol{Ax} \leq \boldsymbol{B}$ is the set of constraints we have to find.

From an implementation point of view, the BP problem albeit still complex, is of the same complexity class as the Boolean Satisfiability problem (SAT) that the greedy wire lifting algorithm uses to verify existence of a mapping between FEOL graph H and complete graph G. Further, unlike the greedy algorithm, a binary programming approach actually produces solutions that are provably optimal.

Now the problem becomes how to find such a set of constraints. Let us return to the fundamentals: k-security is about how many different vertices in complete graph can be mapped to the same vertex in FEOL graph. An apparent special case that satisfies this definition is that if s vertices of the same color (i.e., gates of the same model) in either graph are indistinguishable from each other, we may say that $k = s$ for those s vertices; or more generally, for each group of vertices in either graph that have a

common identifiable feature that makes them distinct from other vertices and indistinguishable from among themselves, the security k of each vertex in this group equals to the number of vertices in this group s. Therefore, if we can express the unique identification of each of these distinct groups of vertices in terms of variables in our BP problem, we may constrain counts of each such expression to have a sum of at least k instances, and this will suffice as the set of constraints we need.

Vertices in a DAG have only three identifiable characteristics: its own color, the edges connected to it and vertices connected to these edges, and edges as well as vertices further connected. It is easy to see the third characteristic is most likely computationally the most complex: a vertex can be connected to a large number of vertices. If we were to keep multi-connected vertices, the expression we seek will need to be able to distinguish group of interconnected vertices the vertex in question is connected to. In the example as shown in Figure 5.3a, this means we have to be able to store the topology of both subgraphs on left and on right, in order to be able to distinguish the blue vertex on the left from the blue vertex on the right. This is obviously going to be computationally complex and needs to be excluded from solution space with our constraint.

Constraints that eliminate subgraphs with more than two interconnected vertices have to do so while only being informed of the lifting decision on the edge to be decided only, otherwise due to number of possible pins of standard cells the problem will likely have to enumerate all possible combinations of relationship among edges and unlikely

trivial. This forces us to restrict each vertex to keep at most one edge as is shown in Figure 5.3c. As is shown in Figure 5.3b, any combination of more than one edge per vertex will allow existence of subgraphs with more than two interconnected vertices.

All vertices in a wire lifting solution that satisfies this constraint will be either completely isolated (i.e., all edges lifted) or one of two vertices that forms a pair based on the edge that connects them. The two-vertice-pair scenario has a very useful property for our purpose: that it is uniquely identified by the edge that connects both vertices, and the edge can be uniquely identified with only three pieces of information: the color of the driver vertex, the color of the driven vertex, and the direction. Based on this property, the set of constraints we need $\boldsymbol{Ax} \leq \boldsymbol{B}$ can be written as:

$$\begin{aligned} \boldsymbol{A}_{n_{t,a} \times n_e} \boldsymbol{x} \geq k, \quad a_{i,j} = \begin{cases} 1 & \text{edge } e_j \text{ is of type } t_i \\ 0 & \text{otherwise} \end{cases} \\ \boldsymbol{B}_{n_{t,d} \times n_e} \boldsymbol{x} \leq 0, \quad b_{i,j} = \begin{cases} 1 & \text{edge } e_j \text{ is of type } t_i \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.2}$$

$$\boldsymbol{C}_{n_v \times n_e} \boldsymbol{x} \leq 1, \quad c_{i,j} = \begin{cases} 1 & \text{edge } e_j \text{ is connected} \\ & \text{to vertex } v_i \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

$$n_c - \boldsymbol{D}_{n_r \times n_e} \boldsymbol{x} \geq k \quad d_{i,j} \text{ is the number of vertices of reference } i \text{ that edge } e_j \text{ is connected to.} \tag{5.4}$$

Where k is the requested security level; t_i is a distinct type of edge defined using its driver vertex color, its driven vertex color, and its direction; $n_{t,a}$ is the total number of allowable edge types (i.e., standard cell models of both driving and driven cells of an edge); $n_{t,d}$ is the total number of edge types to be eliminated; n_v is the total number of vertices; n_c is the total number of colors of vertices (i.e., number of standard cell models that exist in the layout). Equation 5.2 constrains edges so that each distinct type of edge either has at least k indistinguishable instances, or are completely lifted to BEOL. The latter kind of types of edges can be determined by tallying total number of edges by types in the complete graph and banning types with fewer than k instances. Equation 5.3 constrains the lifting solution to leave at most one edge per vertex. Equation 5.4 constrains vertex colors to have at least k isolated vertices (i.e., vertices with all edges lifted).

One final problem to solve is for the constraints to fit all possible scenarios. Constraints in the form as shown in Equation 5.2 and 5.4 assumes an optimal solution satisfies, or at least a solution exists that satisfies all named edge types have at least k indistinguishable instances, and each color of vertices to have at least k vertices with all edges lifted. In reality, desirable solutions might not satisfy all of these requirements. Some edge types that have more than k instances in the complete graph may have to be

all lifted to ensure all others having at least k instances, or some colors of vertices may be all connected, leaving no need to lift all edges of at least k vertices of each of these colors.

Determining whether any such scenarios is present is exactly part of the BP problem, and should not be determined before solving the problem. To adapt our constraints to these different possible scenarios, we convert affected constraints into so-called either-or constraints by introduce a few extra variables $\boldsymbol{y}$ and $\boldsymbol{z}$ to choose between the alternatives. The complete description of the BP problem therefore becomes Equation 5.5. This is the complete set of constraints for the BP-based approach of fast wire lifting.

$$
\begin{aligned}
&\text{maximize} \quad \textstyle\sum_1^{n_e} x_i \\
&\text{subject to} \\
&\qquad \boldsymbol{A}_{n_{t,a}\times n_e}\boldsymbol{x} \geq k - M\boldsymbol{y}, \\
&\qquad \boldsymbol{A}_{n_{t,a}\times n_e}\boldsymbol{x} \leq M(\boldsymbol{y} - 1), \\
&\qquad \boldsymbol{B}_{n_{t,d}\times n_e}\boldsymbol{x} \leq 0, \\
&\qquad \boldsymbol{C}_{n_v\times n_e}\boldsymbol{x} \leq 1, \\
&\qquad n_c - \boldsymbol{D}_{n_r\times n_e}\boldsymbol{x} \geq k + M(\boldsymbol{z} - 1) \\
&\qquad n_c - \boldsymbol{D}_{n_r\times n_e}\boldsymbol{x} \leq M\boldsymbol{z} \\
&\qquad \forall i, x_i, y_i, z_i \in \{0, 1\},
\end{aligned}
$$

$$
\begin{aligned}
\text{where} \quad & \boldsymbol{x} = (x_1, x_2, \ldots, x_{n_e})^\top \\
& \boldsymbol{y} = (y_1, y_2, \ldots, y_{n_{t,a}})^\top \\
& \boldsymbol{z} = (z_1, z_2, \ldots, z_{n_r})^\top \\
& a_{i,j}, b_{i,j} = \begin{cases} 1 & \text{edge } e_j \text{ is of type } t_i \\ 0 & \text{otherwise} \end{cases} \\
& c_{i,j} = \begin{cases} 1 & \text{edge } e_j \text{ is connected} \\ & \text{to vertex } v_i \\ 0 & \text{otherwise} \end{cases} \\
& d_{i,j} \text{ is the number of vertices of reference} \\
& i \text{ that edge } e_j \text{ is connected to.}
\end{aligned}
\tag{5.5}
$$

Pin-based definition of edges

Implemented with SCIP [117] solver, experimental wire lifting on Circuit432 benchmark circuit yielded 75% to 155% of edges kept at $1.74 \times 10^5 X$ to $1.06 \times 10^6 X$ speed improvement compared to previous wire lifting algorithm based on greedy algorithm as described in [76] (see Section 5.3.1 for details). This considerable improvement in processing speed opens up another possibility to one small but significant improvement upon existing k-security definitions: define edges based on *pins* instead of *cells* they are connected to.

The previous definition based on cells had a few problems that impacts wire lifting solution security, design cost, and/or difficulty of implementation in a real industrial design. First, it disregards the actual difference between pins. Typical real world stan-

dard cell libraries are unlikely designed to make, for example, different input/output pins to look alike in the layout. In a cell-based definition, two edges might both be leading from an inverter to an AND vertex, while in the netlist one wire is connected to the A pin and the other is connected to the B pin of their respective AND gate. This indicates actual number of indistinguishable wires may be much lower than the algorithm reports, which constitutes a leak of information.

Another problem is with multiple output cells, a most common example is flip-flops. Normally combinational logic gates in the standard cell library have only one output, in which case all edges sharing the same driver vertex indeed belong to the same net, which makes the possibility of two edges sharing the same driver and driven vertices slim. However, typical flip-flops offer two outputs, Q and QN, where one is the inverted signal of the other. Both pins being connected to the same cell may very well serve a legitimate purpose, and therefore can be quite common in synthesized logic modules of sufficient size. A cell-based definition will be unable to distinguish different wires in this scenario and treat all of them as the same edge.

The first problem can be overcome by obfuscating standard cell libraries, while the second by constraining the synthesizer to eliminate such topology. However, it is also obvious that such remedies will likely result in further overhead and possible yield loss. It is also possible to modify the greedy wire lifting algorithm to work with pin-based definitions, as long as exponentially increased processing time is not an issue.

Further, using pin-based definition of edges opens up the possibility of hierar-

chical wire lifting. It is already observed in [76] that depending on the function of the design, the functional circuits sometimes consist of repeating structures (in other words, repetition of the same subgraph) that can be exploited to reduce the workload of the wire lifting algorithm, e.g. multiple encryption rounds commonly present in cryptographic devices such as Advanced Encryption Standard (AES).

Since pin-based definition of edges practically allows netlists consisting of any kind of gates with unlimited number of input and output pins, it no longer matters to the wire lifting algorithm whether gates in these netlist are standard cells or logic submodules. Therefore, wire lifting can be applied on the design hierarchy that consists of multiple instantiated logic submodules as well, which likely will greatly reduce the processing time even further. Furthermore, in designs where repetition of netlist topology is not apparent, netlists can be analyzed to identify recurring identical subcircuits, which can then be replaced with logic submodules to take advantage of hierarchical wire lifting. Investigation of such optimizations is beyond the scope of this chapter, although it does make for a promising lead of future work on the problem.

5.2.3 Implementation Flow

Although the proposed BP-based approach of wire lifting greatly alleviates the time complexity of wire lifting solution generation, binary programming remains an NP-complete problem. Therefore, implementation of proposed OBISA technique needs to provide solution to two specific problems:

1. Implementing the proposed OBISA technique on a reasonably-sized layout;

2. Converting any given design to layouts of the first kind.

For the first problem, we show a layout design flow modified from the original BISA implementation flow in Section 5.2.3; For the second problem, we propose to divide the layout along logic module boundaries and apply OBISA flow on each logic modules, shown in Section 5.2.3; in corner cases where this is not realistic or efficient, we present an alternative approach where the layout is divided using geometrical boundaries, and shown in Section 5.2.3.

Implementation flow on a reasonably-sized layout

The proposed OBISA flow is shown in Figure 5.4. It shares initial steps with classic BISA but departs at BISA cell insertion. Boxes shaded with blue slashes represent procedures already present in BISA flow, while boxes shaded with red crosses represent new procedures in this approach. Instead of performing optimized BISA cell placement using dynamic programming for lower resource consumption, our need for security requires cell model compensation as well as random placement for maximal obfuscation. Cells of the rare gate models are placed before others to compensate gate model distribution. The locations of these gates are chosen randomly to diminish possible leakage of information in FEOL layout; for the same purpose, remaining white spaces are filled with BISA cells with random gate models. This can be done with filler cell insertion function provided by the layout editor. This step uses all allowable standard cell models

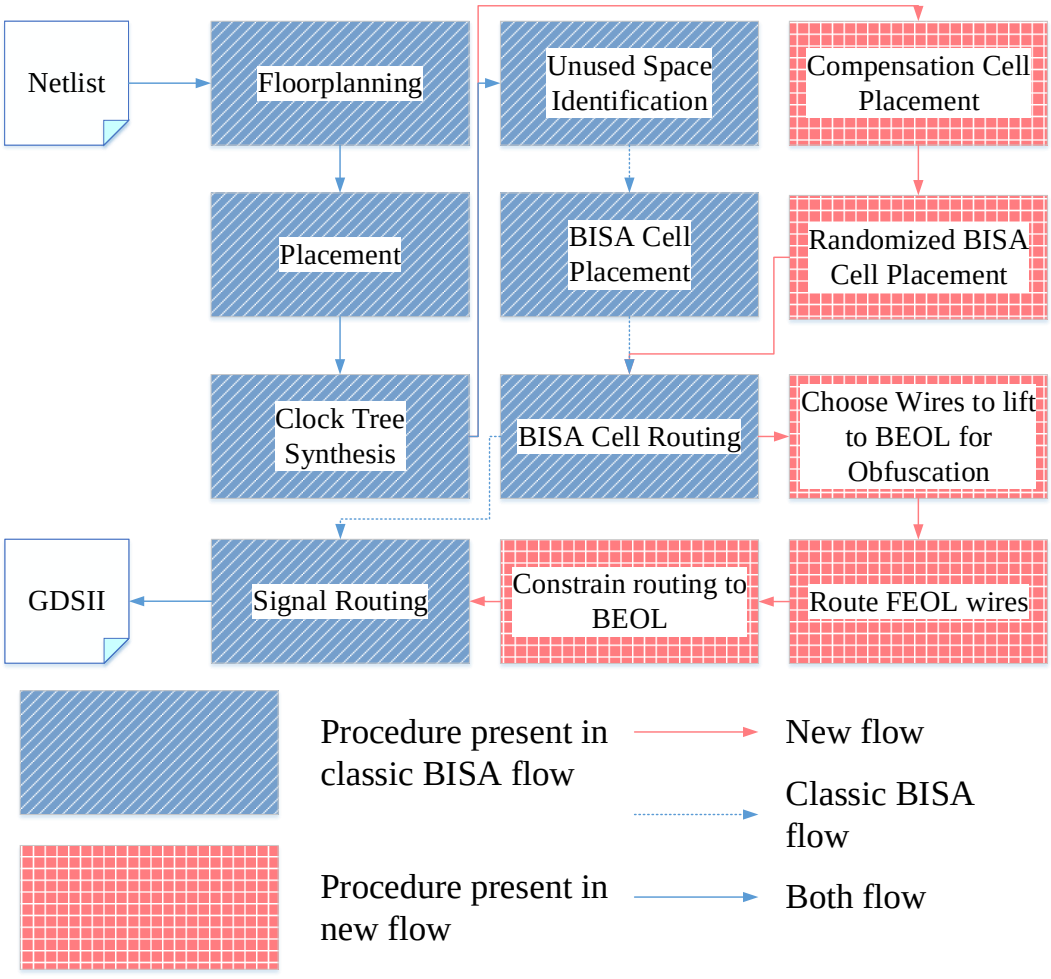
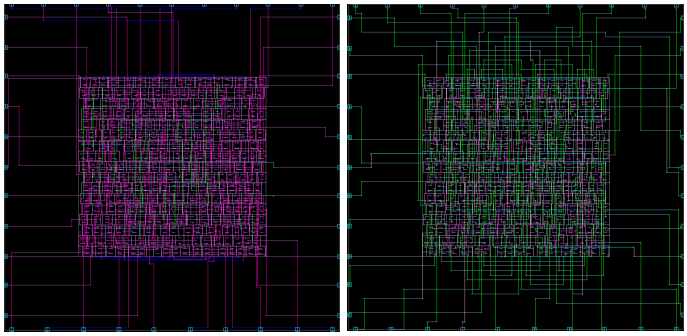


Fig. 5.4: Implementation flow of proposed OBISA technique on a reasonably-sized layout.

from the library to minimize entropy in standard cell model distribution and simplify BISA routing for the next stage. After that, classic BISA cell routing is performed. Before signal routing, an optimized wire lifting solution is found for the complete layout, using the BP-based technique we have discussed in Section 5.2.2. This can be done by writing problem description files using information from the layout editor, then calling external integer programming solver tools to find the solution.

Actual wire lifting (i.e., placing lifted wires in BEOL and kept wires in FEOL) can then be performed with the help of layout-editor. In this proposed approach, we elevate all segments of chosen wires to BEOL to eliminate the opportunity to guess direction of dangling wire segments in FEOL, leaving only vias in the FEOL. In addition to hiding direction of dangling wires by eliminating dangling wires, this approach also allows insertion of dummy vias to pins in FEOL, which will effectively hide real BEOL wires among dummy vias to further obfuscate the FEOL layout against proximity and redesign attacks. This step is done by first routing chosen wires to be kept in FEOL by restricting routing layers to FEOL layers, then routing all other wires after restricting routing layers to BEOL layers. Dummy vias can then be added randomly. One sample result of this procedure is shown in Figure 5.5. In this example, Circuit432 from ISCAS'85 is used, and split is assumed to happen between M3 and M4 layers. The layout without wire lifting shown in Figure 5.5a shows most wires in purple, which is the color assigned to M2 layer, while the layout with $k = 46$ wire lifting shown in Figure 5.5b shows most wires in green, which is the color assigned to M4 layer. Unlike similar



(a) Circuit432 layout without wire lifting optimization.

(b) Circuit432 layout using $k = 46$ wire lifting solution.

Fig. 5.5: Circuit432 layout, with and without wire lifting optimization.

layout presented in [76], layout in Figure 5.5b does not appear to have significantly more congestion than layout in Figure 5.5b, probably due to the fact that placement optimization is not done on FEOL information only and therefore does not suffer from wire length overhead likely caused by an under-optimized placement.

The rest of the design flow does not differ from conventional back-end design flow.

Hierarchical wire lifting

Designs larger than tens of thousands of gates typically need to be partitioned for wire lifting it efficient. Optimizing efficiency and effectiveness in partitioning a design for wire lifting can be a quite complicated problem in itself; however, the same partitioning problem for other purposes has existed in design practices for quite some time, and it makes sense to reuse such partitions for wire lifting as well, as it saves processing

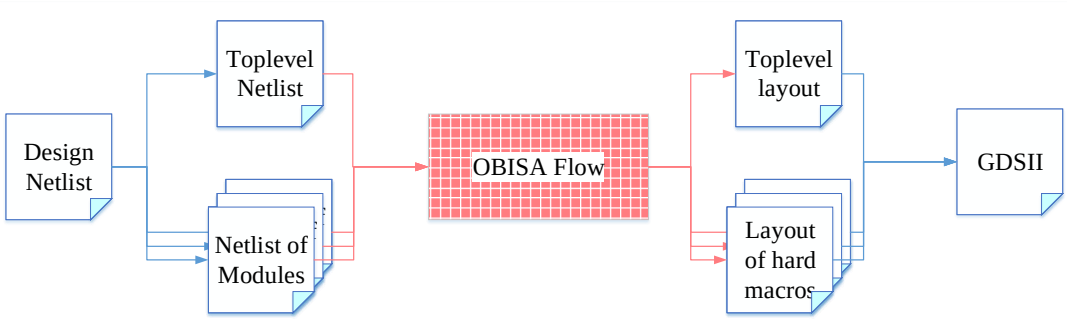


Fig. 5.6: Hierarchical wire lifting: Apply OBISA flow to each logic module, then integrate into final GDSII.

time if nothing else. Further, design partitioning for purposes such as design reuse can be expected to partition repetitive sub-circuits, which coincides with requirements of wire lifting. Therefore, in this flow we propose to reuse partitions already existing in a hierarchical layout design flow by simply performing OBISA insertion and wire lifting to each hard macro cell it instantiates; in designs that do not use hierarchical layout design, logic modules instantiated by design netlist can be used instead.

A flow diagram of the proposed hierarchical wire lifting method is shown in Figure 5.6. In order to manage the amount of computation needed for wire lifting, designs are first partitioned into hard macros (Figure 5.6). In this proposed technique, logic submodules are simply used as partitions; if recurring circuit subgraphs is discovered, this partition can be performed for each discovered subgraph.

Geometry-based partitioning

In addition to size, real industrial-scale designs pose unique challenges to efficient wire lifting which require further attention. For example, some of these challenges may

include:

- Some logic modules are instantiated multiple times, while some other modules are instantiated only once. This leads to the need of compensation in gate types and security levels among non-uniformly instantiated modules.

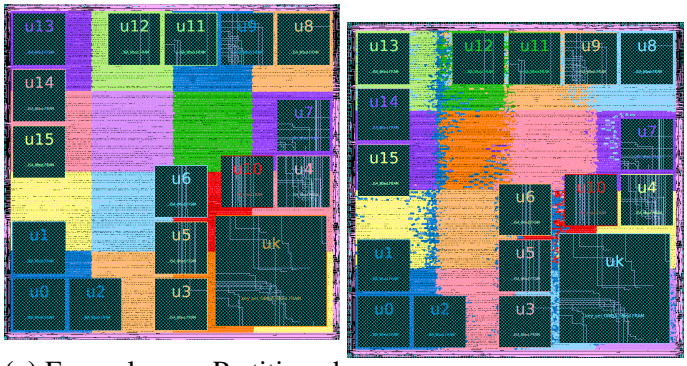
- Some logic modules are consisted of few types of gates. For example, *key_sel* module of DES core is consisted of only flip-flops and function revealing logic gates such as multiplexers. This leads to need to hide this unique composition with OBISA cells.

- Some logic modules, such as *one_round* module of AES core, can be too large to efficiently solve, for which no hierarchical division is apparent; some other logic modules may be too small to provide enough white space for OBISA gate model compensation.

Some of these challenges can be addressed with clever applications of constraints and partition rules. For such challenges the following arrangements are made in our implementation:

- Use gate types from other logic modules with more types of gates for OBISA gate type compensation on logic modules with fewer types of gates, in order to hide the *standard-cell composition bias* present in such modules.

- Use higher utilization ratio for very small modules to accommodate OBISA cells.



(a) Example: Partitioned toplevel of a DES core geometrically.

(b) Partitions updated to include fringe cells.

Fig. 5.7: Layout partitioning for simplified wire lifting.

- Assign lower security level k for more frequently instantiated modules and higher security level for modules less frequently instantiated.

- Divide large modules into partitions to reduce time complexity of finding their wire lifting solutions.

However, corner cases do exist where elegant and efficient approaches might fail. To prepare for such eventualities, we present a simple geometry-based partitioning scheme to complement hierarchical-based partitioning.

This geometric partitioning simply partitions cells in the layout into $n \times n$ rectangular regions based on their location, as shown in Figure 5.7a. Wire lifting can then be performed for each partition with updated security level divided by the number of partitions.

This method of partition leads to two more questions to be answered: One, how to determine the wire lifting solution of wires connecting cells belonging to different

partitions; Two, edges that are not rarer than the security level of the module may become rarer than that of each partition, whose lifting should be decided independent from partitions. To address the first problem, we introduce the concept of *fringe cells*, defined as cells belonging to other partitions that are connected to cells of current partition through edges. They, along with edges connecting them to cells in current partition, will be included when solving wire lifting problem of each partition. If any edge connecting a fringe cell is kept in any partition, the constraint representing that fringe cell in Equation 5.3 is changed to zero for all future partitions so that no other edge leading to that cell will be kept. We term edges featured in the second problem as *rare edges* and pulled from the consideration of each partition, and decided globally after solution of all partitions are found. To avoid solution of each partition and solution of rare edges from affecting solution of the other, constraints from Equation 5.3 and 5.4 involving rare edges are modified so that no matter the rare edges are lifted or kept, no cell will have more than one edges kept, and isolated gate counts of each gate model will be at least as large as security level of that partition. Specifically, constraints of cells connected to rare edges become

$$\begin{aligned} \boldsymbol{C'}_{n_{vre}\times n_e}\boldsymbol{x} \qquad &\leq 0, c'_{i,j} = \begin{cases} 1 & \text{rare edge } e_j \text{ is} \\ & \text{connected to vertex } v_i \\ 0 & \text{otherwise} \end{cases} \\ n_c - \boldsymbol{D}_{n_{cre}\times n_e}\boldsymbol{x} &\geq k + \boldsymbol{n}_{n_{cre}\times 1} \end{aligned} \tag{5.6}$$

Where element $d_{i,j}$ of matrix $\boldsymbol{D}_{n_{cre} \times n_e}$ is the number of vertices of reference i that edge e_j is connected to, n_{vre} (number of rows of matrix $\boldsymbol{C'}_{n_{vre} \times n_e}$) is the number of vertices connected to rare edges, n_{cre} (number of rows of matrix $\boldsymbol{D}_{n_{cre} \times n_e}$) is the number of gate models that have vertices connected to rare edges, and element n_i of vector $\boldsymbol{n}_{n_{cre} \times 1}$ is the number of vertices of gate model i that are connected to rare edges. A diagram that elaborates on the entire process of partitioned wire lifting is shown in Figure 5.8. One disadvantage of partitioned wire lifting is that fewer edges are kept, likely a result

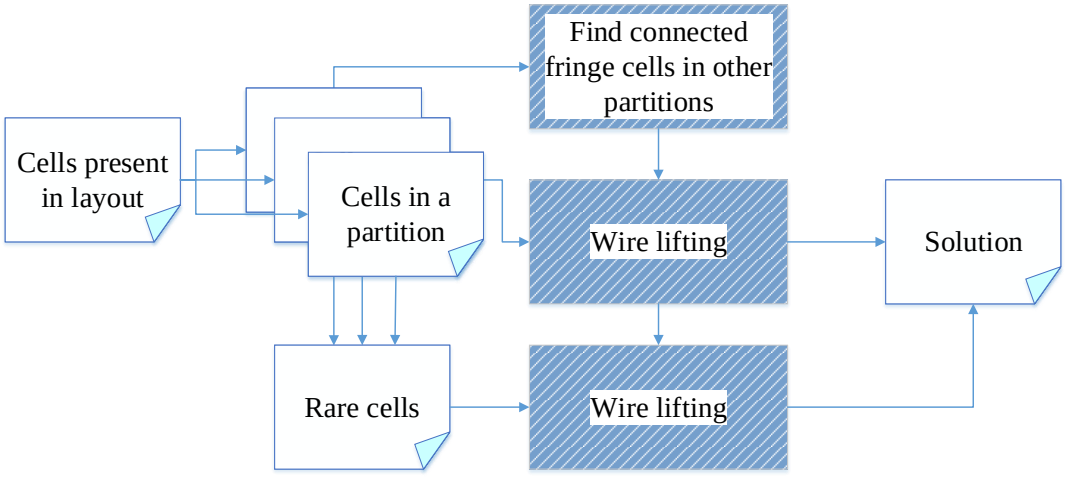


Fig. 5.8: Flow of partitioned wire lifting.

of the fact that partitioning caused types of edges being partitioned and wire lifting of each partition lacks knowledge of choices made in other partitions to keep or lift each type of edge. During our partitioned wire lifting of the *one_round* module which has over $30,000$ edges, only $2,582$ edges were kept. Nevertheless, we also find it rarely necessary to perform partitioned wire lifting in actual implementation, since it would only be required for large modules that does not have sub-modules. Toplevel

layouts do have very large vertex and edge count, but are also likely to require much higher security level due to number of logic modules it instantiates. This tends to greatly reduce computational complexity for the proposed binary programming based wire lifting algorithm since it eliminates all but a small subset of the possible solution space.

5.3 Experimental Evaluation

In this section we present experimental evaluation results to support our claims about the proposed technique, as well as explore implementation costs in terms of timing, area, power, and implementation time. Specifically, the following topics will be discussed:

1. Comparison of processing time and number of wires kept between by greedy wire lifting algorithm and proposed BP-based wire lifting algorithm;

2. Comparison of wire lifting performance between cell-based and pin-based definition of edges;

3. Evaluation of security of layout protected using proposed technique, in terms of known layout-based split manufacturing security metrics;

4. Demonstration of application on designs of industrial dimensions and evaluation of design overhead in terms of area, power, and path delays.

Results presented are collected using three benchmark circuits: Circuit432 from ISCAS'85 circuits, DES and AES from www.opencores.org. Circuit432 is used to evaluate performance of proposed technique with regard to existing greedy wire lifting approach since it was used for this purpose in [76]. The small size of this benchmark circuit poses a particular challenge to OBISA insertion, which is not enough white spaces are left when normal floorplanning density is used, forcing a trade-off between area overhead and restriction on number of standard cell models. To study this limitation, two netlists of Circuit432 benchmark circuits are synthesized: one where only three standard cell models are allowed, and one without such restriction *. DES and AES cores are used in demonstration of application on designs of large scale and evaluation of design overhead. For each synthesized netlist, three layout are created: *Ctrl* is the control group where neither OBISA insertion nor wire lifting is performed; *OBISA-Only* has OBISA cell occupying white spaces, but routed normally; *OBISA-Lifted* underwent both OBISA cell insertion and wire lifting.

Proposed BP-based wire lifting algorithm is implemented by first generating the problem formulation using a script designed as a utility called within the layout editor and then solved by calling a third-party integer linear programming solver. Since only topographical information is used in generation of problem formulation, any layout editor or even only the netlist file will suffice, and any integer programming solver can be used for the task. The presented results are collected using Synopsys IC Compiler

*For Circuit432, not restricting standard cell models lead to 12 standard cells being used.

Table 5.1: Comparison of Binary Programming (BP) and greedy algorithm-based wire lifting in terms of n_e kept and time consumption.

Method		k=46	k=32	k=20	k=16	k=12	k=8	k=4
BP	n_e kept	48	52	96	121	123	123	123
	Time (sec)	1.3	1.35	3.65	1.43	1.34	1.68	1.38
Greedy	n_e kept	≥ 26	56	101	78	152	138	165
	Time (day)	> 29	12.27	7.34	3.38	16.75	6.05	3.65
Speed improvement		> 1.92e6X	7.85e5X	1.74e5X	2.04e5X	1.08e6X	3.11e5X	2.29e5X
% of edges kept		$\leq 185\%$	93%	95%	155%	81%	89%	75%

and/or Design Compiler environment for the script, problem formulation is written in IBM CPLEX LP format, and solved with SCIP Optimization Suite [117]. MiniSat, as was used in [76], is used as the Boolean satisfiability problem (SAT) solver to determine existence of legal mapping between FEOL and complete graph used in greedy wire lifting algorithm.

5.3.1 Performance of BP-based Wire Lifting Algorithm

All comparisons for the purpose of comparing processing speed were made on Circuit432 benchmark circuit synthesized with only 3 standard cell models: AND2X1, OR2X1, and INVX1. The definition of edges used in proposed BP-based wire lifting algorithm is also restricted to cell-based definition. Such restrictions were made due to the simple fact that the greedy algorithm based wire lifting is neither efficient enough to conclude processing within reasonable time limit, nor is designed to handle pin-based edge definition, and thus will not be able to perform well for benchmark circuits with standard cells that have multiple different outputs (e.g., flip-flops). Both evaluations took place on same server computer featuring 24-core 1995.216 MHz Intel CPUs, 384 GB total memory at 1333 MHz.

From Table 5.1, we can see even under favorable circumstances, the greedy algorithm based approach is inferior in terms of processing speed by $1.74\text{e}5$ to $1.08\text{e}6$ times. Proposed BP-based approach took seconds to complete optimization that took greedy algorithm based approach days or even weeks to find. Another observation is that while the BP-based approach does not appear particularly affected by requested security level k, high security level k significantly impacts the time taken by greedy algorithm based approach, to the extent that its evaluation of $k = 46$ solution did not conclude (judging from the best case n_e it could come up with, far from concluding) even after 29 days of processing. This is likely resulting from differences both approaches approach security levels. For the BP-based approach, a higher security level means only a larger integer being used on the right side of the constraint equation, and unlikely to negatively impact its time complexity; rather, as we shall see later in our implementations on larger benchmark circuits, higher security level may even reduce the number of possible solutions and improve its solution speed. On the other hand, the greedy algorithm based approach evaluates security level of each candidate solution by enumerating k different isomorphic mappings between FEOL and the complete graph, a process that becomes exponentially more difficult as k increases.

A final row of data in Table 5.1 gives the percent of number of edges kept by the proposed BP-based approach as compared to greedy algorithm approach. The worst case performance in this metric gives us 75% of edges kept by BP-based approach as compared to greedy algorithm approach, while best case performance ranges between

155% and 185%. This result has two implications: one, that for most security levels the performance of BP-based approach in terms of edges kept is sufficient, seeing that only in three occasions it yields a worse result than 90%, and one among them was 89%; two, that the result of greedy approach in this regard is much more erratic than that of BP-based approach: in some cases, a solution for a lower security level retained less edges than another for a higher security level. This is hardly surprising, since the quality of solutions produced by BP solver is *mathematically guaranteed to be optimal* under given constraints, while the result of the greedy approach predominantly relies on the quality of its earlier choices of kept edges. To avoid picking an edge that dooms the whole optimization, the greedy algorithm based approach randomly selects among all available edges at each step; therefore, it would take many repeated searches to improve the quality of its solutions. Unfortunately, as we have seen from the processing time comparison, time necessary for repeated searches is one resource it does not have. Therefore it is very much likely, and corroborated by results in Table 5.1, that wire lifting solutions provided by the greedy approach are not optimal.

Further, as we have discussed earlier in Section 5.2.2, the cell-based definition of edges is neither efficient nor secure, which may make the few more edges kept meaningless. We shall discuss this in detail in Section 5.3.2.

Table 5.2: Comparison of security and n_e between cell-based and pin-based definition of edges.

Security level k		46	32	20	16	12	8	4
n_e kept	Cell-based	48	52	96	115	119	121	123
	Pin-based	48	50	68	105	117	120	123
Security level of cell-based		14	13	7	5	2	2	4

5.3.2 Pin-based vs. Cell-based Definition of Edges

In this sub-section we present comparisons of BP-based wire lifting results and their evaluations to demonstrate the difference between cell-based and pin-based definition of edges. Shown in Table 5.2 are number of edges kept n_e when cell-based definition and pin-based definition of edges are used, as well as evaluated level of security k using pin-based definition of edges on cell-based wire lifting results. As can be gathered from the results, not only does n_e differ when the definition of edge is changed, so does the security level. Since it is imprudent to assume the attacker is unable to distinguish pins from the layout, we must assume that cell-based definition of edges in fact leads to lower level of security than requested, as is evidented by results in Table 5.2.

Having shown the inferiority of cell-based definition of edges, we switch to pin-based definition of edges for results shown in the remainder of this section.

5.3.3 Security Evaluation with Known Layout-based Metrics

In this sub-section we present evaluations of proposed method in terms of existing layout-based security metrics for split manufacturing techniques. We are presenting results taken with the following metrics:

- Security against proximity attack is evaluated by a percentage of correct guess based on geometric proximity, as well as neighborhood connectedness ratio $C(R)$.

- Security against identification of functionality through standard cell composition bias is computed with the metric of the same name as defined in [77].

The metric of entropy in FEOL standard cells will not be evaluated as its definition as given by [77] overlaps and goes against the principle of definition of security level as number of possible mappings from FEOL graph to graph of the complete layout: The definition of entropy asserts that an FEOL layout with lower entropy is more secure, and it reaches lowest when the distribution is as unbalanced as possible, while the same situation leads to very low security level k achievable. We find it unconvincing that a highly unbalanced distribution of number of instances per standard cell model would be more secure than a balanced distribution, and it seems apparent to us that rarely instantiated cells will betray design information and help matching FEOL cells to netlist cells. Further, the concept of distribution entropy also contradicts standard cell composition bias metric for logic modules where such bias is pronounced: if a technique is applied to compensate for this bias, it will make the layout more secure according to standard cell composition bias metric, and less secure according to entropy in FEOL standard cells metric.

Table 5.3: Success rate of proximity attacks.

Circuit432	Ctrl	OBISA		Ctrl	OBISA	
		Only	Lifted		Only	Lifted
#Std-cell	3			12		
OBISA insertion	No	Yes	Yes	No	Yes	Yes
Lifted	No	No	Yes	No	No	Yes
Success	5.38%	1.99%	6.42%	0.00%	0.54%	0.27%
#Cell	220	293	293	115	309	309
#OBISA cell	0	73	73	0	194	194
Open pins	93	184	680	102	952	954
Hit pairs	5	3	28	0	1	2

key_sel, DES core	Ctrl	OBISA	
		Only	Lifted
OBISA insertion	No	Yes	Yes
Lifted	No	No	Yes
Success	0.24%	0.0003%	0.006%
#Cell	1608	3801	3801
#OBISA cell	0	2193	2193
Open pins	5461	13069	14957
Hit pairs	9	4	69

Security against proximity attack

This metric is studied by simulating a proximity attack on sample layouts and calculating percentage of correct guesses. Further, neighborhood connectedness ratio is plotted for a number of neighborhood radii. Layouts at various stages of implementation in the proposed OBISA flow were created to evaluate impact of each measure on success rate of proximity attack. In addition to Circuit432, *key_sel* module of DES core (to be elaborated in Section 5.3.4) is also shown as an example of effect on larger benchmarks. The results are shown in Table 5.3.

A first impression from the results as shown in Table 5.3 is that the number of successful guesses for each layout can be rather stochastic. This likely depends on number of nets that actually had been as short as possible, an understandable objective of placement optimization. However, the number of open pins in FEOL does indeed become greatly improved by OBISA cell insertion as well as wire lifting. This on the other hand is likely more significant than possibilities of proximity attack being

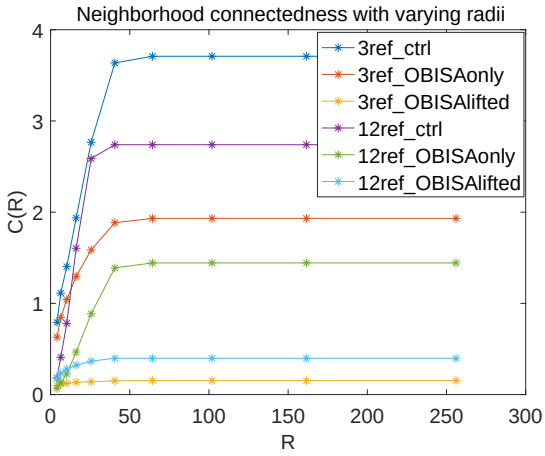


Fig. 5.9: Neighborhood connectedness ($C(R)$) curve as radii (R) increases, on Circuit432 layouts.

successful, as guess-based attack might not be always based on proximity, but all guess-based attack are universally more difficult as number of open pins in FEOL increase. If necessary, number of open pins in the FEOL can be further increased *arbitrarily* by adding dummy vias to BEOL layers that do not lead to BEOL wires.

The neighborhood connectedness ($C(R)$) plot of the same layouts investigated in Table 5.3 is shown in Figure 5.9. As can be seen from the figure, all $C(R)$ curves saturates as radii increases, but both OBISA insertion and wire lifting reduces the eventual saturated $C(R)$. As have been pointed out in [77], the lower the measure, the more "spread out" the circuit is, and less functional information is leaked, resulting in a more secure FEOL layout.

Table 5.4: Standard cell composition bias of *key_sel* and DES toplevel.

	key_sel		des	
	ctrl	OBISA	ctrl	OBISA
Flip-Flops	840	840	1144	1144
Muxes	768	768	0	984
XORs/XNORs	0	0	562	2324
#Cells	1608	3800	1761	29276
bias	6.67E-01	2.82E-01	6.53E-01	5.74E-02

Standard cell composition bias

In this evaluation, *key_sel* module and toplevel module of DES core are examined for its particular design characteristic. Being a control module, functional cells in both modules consists only of flip-flop and multiplexers. Thus, either unsecured module will be very weak in terms of Standard cell composition bias. This is compensated by inserting OBISA cells that are common in other modules of the same DES core. As shown in Table 5.4, standard cell composition bias of both modules decreased more than 50% after OBISA insertion.

5.3.4 Implementation and Overhead on Large Designs

Two particular benchmarks were used in this study: an Advanced Encryption Standard (AES) and a Data Encryption Standard (DES) core. Crypto-cores are selected on the grounds that they are more likely targeted by attacks and usually require higher security reinforcements. AES and DES are reasonably sized: after synthesis, the 256-bit AES core we have selected has $657,292$ gates, while the 64-bit DES core has $15,651$. Further, DES was also chosen in [76], and will likely serve as a good basis of compari-

son. Both designs are large enough to make lifting of a flattened netlist computationally heavy, and therefore necessitates partitioned wire lifting.

We have chosen a performance optimized triple-DES core from opencores.org to serve as benchmark in this experiment. Since the original design consists of 3 identical DES cores connected in ECB mode, it makes more sense to only implement one of these cores. Each DES core in this design is consisted of 16 instances of *crp* module and 1 instance of *key_sel* module. Similarly, the 256-bit AES core we have chosen is consisted onf 16 instances of *one_round* module, 7 instances of *expand_key_type_A_256* module, 6 instances of *expand_key_type_B_256* module, and 1 instance of *final_round* module. Finally, both DES and AES core instantiates interface cells such as flip-flops and multiplexers on their toplevel.

In our implementation we chose a security level $k = 16$ for *one_round* of AES and $k = 10$ for *crp* of DES core. These coefficients were chosen following the guideline as was discussed in Section 5.2.3, so that the overall security level can be made higher. This leads to an overall security level of $k = 208$ for AES core and $k = 160$ for DES core. To help improve efficiency, geometry-based partitioning was performed on both toplevel modules and *one_round* module of AES core. Implementation overheads in terms of power, timing delay, and area of each module are summarized in Table 5.5 and 5.6.

Both tables provide two sets of comparisons:

1. in terms of total wire length, number of OBISA cells inserted as compared to

Table 5.5: Power, timing, and area overheads of wire-lifted DES modules.

Module		key_sel			crp		
Layout		OBISA		Ctrl	OBISA		Ctrl
		Only	Lifted		Only	Lifted	
Power (W)	Internal	4.80E-03	4.25E-03	3.79E-03	1.52E-03	1.51E-03	1.50E-03
	Switching	7.48E-04	7.07E-04	5.71E-04	1.30E-03	1.18E-03	1.09E-03
	Leakage	2.88E-04	2.88E-04	2.13E-04	3.77E-05	3.77E-05	2.85E-05
	Total	5.84E-03	5.24E-03	4.58E-03	2.86E-03	2.72E-03	2.62E-03
Path Delays (ns)	Min	0.4	0.5	0.32	0.87	0.82	0.8
	Median	0.82	0.64	0.48	0.9	0.86	0.83
	Max	1.02	0.78	0.59	1.05	0.98	0.94
Total wire length(μm)		4.25E+05	3.37E+05	1.40E+05	6.86E+04	6.74E+04	2.70E+04
Area (μm^2)		67599.4			10354.2		
#Std-cell		9			28		
#Cell		4381		1608	1099		745
#OBISA cell		2773		0	354		0
Security Level k		1	160	1	1	10	1

Module		des		
Layout		OBISA		Ctrl
		Only	Lifted	
Power (W)	Internal	2.93E-03	2.94E-03	2.86E-03
	Switching	9.92E-03	9.39E-03	8.78E-03
	Leakage	1.32E-03	1.32E-03	2.41E-04
	Total	1.42E-02	1.37E-02	1.19E-02
Path Delays (ns)	Min	0.37	0.39	0.35
	Median	0.41	0.44	0.37
	Max	0.61	0.64	0.58
Total wire length(μm)		4.12E+06	3.99E+06	1.12E+06
Area (μm^2)		753423		
#Std-cell		34		
#Cell		29293		1778
#OBISA cell		27515		0
Security Level k		1	160	1

Table 5.6: Power, timing, and area overheads of wire-lifted AES modules.

Module		final_round			one_round		
Layout		OBISA		Ctrl	OBISA		Ctrl
		Only	Lifted		Only	Lifted	
Power (W)	Internal	6.70E-03	6.64E-03	6.44E-03	1.02E-02	1.01E-02	9.81E-03
	Switching	7.70E-03	7.49E-03	6.53E-03	1.16E-02	1.14E-02	1.12E-02
	Leakage	4.24E-04	4.24E-04	3.08E-04	6.40E-04	6.40E-04	6.39E-04
	Total	1.48E-02	1.46E-02	1.33E-02	2.25E-02	2.21E-02	2.16E-02
Path Delays (ns)	Min	1.37	1.32	1.2	1.83	1.79	1.71
	Median	1.42	1.37	1.24	1.92	1.88	1.79
	Max	1.7	1.62	1.42	2.46	2.28	2.2
Total wire length (μm)		1.08E+06	1.07E+06	4.90E+05	1.65E+06	1.64E+06	1.12E+06
Area (μm^2)		119882			177073		
#Std-cell		31			37		
#Cell		12377		8236	17688		11856
#OBISA cell		4141		0	5832		0
Security Level k		1	208	1	1	16	1
Module		expand_key_type_A_256			expand_key_type_B_256_OBISA		
Layout		OBISA		Ctrl	OBISA		Ctrl
		Only	Lifted		Only	Lifted	
Power (W)	Internal	3.53E-03	3.51E-03	3.06E-03	3.28E-03	3.09E-03	3.26E-03
	Switching	2.40E-03	2.23E-03	1.99E-03	2.09E-03	1.91E-03	1.96E-03
	Leakage	2.39E-04	2.39E-04	1.74E-04	2.31E-04	2.31E-04	1.74E-04
	Total	6.17E-03	5.98E-03	5.23E-03	5.61E-03	5.23E-03	5.39E-03
Path Delays (ns)	Min	1.15	1.12	1.09	1.13	1.13	1.11
	Median	1.23	1.19	1.16	1.2	1.19	1.17
	Max	1.69	1.56	1.48	1.55	1.53	1.47
Total wire length (μm)		5.00E+05	4.89E+05	2.18E+05	2.16E+05	2.41E+05	2.14E+05
Area (μm^2)		58680.1			58602.7		
#Std-cell		30			30		
#Cell		4662		2636	4760		2636
#OBISA cell		2020		0	2124		0
Security Level k		1	30	1	1	35	1

Module		aes_256_hier1		
Layout		OBISA	Ctrl	
			Only	Lifted
Power (W)	Internal	3.49E-02	3.27E-02	2.71E-02
	Switching	1.11E-01	7.89E-02	8.78E-02
	Switching	1.18E-02	1.18E-02	1.92E-04
	Total	1.58E-01	1.23E-01	1.15E-01
Path Delays (ns)	Min	0.33	0.33	0.35
	Median	0.49	0.49	0.44
	Max	1.63	1.14	1.27
Total wire length (μm)		1.19E+07	1.10E+07	1.06E+07
Area (μm^2)		5674310		
#Std-cell		106		
#Cell		108087		2636
#OBISA cell		107036		0
Security Level k		1	208	1

that of functional cells, as well as number of standard cell models instantiated: Results are provided to illustrate impact upon OBISA due to specific functional module design. For example, in *key_sel* module, due to low functional cell count and high security level necessary, fewer standard cell models were used during OBISA insertion. On the other hand, close total wire length results between OBISA-inserted layout with (*Lifted* column) and without (*Only* column) wire lifting help to explain why little power and path delay difference were observed between these two types of layouts.

2. In terms of area, power, and path delays OBISA-reinforced layout that underwent wire lifting (*Lifted* column under *OBISA*) column) is compared against similarly OBISA-reinforced layout without wire lifting (*Only* column under *OBISA* column) as well as layout of same module without any security enhancement (*Ctrl* column). Area result are the same for all three scenarios since the same utilization ratio 0.6 was used for all layouts during their floorplanning stage. There is a slight increase in terms of power and path delays in the *Lifted* column with regard to the *Ctrl* column, but in all implementations quite small, and the worst-case path delay overhead in both cores are 3.64% and 4.08% respectively, while the total power overheads are 12.73% and 6.96%. Based on these results, we are confident to conclude the proposed wire lifting based OBISA technique introduces no significant performance overhead to the original circuitry.

Implementation results shown in Table 5.5 and Table 5.6 points at two improve-

ments of significance that were achieved on top of the performance reported in [76]:

1. a much larger and more standard design (AES) was processed to a much higher level of security; and

2. overheads in area, delay, and power are reduced from tens to hundreds percent to around ten percent in power, less than five percent in delay, and zero percent in area; further, limitation on number of standard cell models was also removed.

The first difference between the AES module and DES module is their difference in size: *one_round* module of AES has more than ten times as many gates as *crp* module of DES, even before we consider additional cells brought about by insertion of OBISA circuitry. All things considered, the OBISA-inserted AES core consisted of $385,001$ gates, more than 25 times as many gates as a DES core without OBISA insertion. Another difference is in the fact that DES core is a very specific design: only its *key_sel* module and its topmodule have flip-flops, both of which are instantiated only once. Therefore, implementation on AES core, whose modules are all clocked, demonstrates the ability to be applied on synchronous design, as we have predicted during our introduction of our pin-based definition of edges in Section 5.2.2. A final observation is that the previously reported security level $k = 64$ [76] can be achieved with $k = 4$ for each *crp* core and lifting all edges in *key_sel* and top module; on the other hand, our proposed BP-based wire lifting approach allowed presented implementation to reach security levels such as $k = 160$ and $k = 208$ with ease. This supported our early observation that satisfying an arbitrarily high security level is not only easy for

the proposed BP-based approach, it often takes it even less time to conclude than lower security levels which may have more viable solution candidates.

Equally significant is the reduction in overheads. As was theorized previously in Section 5.2.3, the huge overhead* in [76] was most likely result of the approach of eliminating layout cues by preventing place and route tool to optimize the design according to its function. Our implementation result corroborated our earlier theory that removal of this restriction would greatly reduce the overhead, and our evaluation in terms of known layout-based split manufacturing security metrics supported our hypothesis that it would not greatly impact security performance. Even if it did, it would be more than sufficiently compensated by greatly improved security level k. Our theory that OBISA insertion would help remove the restriction on number of standard cell models was also supported by our implementation result: only design where any such restriction was felt was *crp* module of only 745 gates, where we achieved $k = 10$, and could have further improved that number had we allowed ourselves overhead in area.

Based on such significant improvements in achievable security level, size of the design, range of applicability, and overhead reduction, we consider the presented OBISA technique the first technique capable of realistic implementation on industrial-sized applications.

*54% to 92% in power, 73% to 114% in delay, 167% to 502% in area were reported in [76].

5.4 Summary

In this chapter, we have presented a novel implementation approach of Obfuscated Built-In Self-Authentication (OBISA) technique that combines hardware Trojan deterrence through Built-In Self-Authentication (BISA) circuit insertion as well as optimized split manufacturing through wire lifting. The resulting technique is shown to be efficient, secure, and introduces very low performance overhead to the functional design that it is fit for industrial level of integration. The presented implementation flow is tailored to work with all mainstream EDA tools. In the future, the proposed flow could be further improved by expanding the presented technique to further reduce overhead and improve solution generation efficiency.

Chapter 6

Conclusion

This dissertation presents investigation of advanced CAD techniques in two main approaches: In *concurrent application of logic and memory tests*, a novel design-for-test architecture was proposed, and a flow presented to automate application of this architecture; In *sensor selection for sensor-based speed binning*, *framework to assess layout for vulnerabilities to microprobing attacks*, and *obfuscated built-in self-authentication*, optimization algorithms such as genetic algorithm and integer programming were used to help find optimal design solutions. Meanwhile, the problems these techniques seek to address are new. In *concurrent application of logic and memory tests* and *sensor selection for sensor-based speed binning*, the problems to be addressed are relatively apparent result of technological advancement; while in *framework to assess layout for vulnerabilities to microprobing attacks*, and *obfuscated built-in self-authentication*, the problems are not entirely apparent from the technological point of view alone. Nevertheless, our application of the proven approaches demonstrated significant improvement in all investigated problems, and shows potential for further improvement on investigated and further problems in the field of IC design.

6.1 Future Work

Improving state-of-art CAD practice to adapt to changing industry demand is a continuous process. As technology nodes and level of integration in SOC ICs advance, so must our proposed solutions to challenges they introduce; and as attacks on hardware security develop, so must countermeasures designed to stop them. In the rest of my doctorate program I would like to further improve the proposed methods in the following aspects:

1. **Sensor Selection for Sensor-based Speed Binning:** Further validation of the proposed approach can be beneficial, both for verification of validity on larger sample sizes, and under samples fabricated under a wider range of technology nodes.

2. **Concurrent Application of Logic and Memory Tests:** The proposed architecture was evaluated using OpenSPARC T1 core, a microprocessor unit design. Its applicability can be further verified using SOC benchmark circuits, and in cooperation with existing techniques designed for other important performance metrics, such as test scheduling and test power reduction.

3. **Framework to Assess Layout for Vulnerabilities to Microprobing Attacks:** Much remains to be done in this field. The proposed assessment framework and algorithm demonstrates possibility of a more customized active shield design than simply occupying an entire routing layer to cover the whole layout. Indeed,

with the help of the proposed algorithm it is possible to construct an active shield on the same layer as functional routing. Such a design will be advantageous to current entire layer design, not only because of reduced cost in entire routing layers, but also in increasing the amount of reverse engineering necessary for the attack through redundant coverage of security critical wires, thereby further improving security against microprobing attacks. Further, microprobing attack is still very under-studied. Existing research exist in documenting attack methods, investigating circuit diagnostic tools that may serve as technological enablers, shield designs and non-shield protections, but few seek to put all related work into the same big picture. The proposed approach made a first step in taking the first three categories in consideration, and certainly could benefit in including non-shield protection designs such as t-private circuit and related attacks such as side-channel attacks.

4. **Obfuscated Built-In Self-Authentication:** OBISA combines hardware Trojan prevention and wire-lifting techniques to address the complete plethora of threats posed by untrusted foundries. Since we have shown that by taking advantage of BISA insertions the obfuscation effect of wire-lifting technique can also be improved, it hints at possibility of further enhancing obfuscation effect using BISA insertion by combining it with other obfuscation techniques. Moreover, current wire-lifting technique, although secure, is still computationally prohibitive. Future works should also consider the possibility of making it less so, so that it can

see a real chance at industrial scale implementation.

6.2 IMPACT OF THE DISSERTATION WORK

I was invited to present our work on concurrent application of logic and memory tests at 2015 IEEE International Test Conference (ITC). Our paper on proposed framework to assess circuit layout for vulnerabilities to microprobing attack [118] was awarded the best paper award at 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST).

Bibliography

[1] "International technology roadmap for semiconductors, 2013 edition, interconnect," 2013. [Online]. Available: http://www.itrs2.net/2013-itrs.html

[2] D. Belete, A. Razdan, W. Schwarz, R. Raina, C. Hawkins, and J. Morehead, "Use of dft techniques in speed grading a 1 ghz+ microprocessor," in *Test Conference, 2002. Proceedings. International*, 2002, pp. 1111–1119.

[3] B. Cory, R. Kapur, and B. Underwood, "Speed binning with path delay test in 150-nm technology," *Design Test of Computers, IEEE*, vol. 20, no. 5, pp. 41 – 45, sept.-oct. 2003.

[4] D. Wang and W. Mcnall, "A statistical model based asic skew selection method," in *Microelectronics and Electron Devices, 2004 IEEE Workshop on*, 2004, pp. 64 – 66.

[5] X. Wang, M. Tehranipoor, and R. Datta, "Path-ro: A novel on-chip critical path delay measurement under process variations," in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, nov. 2008, pp. 640 –646.

[6] S. Pei, Z. Li, H. Li, X. Li, and S. Wei, "A unified architecture for speed-binning and circuit failure prediction and detection," in *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, vol. 2, may 2012, pp. 418 –421.

[7] E. J. Jang, A. Gattiker, S. Nassif, and J. Abraham, "An oscillation-based test structure for timing information extraction," in *VLSI Test Symposium (VTS), 2012 IEEE 30th*, april 2012, pp. 74 –79.

[8] H. Onodera and H. Terada, "Characterization of wid delay variability using ro-array test structures," in *ASIC, 2009. ASICON '09. IEEE 8th International Conference on*, oct. 2009, pp. 658 –661.

[9] A. Ghosh, R. Rao, C.-T. Chuang, and R. Brown, "On-chip process variation detection and compensation using delay and slew-rate monitoring circuits," in

Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on, march 2008, pp. 815 –820.

[10] S. Wang, J. Chen, and M. Tehranipoor, "Representative critical reliability paths for low-cost and accurate on-chip aging evaluation," in *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*, nov. 2012, pp. 736 – 741.

[11] S. Paul, S. Krishnamurthy, H. Mahmoodi, and S. Bhunia, "Low-overhead design technique for calibration of maximum frequency at multiple operating points," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, nov. 2007, pp. 401 –404.

[12] "International technology roadmap for semiconductors, 2013 edition," 2013. [Online]. Available: http://www.itrs2.net/2013-itrs.html

[13] "International technology roadmap for semiconductors, 2013 edition, test," 2013. [Online]. Available: http://www.itrs2.net/2013-itrs.html

[14] S. Skorobogatov, "Physical attacks on tamper resistance: progress and lessons," in *Proc. of 2nd ARO Special Workshop on Hardware Assurance, Washington, DC*, 2011.

[15] R. Anderson, *Security engineering: A guide to building dependable distributed systems*. Wiley, 2001.

[16] C. Tarnovsky, "Tarnovsky deconstruct processor," youtube. [Online]. Available: https://www.youtube.com/watch?v=w7PT0nrK2BE

[17] "IARPA Trusted Integrated Circuits (TIC) program announcement," http://www.fbo.gov.

[18] R. Jarvis and M. McIntyre, "Split manufacturing method for advanced semiconductor circuits," May 27 2004, uS Patent App. 10/305,670. [Online]. Available: http://www.google.com/patents/US20040102019

[19] Q. Liu and S. Sapatnekar, "Confidence scalable post-silicon statistical delay prediction under process variations," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, June 2007, pp. 497–502.

[20] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "Varius: A model of process variation and resulting timing errors for microarchitects," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 21, no. 1, pp. 3–13, Feb 2008.

[21] T. Karnik, S. Borkar, and V. De, "Probabilistic and variation-tolerant design: Key to continued moore's law."

[22] J. Chen and M. Tehranipoor, "Critical paths selection and test cost reduction considering process variations," in *Proceedings of the 2013 22Nd Asian Test Symposium*, ser. ATS '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 259–264. [Online]. Available: http://dx.doi.org/10.1109/ATS.2013.55

[23] W. Corbin, B. Kessler, E. Nelson, T. Obremski, and D. Wheater, "Testing logic and embedded memory in parallel," Sep. 5 2006, uS Patent 7,103,814. [Online]. Available: https://www.google.com/patents/US7103814

[24] K. Anzou and C. Tokunaga, "Semiconductor integrated circuit," Aug. 7 2007, uS Patent 7,254,762. [Online]. Available: http://www.google.com/patents/US7254762

[25] M. Seuring, "Combining scan test and built-in self test," *Journal of Electronic Testing*, vol. 22, no. 3, pp. 297–299, 2006.

[26] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core based system chips," in *Test Conference, 1998. Proceedings., International*. IEEE, 1998, pp. 130–143.

[27] Y. Zorian, "Embedded memory test and repair: infrastructure ip for soc yield," in *Test Conference, 2002. Proceedings. International*. IEEE, 2002, pp. 340–349.

[28] F. DaSilva, Y. Zorian, L. Whetsel, K. Arabi, and R. Kapur, "Overview of the ieee p1500 standard," in *null*. IEEE, 2003, p. 988.

[29] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, S. M. Reddy, and K. Chakrabarty, "On concurrent test of core-based soc design," in *SOC (System-on-a-Chip) Testing for Plug and Play Test Automation*. Springer, 2002, pp. 37–50.

[30] Q. Xu and N. Nicolici, "On concurrent test of wrapped cores and unwrapped logic blocks in socs," in *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*. IEEE, 2005, pp. 10–pp.

[31] A. Sehgal, J. Fitzgerald, and J. Rearick, "Test cost reduction for the amdtm athlon processor using test partitioning1," 2007.

[32] R. Tekumalla, P. Chaudhuri, P. Kumar, and K. Shah, "Efficient wrapper cell design for scan testing of integrated," May 27 2014, uS Patent 8,738,978. [Online]. Available: https://www.google.com/patents/US8738978

[33] S. K. Goel, E. J. Marinissen, A. Sehgal, and K. Chakrabarty, "Testing of socs with hierarchical cores: common fallacies, test access optimization, and test scheduling," *Computers, IEEE Transactions on*, vol. 58, no. 3, pp. 409–423, 2009.

[34] P.-L. Yang, C.-C. Lin, M.-Z. Kuo, S.-H. Dhong, C.-M. Lin, K. Huang, C.-N. Peng, and M.-J. Wang, "A 4-ghz universal high-frequency on-chip testing platform for ip validation," in *VLSI Test Symposium (VTS), 2014 IEEE 32nd*. IEEE, 2014, pp. 1–6.

[35] K.-J. Lee, T.-Y. Hsieh, C.-Y. Cha, Y.-T. Hong, and W.-C. Huang, "On-chip soc test platform design based on ieee 1500 standard," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 7, pp. 1134–1139, 2010.

[36] L.-C. Chen, P. Dickinson, P. Mantri, M. Gala, P. Dahlgren, S. Bhattacharya, O. Caty, K. Woodling, T. Ziaja, D. Curwen *et al.*, "Transition test on ultrasparc-t2 microprocessor," in *2008 IEEE International Test Conference*. IEEE, 2008, pp. 1–10.

[37] S. Hamdioui and Z. Al-Ars, "Scan more with memory scan test," in *Design & Technology of Integrated Systems in Nanoscal Era, 2009. DTIS'09. 4th International Conference on*. IEEE, 2009, pp. 204–209.

[38] V. Devanathan, A. Hales, S. Kale, and D. Sonkar, "Towards effective and compression-friendly test of memory interface logic," in *2010 IEEE International Test Conference*. IEEE, 2010, pp. 1–10.

[39] R. Tekumalla and P. Krishnamoorthy, "At-speed scan test of memory interface," in *North Atlantic Test Workshop, Proc. of*. IEEE, 2013.

[40] V. Ray, "Freud applications of fib: Invasive fib attacks and countermeasures in hardware security devices," in *East-Coast Focused Ion Beam User Group Meeting*. Feburuary, 2009.

[41] C. Tarnovsky, "Security failures in secure devices," in *Black Hat Briefings*. Feburuary, 2008.

[42] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert, "Breaking and entering through the silicon," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 733–744.

[43] C. Boit, C. Helfmeier, and U. Kerst, "Security risks posed by modern ic debug and diagnosis tools," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*. IEEE, 2013, pp. 3–11.

[44] J.-M. Cioranesco, J.-L. Danger, T. Graba, S. Guilley, Y. Mathieu, D. Naccache, and X. T. Ngo, "Cryptographically secure shields," in *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 25–31.

[45] P. Laackmann and H. Taddiken, "Apparatus for protecting an integrated circuit formed in a substrate and method for protecting the circuit against reverse engineering," Sep. 28 2004, uS Patent 6,798,234.

[46] M. Ling, L. Wu, X. Li, X. Zhang, J. Hou, and Y. Wang, "Design of monitor and protect circuits against fib attack on chip security," in *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on*. IEEE, 2012, pp. 530–533.

[47] A. Beit-Grogger and J. Riegebauer, "Integrated circuit having an active shield," Nov. 8 2005, uS Patent 6,962,294. [Online]. Available: https://www.google.com/patents/US6962294

[48] S. Briais, J.-M. Cioranesco, J.-L. Danger, S. Guilley, D. Naccache, and T. Porteboeuf, "Random active shield," in *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on*. IEEE, 2012, pp. 103–113.

[49] U. Guin, D. Forte, and M. Tehranipoor, "Anti-counterfeit techniques: from design to resign," in *2013 14th International Workshop on Microprocessor Test and Verification*. IEEE, 2013, pp. 89–94.

[50] M. M. Tehranipoor, U. Guin, and D. Forte, "Counterfeit integrated circuits," in *Counterfeit Integrated Circuits*. Springer, 2015, pp. 15–36.

[51] U. Guin, Q. Shi, D. Forte, and M. M. Tehranipoor, "Fortis: A comprehensive solution for establishing forward trust for protecting ips and ics," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 4, p. 63, 2016.

[52] U. Guin, "Establishment of trust and integrity in modern supply chain from design to resign," 2016.

[53] K. Xiao, "Techniques for improving security and trustworthiness of integrated circuits," 2015.

[54] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 112–125, 2012.

[55] J. Li and J. Lach, "At-speed delay characterization for ic authentication and trojan horse detection," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 8–14.

[56] Y. Jin, N. Kupp, and Y. Makris, "Dftt: Design for trojan test," in *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 1168–1171.

[57] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri, "Design and analysis of ring oscillator based design-for-trust technique," in *29th VLSI Test Symposium*. IEEE, 2011, pp. 105–110.

[58] H. Salmani and M. Tehranipoor, "Layout-aware switching activity localization to enhance hardware trojan detection," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 76–87, 2012.

[59] R. S. Chakraborty and S. Bhunia, "Security against hardware trojan through a novel application of design obfuscation," in *Proceedings of the 2009 International Conference on Computer-Aided Design*. ACM, 2009, pp. 113–116.

[60] M. Banga and M. S. Hsiao, "Odette: A non-scan design-for-test methodology for trojan detection in ics," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 18–23.

[61] R. S. Chakraborty and S. Bhunia, "Harpoon: an obfuscation-based soc design methodology for hardware protection," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 10, pp. 1493–1502, 2009.

[62] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 83–89.

[63] X. Yang, B.-K. Choi, and M. Sarrafzadeh, "Routability-driven white space allocation for fixed-die standard-cell placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 4, pp. 410–419, Apr 2003.

[64] S. Charlebois, P. Dunn, and G. Rohrbaugh, "Method of optimizing customizable filler cells in an integrated circuit physical design process," Oct. 28 2008, uS Patent 7,444,609. [Online]. Available: https://www.google.com/patents/US7444609

[65] 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*, vol. 33, no. 12, pp. 1778–1791, 2014.

[66] M. Bushnell and V. D. Agrawal, *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*. Springer Science & Business Media, 2000, vol. 17.

[67] S. Jha and S. K. Jha, "Randomization based probabilistic approach to detect trojan circuits," in *High Assurance Systems Engineering Symposium, 2008. HASE 2008. 11th IEEE*. IEEE, 2008, pp. 117–124.

[68] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 15–19.

[69] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using ic fingerprinting," in *2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 2007, pp. 296–310.

[70] S. Narasimhan, X. Wang, D. Du, R. S. Chakraborty, and S. Bhunia, "Tesr: A robust temporal self-referencing approach for hardware trojan detection," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 71–74.

[71] J. Aarestad, D. Acharyya, R. Rad, and J. Plusquellic, "Detecting trojans through leakage current analysis using multiple supply pad s," *IEEE Transactions on information forensics and security*, vol. 5, no. 4, pp. 893–904, 2010.

[72] Y. Alkabani and F. Koushanfar, "Consistency-based characterization for ic trojan detection," in *Proceedings of the 2009 International Conference on Computer-Aided Design*. ACM, 2009, pp. 123–127.

[73] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 51–57.

[74] K. Xiao, X. Zhang, and M. Tehranipoor, "A clock sweeping technique for detecting hardware trojans impacting circuits delay," *IEEE Design & Test*, vol. 30, no. 2, pp. 26–34, 2013.

[75] J. J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1259–1264.

[76] F. Imeson, A. Emtenan, S. Garg, and M. Tripunitara, "Securing computer hardware using 3d integrated circuit (ic) technology and split manufacturing for obfuscation," in *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, 2013, pp. 495–510.

[77] M. Jagasivamani, P. Gadfort, M. Sika, M. Bajura, and M. Fritze, "Split-fabrication obfuscation: Metrics and techniques," in *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 7–12.

[78] Y. Wang, P. Chen, J. Hu, and J. J. Rajendran, "The cat and mouse in split manufacturing," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 165.

[79] J. Magaña, D. Shi, and A. Davoodi, "Are proximity attacks a threat to the security of split manufacturing of integrated circuits?" in *Proceedings of the 35th International Conference on Computer-Aided Design*, ser. ICCAD '16. New York, NY, USA: ACM, 2016, pp. 90:1–90:7. [Online]. Available: http://doi.acm.org/10.1145/2966986.2967006

[80] C. T. O. Otero, J. Tse, R. Karmazin, B. Hill, and R. Manohar, "Automatic obfuscated cell layout for trusted split-foundry design," in *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 56–61.

[81] Y. Xie, C. Bao, and A. Srivastava, "Security-aware design flow for 2.5d ic technology," in *Proceedings of the 5th International Workshop on Trustworthy Embedded Devices*, ser. TrustED '15. New York, NY, USA: ACM, 2015, pp. 31–38. [Online]. Available: http://doi.acm.org/10.1145/2808414.2808420

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

[83] Q. Shi, K. Xiao, D. Forte, and M. M. Tehranipoor, "Obfuscated built-in self-authentication," in *Hardware Protection through Obfuscation*. Springer International Publishing, 2017, ch. 11, pp. 263–289.

[84] R. J. Turk *et al.*, *Cyber incidents involving control systems*. Idaho National Engineering and Environmental Laboratory, 2005.

[85] M. Mitchell, *An introduction to genetic algorithms*. Cambridge, Mass: MIT Press, 1996.

[86] X. Wang, M. Tehranipoor, S. George, D. Tran, and L. Winemberg, "Design and analysis of a delay sensor applicable to process/environmental variations and aging measurements," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 8, pp. 1405–1418, Aug 2012.

[87] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: http://dx.doi.org/10.1007/BF00994018

[88] X. Liang, R. Canal, G.-Y. Wei, and D. Brooks, "Process variation tolerant 3t1d-based cache architectures," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 40. Washington, DC, USA: IEEE Computer Society, 2007, pp. 15–26. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2007.33

[89] K. Numpacharoen and A. Atsawarungruangkit, "Generating correlation matrices based on the boundaries of their coefficients." [Online]. Available: http://ssrn.com/abstract=2127689

[90] The MathWorks, Inc., "MATLAB®release 2012b, and statistics and machine learning toolbox™ are copyrights of MathWorks, Inc." Natick, Massachusetts, United States.

[91] Synopsys, Inc., "Synopsys® is the copyright of Synopsys, Inc." Mountain View, California, United States.

[92] ——, "HSPICE® is the copyright of Synopsys, Inc." Mountain View, California, United States.

[93] *HSPICE User Guide: Basic Simulation and Analysis*, K2015.06 ed., Synopsys, Inc.

[94] Oracle, Inc., "OpenSPARC™ is the copyright of Oracle, Inc." Santa Clara, California, United States.

[95] "OpenSPARC T1 Micro Architecture Specification," http://www.oracle.com/technetwork/systems/opensparc/t1-01-opensparct1-micro-arch-1538959.html.

[96] "Synopsys 32/28nm Generic Library for Teaching IC Design," https://www.synopsys.com/COMMUNITY/UNIVERSITYPROGRAM/Pages/32-28nm-generic-library.aspx.

[97] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 23, no. 5, pp. 776–792, 2004.

[98] W.-T. J. Chan, A. B. Kahng, S. Nath, and I. Yamamoto, "The itrs mpu and soc system drivers: Calibration and implications for design-based equivalent scaling in the roadmap," in *2014 IEEE 32nd International Conference on Computer Design (ICCD)*. IEEE, 2014, pp. 153–160.

[99] Y. Li, O. Mutlu, D. S. Gardner, and S. Mitra, "Concurrent autonomous self-test for uncore components in system-on-chips," in *2010 28th VLSI Test Symposium (VTS)*. IEEE, 2010, pp. 232–237.

[100] F. DaSilva, "Ieee standard testability method for embedded core-based integrated circuits," *IEEE Std 1500TM-2005*, 2005.

[101] Z. Al-Ars, S. Hamdioui, G. Gaydadjiev, and S. Vassiliadis, "Test set development for cache memory in modern microprocessors," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 16, no. 6, pp. 725–732, 2008.

[102] R. Tekumalla and P. Krishnamoorthy, "At-speed scan test of memory interface," in *IEEE North Atlantic Test Worshop*. IEEE, 2013, pp. 1–10.

[103] Q. Xu and N. Nicolici, "Wrapper design for testing ip cores with multiple clock domains," in *Proceedings of the conference on Design, automation and test in Europe-Volume 1*. IEEE Computer Society, 2004, p. 10416.

[104] A. Chandra and K. Chakrabarty, "Reduction of soc test data volume, scan power and testing time using alternating run-length codes," in *Proceedings of the 39th annual Design Automation Conference*. ACM, 2002, pp. 673–678.

[105] K. Chakrabarty, V. Iyengar, and M. D. Krasniewski, "Test planning for modular testing of hierarchical socs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 3, pp. 435–448, 2005.

[106] E. Larsson, "Efficient test solutions for core-based designs," *Introduction to Advanced System-on-Chip Test Design and Optimization*, pp. 215–251, 2005.

[107] M. A. Breuer and A. D. Friedman, "Diagnosis and reliable design of digital systems," *Digital System Design Series, London: Pitman, 1977*, 1977.

[108] A. J. van de Goor and J. De Neef, "Industrial evaluation of dram tests," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 1999, p. 123.

[109] S. Hamdioui, Z. Al-Ars, and A. J. van de Goor, "Testing static and dynamic faults in random access memories," in *VLSI Test Symposium, 2002.(VTS 2002). Proceedings 20th IEEE*. IEEE, 2002, pp. 395–400.

[110] M. S. Abadir and H. K. Reghbati, "Functional testing of semiconductor random access memories," *ACM Computing Surveys (CSUR)*, vol. 15, no. 3, pp. 175–198, 1983.

[111] S. Hamdioui, A. J. van de Goor, and M. Rodgers, "March ss: A test for all static simple ram faults." in *MTDT*, 2002, pp. 95–100.

[112] S. Manich, M. S. Wamser, and G. Sigl, "Detection of probing attempts in secure ics," in *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 134–139.

[113] "Freepdk45: Metal layers." [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Metal_Layers

[114] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology-CRYPTO 2003*. Springer, 2003, pp. 463–481.

[115] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor, "A survey on chip to system reverse engineering," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 1, p. 6, 2016.

[116] K. Xiao and M. Tehranipoor, "Bisa: Built-in self-authentication for preventing hardware trojan insertion," in *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 45–50.

[117] G. Gamrath, T. Fischer, T. Gally, A. M. Gleixner, G. Hendel, T. Koch, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S. Schenker, R. Schwarz, F. Serrano, Y. Shinano, S. Vigerske, D. Weninger, M. Winkler, J. T. Witt, and J. Witzig, "The scip optimization suite 3.2," ZIB, Takustr.7, 14195 Berlin, Tech. Rep. 15-60, 2016.

[118] Q. Shi, N. Asadizanjani, D. Forte, and M. M. Tehranipoor, "A layout-driven framework to assess vulnerability of ics to microprobing attacks," in *Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on*, May 2016.