REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
14-12-2015	Final Report	15-Sep-2011 - 14-Sep-2014

4. TITLE AND SUBTITLE
Final Report: Design-for-Hardware-Trust Techniques, Detection Strategies and Metrics for Hardware Trojans

5a. CONTRACT NUMBER
W911NF-11-1-0480

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS
Mark Tehranipoor

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
University of Connecticut - Storrs
Sponsored Program Services
438 Whitney Road Ext., Unit 1133
Storrs, CT 06269 -1133

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
57958-CS.10

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
The ARO grant provided support for a full time PhD student in Kan Xiao. Kan is graduating in Dec. 2015 and will join Intel in Jan. 2016. Through the course of this project we developed novel hardware Trojan detection techniques based on clock sweeping. The technique takes advantage of the change in circuit delay because of inserted Trojan. The change is of course minimal, hence we developed an effective classification algorithms to detect minor changes due to Trojan and compared them with those changes made by process variations. This technique was implemented on a large number of Xilinx FPGAs and demonstrated its efficiency. The second major contribution is

15. SUBJECT TERMS
Design for trust, hardware Trojan, obfuscation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Mohammad Tehranipoor
UU	UU	UU	UU		19b. TELEPHONE NUMBER 860-486-3471

Report Title

Final Report: Design-for-Hardware-Trust Techniques, Detection Strategies and Metrics for Hardware Trojans

ABSTRACT

The ARO grant provided support for a full time PhD student in Kan Xiao. Kan is graduating in Dec. 2015 and will join Intel in Jan. 2016. Through the course of this project we developed novel hardware Trojan detection techniques based on clock sweeping. The technique takes advantage of the change in circuit delay because of inserted Trojan. The change is of course minimal, hence we developed an effective classification algorithms to detect minor changes due to Trojan and compared them with those changes made by process variations. This technique was implemented on a large number of Xilinx FPGAs and demonstrated its efficiency. The second major contribution is development of a novel concept called built-in self-authentication (BISA). BISA fills in the circuit layouts unused spaces with BISA cells which are functional cells. Using a novel algorithm, the BISA cells are added such that the added structure can be easily tested in less than a micro-second. Any changes to the original circuit and/or BISA circuitries can be easily detected. In the 3rd year of the project, we extended this work to combine BISA with split manufacturing. Hence, BISA not only helps with prevention of hardware Trojans in the circuit layout, but also is able to provide an obfuscation capability. The paper was presented in HOST 2014 and received best paper candidate honor.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

The PI has presented the new concepts developed through the course of this project at many universities, government agencies, and industry.

Number of Presentations: 10.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received		Paper
08/06/2013	2.00	A. Ferraiuolo, X. Zhang, and M. Tehranipoor. Experimental Analysis of a Ring Oscillator Network for Hardware Trojan Detection in a 90nm ASIC, International Conference on -Aided Design (ICCAD). 06-NOV-12, . : ,
08/06/2013	3.00	X. Zhang, K. Xiao, and M. Tehranipoor. Path-Delay Fingerprinting for Identification of Recovered ICs, Int. Symposium on Defect and Fault Tolerance in VLSI Systems (DFTS). 04-OCT-12, . : ,
TOTAL:	**2**	

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received		Paper
08/06/2013	6.00	Nicholas Tuzzio, Kan Xiao, Xuehui Zhang, and Mohammad Tehranipoor. A Zero-Overhead IC Identification Technique using Clock Sweeping and Path Delay Analysis, IEEE GLS-VLSI. 04-MAY-12, . : ,
12/14/2015	9.00	K. Xiao, D. Forte, M. Tehranipoor. Efficient and Secure Split Manufacturing via Obfuscated Built-In Self-Authentication, IEEE Hardware Oriented Security and Trust. 04-MAY-15, . : ,
TOTAL:	**2**	

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received		Paper
08/06/2013	5.00	K. Xiao, X. Zhang, and M. Tehranipoor. A Clock Sweeping Technique for Detecting Hardware Trojans Impacting Circuits Delay, IEEE Design and Test of Computers (08 2012)
08/06/2013	4.00	X. Zhang, A. Ferraiuolo, and M. Tehranipoor, 2.. Detection of Hardware Trojans using a Combined Ring Oscillator Network and Off-chip Transient-Power Analysis, ACM Journal on Emerging Technologies in Computing Systems (11 2011)
12/14/2015	8.00	K. Xiao, D. Forte, M. Tehranipoor. A Novel Built-In Self-Authentication Technique to Prevent Inserting Hardware Trojans, ( )
TOTAL:		**3**

Number of Manuscripts:

Books

Received	Book
TOTAL:	

Received	Book Chapter
TOTAL:	

Patents Submitted

Methods and Systems for Preventing Hardware Trojan Insertion, M. Tehranipoor and K. Xiao, US20140283147 A1 (Application)

Patents Awarded

Awards

The following paper was nominated for best paper award at the 2015 HOST Symposium:

K. Xiao, D. Forte, and M. Tehranipoor, "Efficient and Secure Split Manufacturing via Obfuscated Built-In Self-Authentication,"IEEE Hardware-Oriented Security and Trust (HOST), 2015.

Graduate Students

NAME	PERCENT_SUPPORTED	Discipline
Kan Xiao	1.00	
FTE Equivalent:	**1.00**	
Total Number:	**1**	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Mark Tehranipoor	0.10	No
FTE Equivalent:	**0.10**	
Total Number:	**1**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED	Discipline
Shane Kelly	0.10	ECE Department, Undergraduate student
Andrew Ferraiuolo	0.10	ECE Department, Undergraduate student
FTE Equivalent:	**0.20**	
Total Number:	**2**	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period: 2.00

The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:..... 2.00

The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:..... 2.00

Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):..... 2.00

Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:..... 0.00

The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense 0.00

The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields: 0.00

Names of Personnel receiving masters degrees

NAME

Total Number:

Names of personnel receiving PHDs

NAME
Kan Xiao, PhD
Total Number: **1**

Names of other research staff

NAME PERCENT_SUPPORTED

FTE Equivalent:
Total Number:

Sub Contractors (DD882)

Inventions (DD882)

Scientific Progress

In this project we developed innovative concepts that will impact the way integrated circuits are designed and fabricated by untrusted foundries.

The one concept that has received significant amount of attention is "Built-In Self-Authentication (BISA)". We developed a tool from this technique, and have filed a patent. This technology will be part of a newly established startup by Mark Tehranipoor.

A paper based on this technology has been nominated for best paper award. The paper was published in HOST 2015. The winner will be announced in 2016.

A student has been graduate with a PhD degree from the support of this project and will join Intel in Jan 2016.

Technology Transfer

The following technology will be part of a startup company to be established soon by Mark Tehranipoor:

Methods and Systems for Preventing Hardware Trojan Insertion, M. Tehranipoor and K. Xiao, US20140283147 A1 (Application)

Final Project Report

Proposal Number: 57958CS

Mark Tehranipoor

Summary: The ARO grant provided support for a full time PhD student in Kan Xiao. Kan is graduating in Dec. 2015 and will join Intel in Jan. 2016. Through the course of this project we developed novel hardware Trojan detection techniques based on clock sweeping. The technique takes advantage of the change in circuit delay because of inserted Trojan. The change is of course minimal, hence we developed an effective classification algorithms to detect minor changes due to Trojan and compared them with those changes made by process variations. This technique was implemented on a large number of Xilinx FPGAs and demonstrated its efficiency. The second major contribution is development of a novel concept called built-in self-authentication (BISA). BISA fills in the circuit layouts unused spaces with BISA cells which are functional cells. Using a novel algorithm, the BISA cells are added such that the added structure can be easily tested in less than a micro-second. Any changes to the original circuit and/or BISA circuitries can be easily detected. In the 3rd year of the project, we extended this work to combine BISA with split manufacturing. Hence, BISA not only helps with prevention of hardware Trojans in the circuit layout, but also is able to provide an obfuscation capability. The paper was presented in HOST 2014 and received best paper candidate honor.

2011-2015

Contents

1. Introduction

To lower the cost of IC design and fabrication, the supply chain of the semiconductor industry has been distributed around the world. As the complexity of ICs increases, more and more highly specialized companies get involved in the IC fabrication process to enhance efficiency and improve manufacturability, providing attackers with more opportunities to identify (reverse engineering) circuit functionality and make malicious alterations, known as Hardware Trojans. Hardware Trojans have become a major concern to security-critical applications, such as military, transportation and financial systems. Some Trojans can be inserted into a design if any untrusted tools or IP blocks are used. Other Trojans can be implanted by modifying the layout of the design during GDSII development and fabrication. Trojans in ICs may cause malfunctions, lower the reliability of the ICs, leak confidential information to adversaries or even destroy the system under specifically designed conditions [1][2][10]. Detecting these malicious inclusions and alterations is extremely difficult, due to the following characteristics of Trojans: First, Trojans are small compared to the designs they have altered, which attributes of Trojan-inserted ICs are almost the same as those of Trojan-free ICs. Second, Trojans can be kept dormant during most of their operation, and be activated under very specific conditions. Third, a Trojan's behavior is unknown. Thus, it is very challenging to model Trojan's behavior and devise a Trojan detection technique that can target all types of Trojans.

In general, most research in the past several years focused on hardware Trojan detection which can be divided into two categories: Trojan detection approaches and design-for-trust. Trojan detection approaches can be further categorized into full Trojan activation and side-channel analysis [2]. The first approach [3][4][11] tries to activate Trojans by applying test vectors and comparing the responses with the correct results. Intelligent adversaries will ensure Trojans are activated under very rare conditions and can go undetected under structural and functional tests during manufacturing test process. Due to the numerous logical states in a circuit, it is impractical to enumerate all states of a real design. Additionally, instead of changing the functionality of the original circuit, Trojans can be designed to transmit information with an antenna or modify the specification. Full Trojan activation methods may fail to detect these kinds of Trojans. Side-channel signal analysis has been developed to detect hardware Trojans by measuring circuit parameters, such as power (transient [5][12] and leakage current [6]) and delay [7]. A partial activation (i.e., generating signal transitions in a Trojan's component without fully activating the Trojan) of Trojans is still required for the methods based on transient power as a side-channel signal. In order to improve the effectiveness of detection methods, several design-for-trust techniques are proposed to facilitate Trojan detection techniques [7][8] and make Trojan insertion difficult [9]. However, the original design would be altered after applying these design-for-trust techniques. Moreover, as the size of circuit increases, it will have more quiet (low controllability/observability) nets/gates. Because of complexity of processing and large time/area overhead, such techniques are still difficult to apply to a large design that contains millions of gates.

Additionally, some techniques have been developed recently to obfuscate original circuit design and try to hinder reverse engineering by untrusted foundries. Reverse engineering is another major security threat, because reverse engineering could be used to clone, pirate, and counterfeit a design, or to develop new attacks, including hardware Trojans. Logic obfuscation attempts to hide the functionality and the implementation of a design by insertion of built-in locking mechanisms into the original design. The locking circuits become transparent and the right function appears only when a valid key is applied [31]. The increased complexity of

identifying the genuine functionality without knowing the right input vectors is able to dwarf the ability of inserting a targeted Trojan by attackers. Camouflaging is a layout-level obfuscation technique to create indistinguishable layouts for different gates by adding dummy contacts and faking connections between the layers within a camouflaged logic gate [32]. However, these techniques require additional circuit or design efforts in the design phase (RTL or gate-level). The potential area and performance overheads are the chief concerns to IC designers. Thus, the existing techniques are still questionable to apply to a large design that contains millions of gates. In addition, these techniques need to insert additional gates (logic obfuscation) or modify the original standard cells (camouflaging), which could degrade the chip performance significantly and affect their acceptability in high-end circuits. Therefore, a low-cost and effective technique is necessary today to prevent reverse engineering by untrusted foundry.

In order to overcome the drawbacks described above, we proposed one effective hardware Trojan detection method, one hardware Trojan prevention method, and one design obfuscation method with negligible overhead in this report.

The rest of this report is organized as follows.

1. **Section 2** will present our new method of detecting hardware Trojans impacting circuits delay using a clock sweeping technique.
2. **Section 3** will first propose a novel technique to prevent hardware Trojan insertion and algorithms for placement and routing of added cells are presented in this report.
3. **Section 4** will present a new technique to prevent reverse engineering of the circuit functionality and further prevent hardware Trojan insertion with split manufacturing process.
4. **Section 5** will conclude the report and provide a brief future research direction.

2. A Clock Sweeping Technique for Detecting Hardware Trojans Impacting Circuits Delay

Side-channel signal analysis has been developed to detect hardware Trojans by measuring circuit parameters, such as power (transient and leakage current) and delay. The authors in [13] [7] measure path delays to detect changes caused by Trojans. Although effective, only critical paths in [13] are measured, limiting detections of Trojans inserted on non-critical paths. The authors in [7] use one additional shadow register and one comparator to measure each path delay in the circuit.

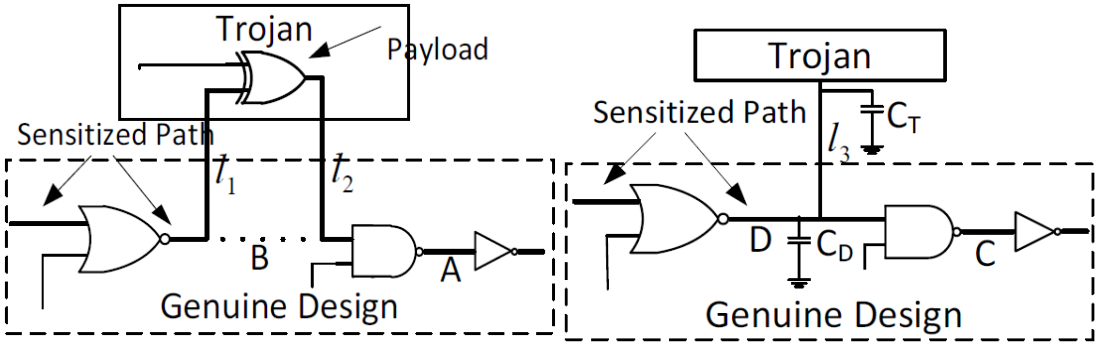
Compared to full Trojan activation and other side-channel signal techniques, a delay-based technique has a unique benefit because it does not need to activate the Trojan either partially or fully. Moreover, each path delay is relatively independent, so it is less affected by other paths of the chip, and a Trojan can potentially contribute more to a path delay change than total circuit power. Existing delay-based Trojan detection methods face the following challenges: (1) to ensure the maximum detection coverage of Trojans that can potentially be placed and distributed on various paths besides critical paths, and (2) the measurement of paths delay at a low cost. Since there is a huge number of paths in a design, any additional hardware for path delay measurement will increase the area and silicon cost significantly. Taking into account these issues, in this work, we propose a “clock sweeping" technique to obtain path delay information without any additional hardware. Transition and path delay fault (TDF and PDF) patterns are used to obtain high coverage on the nodes of critical and non-critical paths. Once the data has been collected by clock sweeping, we generate a series of delay signatures for ICs, and then analyze

whether ICs contain Trojans. Since transitions are easier generated on short paths as demonstrated in [7], the Trojans on the short paths can be detected more efficiently by power-based Trojan detection techniques [5] [6]. This is evident by the fact that nodes on short paths have generally higher controllability and observability [7].

2.1 Background

2.1.1 Trojan Impact on Path Delay

We can expect that intelligent adversaries will try to maintain the original design layout and insert Trojans into unused spaces of the layout to keep the Trojan hidden. In order to simplify this problem, we consider the nodes (outputs of gates) in the genuine IC instead of the paths for the analysis in this section. These nodes might be affected by either a trigger or a payload from a Trojan [13]. Thus, we consider three types of Trojans depending on how they are activated and their action to the functional circuit: Trojans with only payloads (TP), Trojans with only triggers (TT), and Trojans with triggers and payloads (TTP) [13].



(a) (b)

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

(c)

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

(d)

Figure 1: (a) An example of TP, (b) an example of TT, (c) a path's delay without and with TP with short and long l_1 and l_2, and (d) a path delay without and with TT at four different locations.

For any Trojan trying to change the function of design, a payload gate has to be inserted at a node. An example of a TP is shown in Figure 1(a). The sensitized path in the genuine design (the bold line) passes through node *B*. The additional delay consists of the propagation delay of the payload and the delay from the two wires' capacitances (l_1 and l_2). For any internally activated Trojan, the triggered parts will introduce additional interconnections which will cause unavoidable increased capacitance on the node. Besides TP and TT, TTP would have the cumulative effect of TP and TT.

In order to show the effect of TP and TT on paths delay, we performed simulations in 90 nm technology. We inserted two payload gates (minimum-sized NAND) at two positions. One is physically very close to the node (short l_1 and l_2 as in Figure 1(a)) and the other is remote from the node (longer l_1 and l_2). In Figure 1(c), delay of path going through node *B* is measured, and

the results show that a TP has increased the path delay significantly, more so for Trojans with long interconnections.

Next, we place a Trojan gate (minimum-sized NAND) at four different locations, with one input connecting to the node *D* on the sensitized path. The first location is very close to node *D* (l_3 is short as in Figure 1(b)), with locations 2, 3, and 4 being successively further away from node *D*. The delay of sensitized path is measured with and without Trojans for the different locations. This data is shown in Figure 1(d).

The extra delay caused by TT is shown in the third row of Figure 1(d). Although the increased delay is still relatively small at the location 1 (shown as Loc. 1 in Figure 1(d)), the TT effects at locations 2 through 4 are comparable to the effects of the payload shown in Figure 1(c). Thus, as long as the standard cells in standard design style ASICs are well planned and tightly packed, the TT effect could be very obvious. Additionally, if a Trojan is activated and its payload becomes nontransparent, the required transition cannot be generated at payload gate. This faulty function indicates Trojan's existence and makes it easier to detect. *Trojans without triggers and payloads would not be detected using this technique since they most likely use an antenna to receive triggers or leak information.* They can be more effectively detected by power-based Trojan detection approaches [5] [6] because they tend to use more gates and consume more power than TP, TT or TTP.

2.1.2 Clock Sweeping

When using the TDF or PDF test vectors for Trojan detection, only Trojans that increase a path's delay by more than its available slack can be detected. This is unlikely to happen since adversary will design Trojans hard to be detected by these patterns by avoiding critical paths and ensuring that the delay induced by Trojan is smaller than slack. We develop a clock sweeping technique to target shorter paths affected by Trojans without any design or silicon overhead. Clock sweeping involves applying a pattern at different clock frequencies, from a lower speed to higher speeds, which is a common practice in industry used for speed binning of parts. Some paths sensitized by the pattern which are longer than the current clock period start to fail when the clock speed increases. The obtained start-to-fail clock frequency can indicate the delays of the paths sensitized by the patterns.

For example, assume that the six paths in Figure 2(a) can be sensitized by test patterns. The clock period is swept from f_0 to f_5 and the sweep step size is Δt, as shown in Figure 2(b). As an example, path B-D is able to propagate correct values at frequency f_0 to f_3 (pass), and will produce wrong logic values at frequency f_4 (fail). Thus, its start-to-fail clock frequency is f_4, which denotes the length of path B-D is between the frequency f_3 and f_4. When the clock is swept from low frequencies to high frequencies, paths will fail sequentially, with longer paths failing before shorter paths.

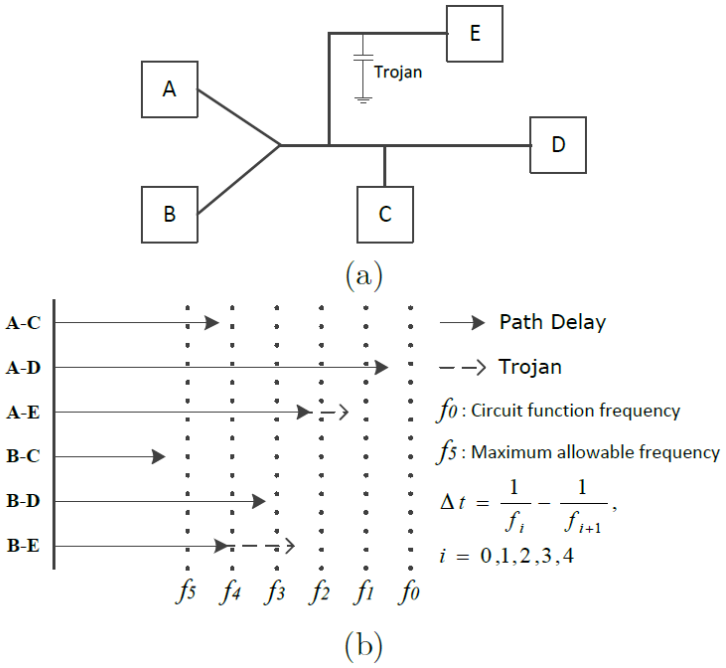


Figure 2: (a) An example circuit and (b) Clock sweeping.

Suppose that a Trojan load is added to a path as shown in Figure 2(a); the additional capacitive load will result in a small extra delay on paths A-E and B-E, which may push the arrow to the right and even fail path A-E at f_2 and path B-E at f_3. In this case, the change of start-to-fail frequency could be detected by clock sweeping. Finally, the maximum frequency applied to the circuit depends on the design characteristics, path-delay distribution in the design, and the on-chip or off-chip frequency generator's limit. Note that in today's designs, most of the paths are long or critical; this is due to the aggressive pipelining to increase circuit performance [14].

2.2 Trojan Detection Methodology

2.2.1 Signature Generation Procedure

Both the nodes on short and long paths have been taken into considerations in our proposed procedure which is shown in Figure 3. The following steps are performed to generate signatures for all ICs.

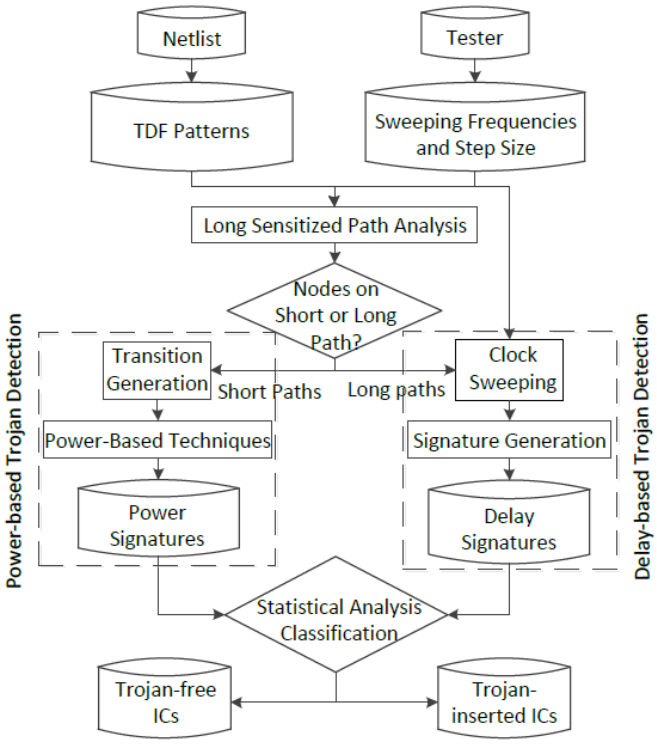


Figure 3: Our proposed signature generation procedure using clock sweeping.

Test Pattern Generation: Similar to the TDF model, here we assume that TP or TT affects a node in the circuit, making the corresponding gate slow to fall or slow to rise. Thus, the TDF patterns are used in our procedure. One advantage of the TDF model is that the number of faults is linear relative to the number of nodes; thus, we can use a portion of all paths to sensitize all nodes in the circuit, and the cost of measuring paths delay could be decreased significantly. Another advantage of using the TDF pattern set is that it is widely used in industry; the ATPG processes are mature and many proposed techniques can be used to increase its fault coverage.

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

Nodes Selection: Once the clock sweeping frequencies are determined, the sensitized path delays larger than the maximum frequency are considered as “long paths". Otherwise, they are called “short paths". For example, in Figure 2(b), B-C is a short path and others are long paths; f_5 is the maximum application frequency. The delay of all the long paths can be obtained from their start-to-fail frequencies. Since the nodes on long paths can be authenticated in clock sweeping, all the patterns sensitizing these long paths will be kept. Patterns only sensitizing short paths are still useful in generating transitions for power-based Trojan detection [5] [6]. This step divides the nodes in a circuit into two groups- *nodes are authenticated either by the clock sweeping or by power-based detection techniques*. Moreover, removing patterns that

sensitize only short paths could save a significant amount of test time during clock sweeping. In this work, we focus on the right branch of the procedure as shown in Figure 3.

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

Signature Generation: By analyzing the pass and fail values, we find the start-to-fail frequency at each flip-flop for each pattern. As an example, Figure 4(a) presents the start-to-fail frequency for each pattern/flip-flop combination in ICs. Long-path patterns are able to sensitize different kinds of paths at different flip-flops: long, short or no path at all. For these flip-flops at which short or no path are sensitized, they always capture ``passing" value during clock sweeping. These invalid pattern/flip-flop combinations need to be discarded from our signature. For example, assuming the pattern 2/flip-flop 2 (P2/FF2) is an invalid combination, the column P2/FF2 has been removed from Figure 4(a). Thus, the final signature length will be shorter, as is shown in Figure 4(a). Each row of the table is a signature for an IC and each column is an element of the signature. The multidimensional signature will be processed by multidimensional scaling described in Section III.D.

	P1/FF1	P1/FF2	…	P2/FF1	P2/FF3	…	PP_1/FFm
IC 1	f_6	f_{14}	…	f_3	f_{20}	…	f_{10}
IC 2	f_7	f_{12}	…	f_4	f_{21}	…	f_9
IC 3	f_5	f_{11}	…	f_3	f_{19}	…	f_{10}
…	…	…	…	…	…	…	…
IC n	f_8	f_{14}	…	f_4	f_{23}	…	f_{12}

(a)

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

(b)

Stage	1	2	3	4	5	6	7
Path1	0.5	0.5	0.13	0.00059	0.00040	0.00023	
Path2	0.5	0.29	0.28	0.063	0.012	0.0024	0.00046
Path3	0.5	0.25	0.11	0.015	0.000296	0.000199	0.000115

(c)

Figure 4: (a) Pattern/flip-flop (P*i*/FF*j*) combinations with start-to-fail frequencies, (b) node coverage on different clock frequencies, (c) transition probabilities on three quiet paths.

2.2.2 Node Coverage Analysis

The objective of our technique is to recognize load capacitance induced by Trojans and capture its impact on a path. The coverage analysis in our technique can be divided into two parts: nodes on long paths and nodes on short paths. Clock sweeping can guarantee that all sensitized long paths will fail at a particular clock frequency. Hence, the node coverage on long paths is dependent on the TDF coverage. The TDF is widely used in VLSI testing, so there have been many approaches proposed to improve the coverage of the TDF in recent years. All these techniques can be applied in our procedure to achieve maximum coverage. In the meantime, a Trojan's load capacitance can slow down both rising and falling transitions. For Trojan detection, one fault, slow-to-rise or slow-to-fall, is sufficient to indicate the existence of Trojans on that node instead of two faults as in the TDF model. Therefore, node coverage will be the ratio of all

detected nodes by either slow-to-rise or slow-to-fall using TDF long-path patterns to the total number of nodes in the circuits.

Figure 4(b) shows the estimated node coverages at different clock frequencies in benchmark s38417 for reference. ATPG patterns which are able to reach 99.4% TDF coverage are used to sensitize all testable paths. The clock period in Figure 4(b) is given in the form of the percentage of the longest critical path (CP) in the circuit.

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

In general, the quietest nodes which have very few transitions are usually on the long paths in a circuit [8]. In order to verify this claim, we apply random patterns to primary inputs and scan chain, and then calculate transition probability for each gate. The gates with low transition probabilities are selected to be analyzed. We choose three quiet paths and their transition probabilities at different stages are shown in the Figure 4(c). As the number of stages of the path increases (1 through 7), the transition probability goes down.

Trojan gates have a higher probability to switch when they connect to nodes on short paths, which means they tend to consume more power [8]. Power-based Trojan detection [5] [6] can effectively detect Trojans on short paths, so adversaries could put Trojans on long paths to hide them. Our clock sweeping technique is able to make up for the deficiency involved with power-based Trojan detection. Increasing the test coverage in both power-based and delay-based approaches can improve the possibility to identify infected chips.

2.2.3 Statistical Data Analysis

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

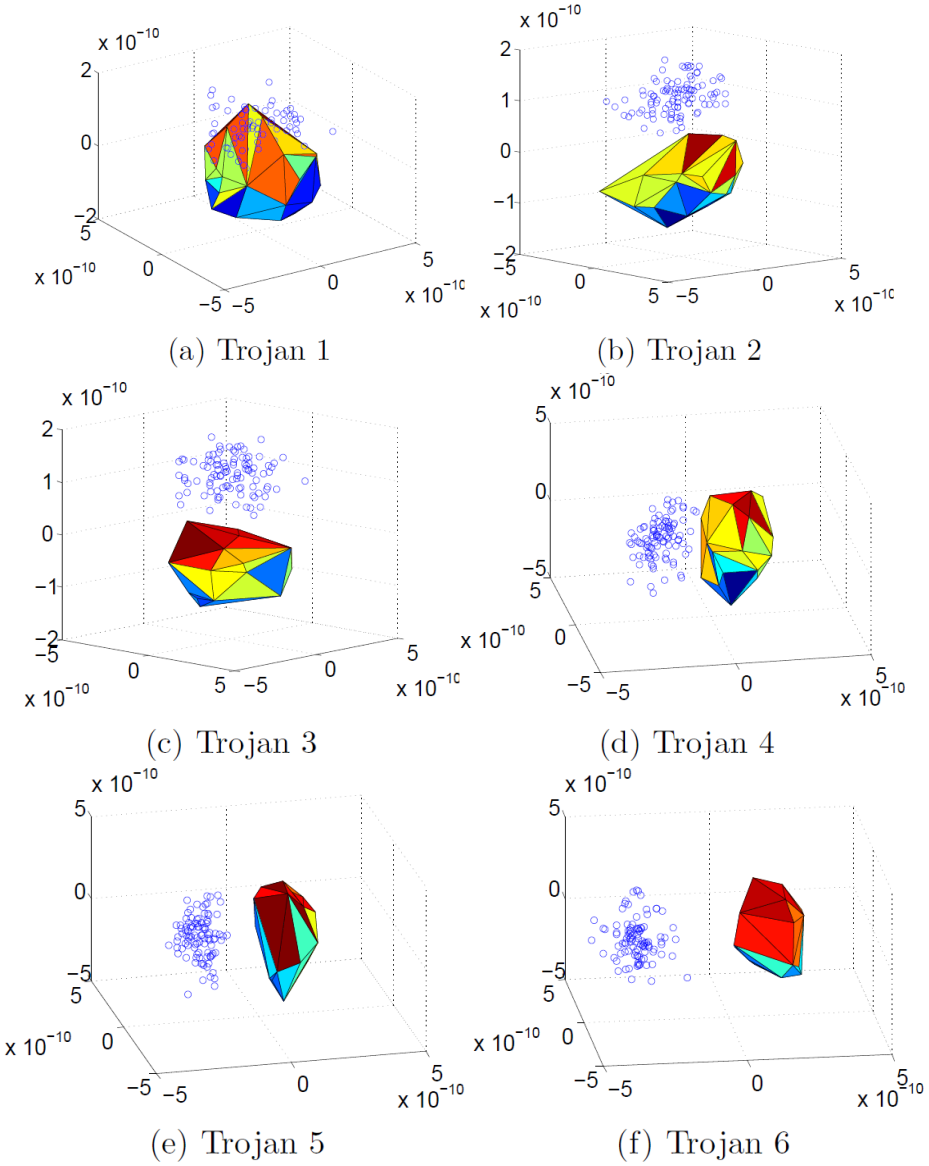
Multidimensional Scaling (MDS) is a method for visualizing dissimilarity in data. The typical goal of MDS is to create a configuration of points in one, two, or three dimensions, whose interpoint distances represent the original dissimilarities. In a similar manner, [13] was first to use PCA/Convex for hardware Trojan detection. The different forms of MDS use different criteria to quantify dissimilarity, such as metric and nonmetric multidimensional scaling [15]. The MDS we used in our method maps the original high dimensional space to a lower dimensional space, at the same time attempts to preserve the pairwise distance and finally isolate the dissimilar chips which may carry Trojans. As described in Section III.A, the signature of a chip is composed of a set of path delays. One element in a signature is considered as a dimension. Their Euclidean distances in high-dimensional space depend on the effects of both process variations and Trojans. Since outlying points in high dimensions contain Trojans and their corresponding points in low dimension are still outliers, we can use outlier points in low dimensions to predict statistical likelihood of Trojan presence. For our technique, the *x*-dimensional signatures for Trojan-free ICs will be mapped to a 3-dimensional space by MDS, and then a convex hull will be constructed. If the signature of an IC under authentication is located outside of the convex hull, this IC is considered suspicious and may contain Trojans.

2.3 Results and Analysis

2.3.1 Simulation Results

In order to demonstrate the effectiveness of our proposed technique, the simulations were performed on an implementation of the ISCAS`89 benchmark s38417 using a 90 nm technology library. After synthesis, the s38417 benchmark circuit has 1564 flip-flops and 4046 logic gates.

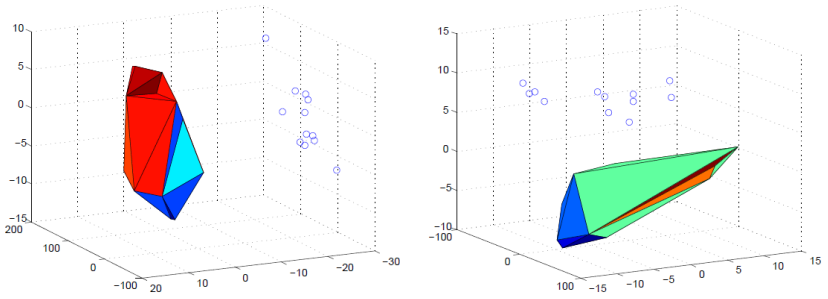
The layout was completed with the Synopsys physical design tool IC Compiler. The Trojan gates were inserted and routed in unused spaces in the layout by using IC Compiler. The impact of process variations on threshold voltage (V_{th}), oxide thickness (T_{ox}) and channel length (L) have been taken into considerations as well. Both the inter-die and intra-die process variations for each of the three previously mentioned parameters are 5% in our simulations. 300 Monte Carlo Simulations, including 200 simulations for Trojan-free chips and 100 for Trojan-inserted chips, were performed at a temperature of 25°C for each Trojan using HSPICE.



(a) Trojan 1 (b) Trojan 2

(c) Trojan 3 (d) Trojan 4

(e) Trojan 5 (f) Trojan 6



(g) Trojan 7 (h) Trojan 8

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

(i) Detection rate at different step sizes.

Figure 5: Outlier analysis using MDS for Trojans 1-6 using simulation ((a)-(f)) and Trojans 7-8 on FPGAs ((g)-(h)), and step sizes analysis (i).

For our delay-based Trojan detection technique, we do not concern ourselves with Trojan's type or how many gates the Trojan has. Instead, we focus on how many triggers and payloads the Trojan will bring to the original design. We have inserted a large number of Trojans in this design among which we selected six Trojans to present results in details. Every Trojan is composed of a few minimum-sized gates, and these gates are placed in the nearest available unused space to keep the wire connections as short as possible. This will make the Trojan harder to detect. These six Trojans were constructed as follows: Trojan 1 has one payload and one trigger (TTP type). Trojan 2 has the same structure as Trojan 1, but is inserted at a different node. Trojan 3 has two payloads with very short connections (TP type). Trojan 4 has four triggers and no payload (TT type). Trojan 5 has three triggers and two payloads (TTP type). Trojan 6 has six triggers and four payloads (TTP type).

The maximum frequency, functional frequency and step size in our simulations are 1.5GHz, 700MHz and 10ps, respectively. By using the methodology described in Figure 3, 300 signatures for 200 Trojan-free ICs and 100 Trojan-inserted ICs are generated. After MDS processing described in Section III.D, the outlying results for Trojans 1 through 6 are shown in Figures 5(a)-(f). The convex hull is formed using signatures of 200 Trojan-free ICs. These points are placed much closer to each other than the rest of hollow dots obtained from Trojan-inserted ICs. According to the distance between outliers and convex hull, two classes of ICs are separated. While Trojan 1's detection rate is 64% (64 out of 100 Trojan-free ICs are detected), from Trojan 2 to Trojan 6, their detection rates are all 100%. The Trojan 1 is the worst case for Trojan with payload and trigger because it has one minimum sized NAND gate which is used as a payload and only one sensitized path passes through this payload. More triggers, payloads and sensitized paths passing through Trojan nodes make detection from Trojan 2 to 6 easier. There might be some limitations associated with the MDS algorithm since it treats the signatures from the ICs as linear. As a part of our future work, we might apply different statistical classification algorithms (such as Diffusion Map and Support Vector Machine) with the aim of further improving the detection rate. However, in our analysis (particularly for Trojans 2 to 6), MDS seems to easily classify all the chips shown in Figures 5(b)-(f). Note that we have randomly inserted Trojans on a very large number of locations in the circuit and was able to observe similar results.

2.3.2 Trojan Size and Location Analysis

Generally, larger Trojans might have more triggers and payloads, so they bring a larger impact to the original circuit. From Trojan 1 to Trojan 6, as the size of Trojans increases, these triggers and payloads affect larger number of nodes in the circuit. In other words, the larger Trojan size means that more paths, sensitized by the TDF patterns, will be impacted by the Trojan. Although Trojans' impacts may be masked by process variations, a larger number of sensitized paths always lead to higher detection possibility. Additionally, larger size Trojan most likely results in longer interconnection for triggers and payloads due to the limited unused space nearby for Trojan insertion. The extra delay induced by the larger Trojan will then increase. This can be seen in Figure 5; as we move from (a) to (f), the points also move away from the convex hull, i.e., the distance between Trojan-inserted ICs and Trojan-free ICs becomes larger. Thus, as the size of the Trojan increases, it becomes easier to separate Trojan-inserted and Trojan-free ICs by using the proposed technique.

In addition to Trojan size, the Trojan's location also has a significant influence on the results. Scan flip-flops can be considered pseudo-primary inputs and outputs. Sensitized paths spread out like a cone from scan flip-flops' outputs and converge at scan flip-flop's inputs. The nodes closer to a scan flip-flop will have more of a chance to influence these sensitized paths. Thus, these Trojans are easier to be detected. In our simulations, although Trojans 1 and 2 have same size, Trojan 2 is closer to scan flip-flops and Trojan 1 is farther away. In Figure 5, hollow dots in (b) are farther from convex hull than that in (a), which means Trojan 2 is easier to be separated than Trojan 1. Thus, we have obtained 100% detection rate for Trojan 2, while Trojan 1 only has a 64% detection rate.

2.3.3 Clock Sweeping Step Size Analysis

For clock sweeping, path delay gained from simulation needs to be translated to their nearest achievable clock frequency according to the clock step size. The range and step size selection are described in Section III.A. From the results shown in Figure 5(i), we can clearly see that the detection rate reduces as the step size increases. The impact of step size becomes less for larger Trojans, because it has more triggers or payloads on sensitized paths and introduces larger extra delay.

2.3.4 FPGA Implementation

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

In this experiment, two types of Trojan with payload are inserted separately in the layout by using Xilinx FPGA Editor. 44 separate FPGA boards were used, with 32 being Trojan-free and 12 being Trojan-inserted. We randomly chose 80 patterns from the 571 TDF patterns for analysis to reduce test and measurement time, so the total number of pattern/flip-flop combinations is 80×145=11600. After removing the invalid pattern/flip-flop combinations which cannot fail any path in the clock sweeping range as described in Section III.A, the remaining number of pattern/flip-flop combinations is 786. These 786 pieces of delay information are used as the signature for each chip. Figure 5(g)-(h) shows the scaled signature by using MDS for Trojan detection. The convex hull is drawn according to the signatures of the 32 Trojan-free FPGAs, and the 12 hollow dots represent the signatures from FPGAs with Trojans. While all hollow dots are

totally separated from the convex hulls, and the detection rate is 100% for both Trojans, we note that the first Trojan is easier to separate as the distance between the hollow dots and convex hull is larger. The reason for this is that the payload gate closer to the scan flip-flops influences 79 sensitized paths, while the other Trojan only affects 9 sensitized paths.

3. Built-In Self-Authentication to Prevent Hardware Trojan Insertion By Untrusted Foundry

In this section, we present a novel Trojan=insertion prevention technique, called built-in self-authentication (BISA), which is able to fill all unused spaces in a circuit layout with functional standard cells instead of non-functional filler cells during layout design. BISA can test functionality of all functional filler cells automatically with low overhead. Therefore, BISA is able to prevent hardware Trojan insertion in limited available spaces. Additionally, BISA's impact on original design is negligible, because there is no connection between BISA circuit and original circuit and they never work simultaneously. In the meantime, the added BISA cells can provide decoupling capacitances to minimize voltage drop when original circuit is working. Moreover, BISA is immune to different types of attacks. Changing or removing any gate in BISA can be detected by BISA itself. Design automation tools have been developed which can apply BISA to designs automatically by designers without any knowledge of Trojans. Finally, BISA eliminates the opportunity for untrusted GDSII developer and foundry to add any malicious circuitry; however, they may still be able to try to carry their intention by removing circuit gates and add their own cells.

3.1 BISA Structure and Insertion Flow

Placement tools are typically conservative, aiming to limit density and spread cells evenly to assure routability. This often leaves small gaps between cells, and it is impossible for EDA tools to fill 100\% of the area with regular standard cells in VLSI designs. In practice, after completing placement and routing, all unused spaces will be filled with filler cells or decoupling capacitor (decap) cells; these cells do not have any functionality. For intelligent attackers, the most covert way to insert Trojans in a circuit layout is by replacing filler cells because removing these non-functional filler cells does have the smallest impact on electrical parameters. If attackers redesign the original layout for Trojan insertion, moving gates' locations and altering wire interconnections will result in significant changes of the electrical parameters, such as power and path delay. These can be detected much more easily by delay-based [13] and power-based techniques [5] [6].

The principal idea of BISA is to fill all unused spaces with functional standard cells, called BISA cells, instead of conventional non-functional filler cells. BISA cells are connected to each other to construct a combinational circuit that is independent from the original circuit. This combinational circuit can have an arbitrary function. By testing the function of the BISA circuit, test engineers are able to check whether these cells have been altered after fabrication. If the adversary inserts a Trojan by changing, removing or modifying any cell in a BISA circuit, the designer can easily detect it using structural test. To cover all BISA cells, we try to construct a combinational BISA circuit without redundant gates; in other words, all gates in the BISA circuit are testable by test patterns that target stuck-at faults. For a design with BISA, Trojan cell insertion will become practically impossible without tampering with the BISA cells. Furthermore, BISA cells

are of the same type of standard cells as in the original circuit, so identifying these cells will be extremely challenging if not impossible.

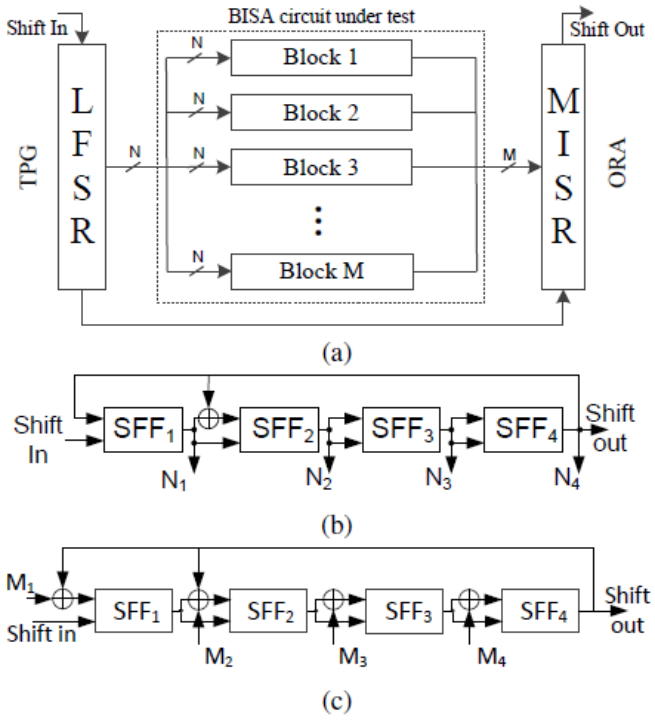


Figure 6: The structure of (a) BISA, (b) a 4-stage LFSR, and (c) a 4-stage MISR.

The BISA structure is intended to detect any abnormal alteration on BISA by producing a different signature than expected. As to how to test the BISA circuit with low overhead, our inspiration came from the logic built-in self-testing (BIST) for VLSI circuits. The BIST was developed to apply random test sequences to determine the correctness of a circuit by examining a signature obtained from the test responses [16]. Basically, BISA consists of three parts: the BISA circuit under test, the test pattern generator (TPG), and the output response analyzer (ORA), as shown in Figure 6(a). The BISA circuit under test is composed of all BISA cells that have been inserted into unused spaces during layout design. In order to increase its stuck-at fault test coverage, the BISA circuit is divided into a number of smaller combinational logic blocks, called BISA blocks shown in Figure 6(a). Each BISA block can be considered as an independent combinational logic block. The TPG generates test vectors that are shared by all BISA blocks. The ORA will process the outputs of all BISA blocks and generate a signature. In this work, we use a LFSR as TPG and MISR as ORA. Examples of 4-stage LFSR and 4-stage MISR are shown in Figure 6(b) and (c). They are used to generate random vectors and compress responses to a signature. SFF in the figure represents scan flip flop. Other types of TPG and ORA can also be applied [16].

There are two operating modes associated with the chip, as shown in Table 1. In functional mode, the original circuit is working normally, but the BISA is in idle mode by disabling LFSR and MISR. BISA stays quiet and does not affect the original circuits, but can fulfill the role of decap cells (see Section 3.3.2). Authentication mode consists of two phases. During the test phase, the LFSR generates N-bit test patterns at every clock cycle

and the N-bit test patterns are shared by all BISA blocks. At the same time, each BISA block outputs one bit so that the MISR receives a total of M bits from M blocks (see Figure 6(a)). The test phase ends until a sufficient number of test patterns have been applied. The number of test patterns is decided in simulation at the BISA design phase, according to a target test coverage. At this time, the value in the MISR is the signature generated from the responses of BISA blocks and it can be shifted out through scan chain. We refer to this as the shift phase. Comparing the obtained signature on a fabricated chip with the correct signature from previous simulations, we would then know whether the BISA structure has been tampered with. Since all the registers in LFSR/MISR and other registers in circuit are connected in a chain, in the shift phase, the signature in MISR can be shifted out while, at the same time, the new seed for LFSR is shifted in if more patterns need to be applied.

In BISA, two control signals are needed to manage the system and trigger the authentication stage. An authentication mode selection (AMS) signal from the primary is required to choose between functional mode and authentication mode (as depicted in Table 1). Therefore, one extra input pin for this signal is required. In the authentication mode, the system needs to switch between shift and test phase. The test mode selection (TMS) signal generated in design-for-test for controlling scan chain can be used here as well, so no additional pin is added. Note that an external clock is applied in all modes. A particular clock will be generated by a tester or other external clock generators for every mode of BISA. Typically, shift clock and authentication clock are much slower than functional clock]16]. The authentication clock frequency is related to the number of gates in a BISA block and the gate types. As the gate number goes up, the depth of the tree structure circuit increases. After BISA design, the depth of every tree structure circuit is determined and known. The timing constraint can be obtained using static timing analysis tool to find the longest path. In the authentication mode, a very slow test clock can be used to ensure no timing violation occurs in BISA circuit. For example, suppose a tree-structure BISA circuit has a logic depth of 10 and up to 55 gates. If the propagation delay of each stage is 500ps, the total delay is 5ns (200MHz). A test clock slower than 20MHz could be used for the authentication step. With such slow speed, the authentication would still finish in few milliseconds.

Table 1: Operation modes in a design with BISA.

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

3.2 BISA Design Flow

The BISA technique is developed to address insertion of Trojans by untrusted GDSII developers and foundries. We assume that the design has passed all necessary verifications at design phase, so it is genuine before being shipped to GDSII developer. Figure 7 shows the proposed BISA design flow and where it fits within the conventional ASIC design flow. The white rectangles in the figure are steps taken in a conventional ASIC design flow, and the grey ones are the additional steps for inserting BISA circuitry. We will describe them in detail in the following subsections.

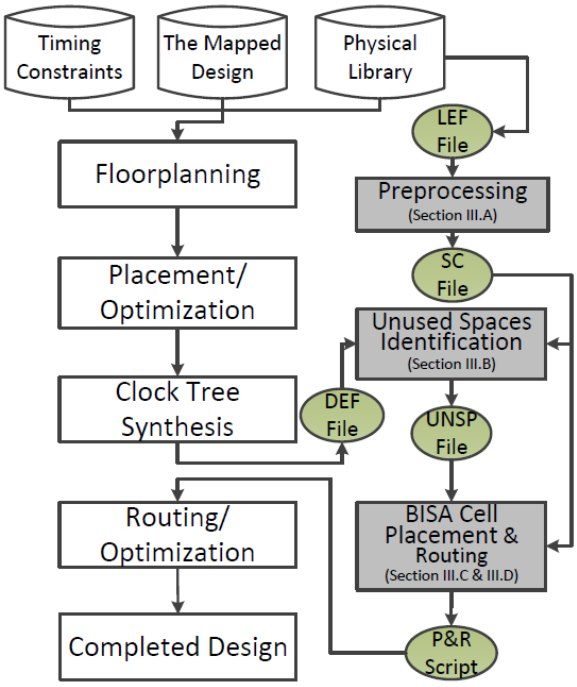


Figure 7: BISA design flow.

i. Preprocessing

First, some information of standard cells in the technology library needs to be known and stored for later steps. Dimension of each standard cell, such as length and width, is needed for identifying and analyzing unused spaces. Once the placement direction and location of a particular cell are given, the coordinates of its four corners can be calculated by the cell's length and width. Additionally, the number of input pins and the name of a cell are saved for later usage. After obtaining the necessary information for all standard cells, BISA cells will be selected from them and marked according to the following criteria: 1) BISA cells must be the minimum-size cell for every logic, so they are resistant to a resizing attack by the adversary (see Section 3.3.3). 2) The amount of decoupling capacitance the cells can provide and the input count should be considered as well. The normalized input count represents the number of inputs of a standard cell if the same cell has the same area of the minimum-size cell (e.g., INVx0 in Table 2). Fewer inputs help to improve test coverage, which will be explained in Section 3.2.4. 3) The smallest cell in the library must also be included in order to ensure that no cell can be inserted in any remaining unused space. Table 2 gives an example generated with Synopsys 90nm library. The columns labeled *area* and *input* show each cell's area and input count. The normalized input count is listed in the fourth column. Once a cell is selected, it will be marked in the last column of the table. A cell with fewer inputs and a larger decoupling capacitance has higher priority to be a BISA cell. Note that cell INVx0, NAND2x0, and NOR2x0 are three smallest cells in this library. Since two inverters could potentially connect in serial to form a buffer and the buffer is redundant logic, INVx0 would not be chosen as a BISA cell. On the other hand, the NAND2x0 cell and the NOR2x0 cell would be selected to ensure BISA circuit contains the smallest cell.

Table 2: Cell information collected at preprocessing step.

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

A program has been developed to acquire all the above-mentioned information from the LEF file in technology library and output an in-house Standard Cell (SC) file format. This SC file must be ready before starting BISA design flow. Figure 7 also shows different input files needed and output files generated when going through the BISA insertion process. Other files (DEF, UNSP, P&R Script) are described in the following subsections.

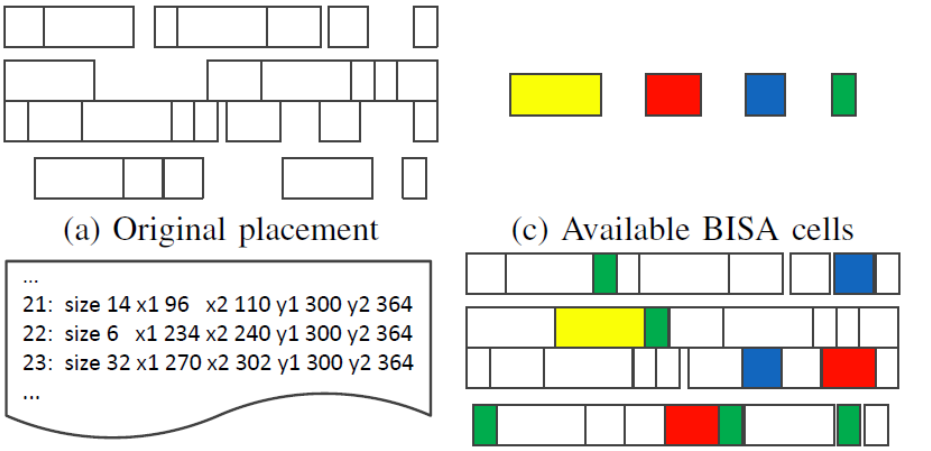


Figure 8: BISA cell insertion and placement.

3.2.1 Unused Space Identification

In the BISA design flow shown in Figure 7, IC designers perform floorplanning, placement, and clock tree synthesis just as they do in conventional layout design. After finishing clock tree synthesis, the original circuit has been placed in the layout, so the flow starts to search for and locate all unused spaces of the layout. We utilize a matrix, which is similar to bitmap in computer graphics, to record the state of each point in the layout. The initial state '0' represents "empty". If one cell occupies this point in the layout, the state is switched to '1'. Therefore, every standard cell placed in the layout will be processed one by one and eventually the bitmap reveals the location and size of unused spaces. To obtain the location of every placed cell, layout design tools such as Synopsys IC Compiler can write a DEF file (.def) that contains coordinates of all placed standard cells. By analyzing coordinate (.def) and its corresponding size (.sc) of every placed cell, unused spaces in layout can be identified. A Perl script has been developed to read the DEF and

SC file, locate used spaces and output an Unused Spaces file (.unsp). The format of the UNSP file is shown in Figure 8(b). The required information, such as the size and coordinates of the four corners, are listed in the UNSP file.

3.2.2 BISA Cell Placement

After unused space identification, the flow completely fills the unused space with BISA cells selected at the preprocessing step. A dynamic programming algorithm is employed to find an optimal solution (see Section 3.2.5 for details). The algorithm is implemented in a program which simulates the cell insertion process and produces a Tcl command script for EDA tools. Available BISA cells and the layout after applying BISA is shown in Figure 8(c) and (d). Note that once BISA has been applied, there may still be some spare spaces left between cells, but not even the smallest cell (green) can be inserted. Therefore, attackers cannot insert any extra cells either.

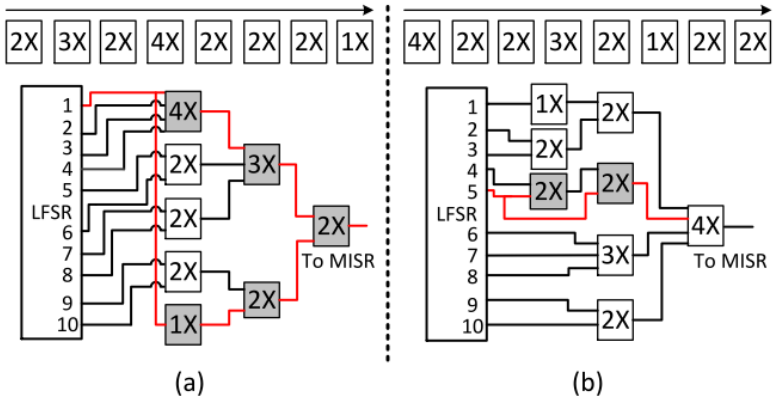


Figure 9: Routing a BISA block.

3.2.3 BISA Cell Routing

After unused space identification, the flow completely fills the unused space with BISA cells selected at the preprocessing step. A dynamic programming algorithm is employed to find an optimal solution (see Section 3.2.5 for details). The algorithm is implemented in a program which simulates the cell insertion process and produces a Tcl command script for EDA tools. Available BISA cells and the layout after applying BISA is shown in Figure 8(c) and (d). Note that once BISA has been applied, there may still be some spare spaces left between cells, but not even the smallest cell (green) can be inserted. Therefore, attackers cannot insert any extra cells either.

- First, we try to create as many BISA blocks as possible to make each BISA block with fewer gates so that higher test coverage is easier to achieve. Since the output of every BISA block will connect to MISR, the number of BISA blocks is determined by the size of MISR. If M is the size of MISR, all placed BISA cells are divided into M groups (BISA blocks).
- Second, redundant gates could deteriorate controllability and observability of the circuit and lower the test coverage significantly, so a tree-structure circuit is proposed to eliminate redundant gates, as shown in Figure 9. If every input is independent of other inputs in a tree-structure circuit, every net is controllable and observable, so the theoretical test coverage of stuck-at fault is 100%. Here, we construct a tree-structure BISA block according to the sequence of cells in a block set.

Figure 9 shows that two different sequences lead to two different tree-structure circuits. The first gate becomes the root of the tree-structure circuitry, i.e., it is on the top (first) level of

tree. The outputs of the next *x* cells (*x* being the number of inputs of the root cell) are connected to its inputs as its children cells, on the second level. On the third level, we do the same to connect new cells to cells on the second level. Cells are sequentially connected to cells on upper levels until all of them are processed, as shown in Figure 9 (a) and (b). After complete routing in each block, all inputs of each block should connect to LFSR sequentially to avoid sharing inputs. In the end, the M bits output from M BISA blocks connect to a MISR with size of M.

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

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

Consider the circuit in Figure 9 as an example. The total number of inputs is *S=18(sum of cells' inputs)-8(cell number)+1=11*. Although Figure 9(a) and (b) show two different tree-structures, their input counts are same. As more cells are added to one block, the number of inputs will increase consistently. If the number of inputs of one block is greater than what LFSR can provide, some inputs have to share LFSR outputs in a broadcasting fashion (i.e., using fanout). The correlation among these shared inputs could potentially result in redundant gates in BISA blocks, thereby affecting test coverage. Let us take the BISA block in Figure 9 (b) as an example. This BISA block has 11 input pins, but the size of LFSR is 10, which is 1 bit less than the BISA block. Thus, one input pin has to share with another input pin in this BISA block. In Figure 9 (a) and (b), all nets with correlation due to the shared input are highlighted in red. The input correlation can impact controllability or observability of the dark gates in the figure. One straightforward way of eliminating redundant gates is to change the tree structure to reconstruct the BISA block by adjusting the cell order, since the stuck-at fault test coverage is highly related to the circuit structure, If one slightly changes the sequence, the structure of BISA block will completely change and thus the test coverage will vary significantly. After trying different sequences in one BISA block, the sequence that has the highest test coverage is kept.

2) **BISA Design Strategy**: Two BISA design methods, static and dynamic, are developed in this work. In the case of static BISA insertion, LFSR and MISR are included in the original design. After inserting BISA cells, the BISA circuits are connected to the pre-designed LFSR and MISR. However, static BISA design offers a couple of potential challenges. First, if the original design does not include LFSR and MISR (i.e. does not use built-in self-test (BIST)), BISA cannot be established. Additionally, if the BISA circuit is larger than the testing capabilities of pre-designed LFSR and MISR, sharing fanout in a BISA block will lower the test coverage and further decrease the reliability of BISA. In order to address this problem effectively, a dynamic BISA design algorithm is proposed to insert both BISA cells and LFSR/MISR into unused space and adjust their composition to ensure LFSR and MISR can fully test the BISA circuitry. The detailed algorithm will be described in Section 3.2.5. In general, the dynamic BISA design method is generally very effective for designs with many large unused spaces, like SOC design. Its shortcoming is that it may not be as effective for very compact designs (those with little unused space available), which will be discussed in Section 3.4.5.

3) **Routability**: Additional connections will be introduced by BISA circuits and will take some routing resources from the routing of the original circuit. In order to ensure routability of the original circuit and reduce the routing resources taken by the additional BISA circuit, we usually put the nearest cells in one BISA block to shorten their interconnections. Additionally, BISA can save more routing resources by sacrificing its own speed. The minimum size for metal wire and high metal layers could be used for BISA routing so that more room can be created for

routings of the original design. Correspondingly, a slow test clock is used for BISA in authentication mode. This allows us to ignore BISA's timing constraints and focus on obtaining maximum coverage with limited routing resources.

At this step, our program can generate a routing script for Synopsys IC Compiler to create nets and connect them logically, as shown in Figure 7. The whole design, including BISA, will be physically routed at routing/optimization step. Once the timing and sign-off of the design are successful, the last step involves the generation of a GDSII/OASIS format of the design for final tape-out.

3.2.4 Place & Route Algorithms

1) **Dynamic Programming Algorithm**: During BISA placement, a method is needed to completely fill each unused space using BISA cells with the fewest total input count. Minimizing the total input count can not only curtail the routings required in BISA circuitry, but also lower the requirements for a LFSR and MISR. In this work, we take a dynamic programming approach for BISA placement. Dynamic programming is one of the classic heuristic algorithms, where a complex problem is solved by breaking it down into smaller sub-problems [17]. The optimal solution of the current problem is based on the optimal solution of its sub-problems. When applicable, this method takes far less time than heuristic search methods that do not take advantage of the sub-problem overlap (like depth-first search). For the problem of filling each unused space, all potential solutions can be derived from adding a BISA cell to a corresponding smaller space that has been solved. Comparing all the possible solutions, we will choose the filling solution with the fewest inputs. A simple example is illustrated in Figure 10. We are trying to fill an unused space with the size of *L*. Assume the optimal solution for any unused space smaller than *L* (≤*L*) has already been solved. Suppose that a total number of four BISA cells (B_i, *i=1,2,3,4*) are available, their sizes are *d1, d2, d3*, and *d4*, and their input count are *k1, k2, k3,* and *k4* respectively, as shown in the brackets in the figure. The optimal solution (*S(L)*) for filling the unused space (*L*) must therefore be from the solutions of a BISA cell B_i plus a sub-solution S(Ui) of the corresponding unused space U_i (*i=1,2,3,4,* U_i=*L*-d_i). Definitely, unused spaces *U1, U2, U3,* and *U4* are smaller than *L* and are *L*'s sub-problems, so the optimal solutions to fill them are already obtained. From these four potential solutions, the one with the fewest inputs will be selected. In the example, the solution of the BISA cell B_4 plus the U_4 is selected for the space of size *L* because its input count (9, $S(U_4)+d_4$) is smaller than others' (12, 11, 12). Next, a larger unused space will be explored until the maximum area of the unused space has been investigated. Our dynamic programming algorithm can be presented visually as

$$S(n) = Min[S(n-d_m)+k_m, m \in \text{BISA cells}] \quad \textbf{(2)}$$

Where *n* is area of current space and *0≤n≤L*.

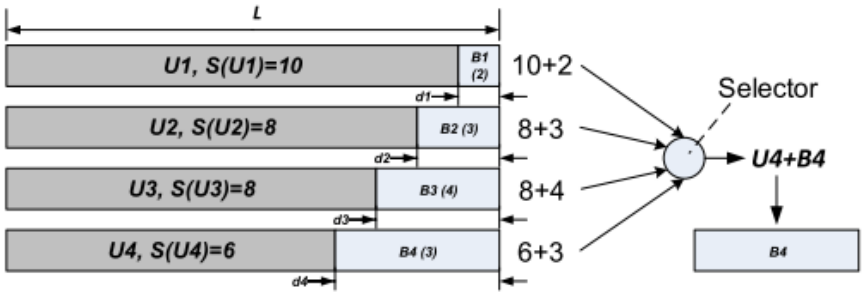


Figure 10: An example using dynamic programming.

The dynamic programming approach seeks to solve each sub-problem only once, thus reducing the number of computations: once the solution to a given subproblem has been

computed, it is saved for later usage. The method can save significant computation time during the placement process because there are usually many unused spaces having the same size. Additionally, a dynamic programming algorithm will examine all possible solutions to the problem and will pick the best one, so it guarantees finding the optimal solution. Alternatives, such as the greedy algorithm used in [18], cannot guarantee the optimality of the solution.

2) **Dynamic BISA Design Algorithm**: The previous static BISA design algorithm [18] is not suitable for a design without pre-designed LFSR or MISR or a design with many large unused spaces, as discussed in Section 3.2.4. Thus, a dynamic BISA design method is necessary, in order to ensure high test coverage. Therefore, all BISA cells, scan flip-flops and XORs for LFSR/MISR will be inserted into unused spaces during the dynamic BISA design. Since unused spaces in layout are fixed, the key point is balancing the number of BISA cells for BISA circuit and the number of scan flip-flops inserted for LFSR and MISR in certain unused spaces. If most of the unused spaces are filled with BISA cells and the size of LFSR/MISR is small, such an LFSR/MISR cannot provide or receive long-enough vectors and thus test coverage will not be very high. If LFSR and MISR are much larger than the corresponding inserted BISA cells, the excessive scan flip-flops could result in a huge power consumption during BISA testing because of switchings in a large number of scan flip-flops at every clock cycle. Thus, the size of LFSR/MISR and BISA circuit should be balanced, in order to achieve 100% test coverage with smallest size of LFSR/MISR. The dynamic BISA design algorithm, depicted in Algorithm 1, has been developed to address this problem. From line 1 to line 3, required information produced at previous steps--like BISA cell size, unused space area, and polynomial primitives of LFSR/MISR [16], etc.--is imported. The dynamic programming algorithm described in Section 3.2.5 finds solutions for all unused spaces smaller and equal to the maximum unused space (line 4). On the line 5, unused spaces are filled with scan flip-flops only as much as possible. Then, the corresponding number of XORs (in the polynomial primitive file) needs to insert. Remaining spaces are then filled with BISA cells, as described in line 7. Next, the number of BISA cells and their inputs is obtained (line 8). Once the BISA cell count, input count, and the size of LFSR and MISR are known, one can determine if current LFSR and MISR are able to fully test BISA circuits. We define a *test index* to represent the testability of current BISA circuits, which can be calculated using the Equation (3).

$$\text{Test Index} = (\text{Sum of Cell Inputs})-(\text{Sum of Cells})-(\text{MISR Size})*((\text{LFSR Size})-1) \quad (3)$$

A positive integer of the test index means that current LFSR and MISR can fully test the BISA circuits, so the process can move ahead to shorten LFSR/MISR and place more BISA cells. The following loop tries to clear the unused space that has the most scan flip-flops and refill it with BISA cells (line 11-15). If the new test index is still greater than a threshold, the action will be approved (line 16). All the BISA placement information is updated, and then the process moves forward to try the next unused space with the most flip-flops (line 17-20). If the test index is smaller than the threshold, this action is unsuccessful. The reason is most likely that the size of LFSR/MISR decreases too much, because excessive scan flip-flops for LFSR and MISR are removed. Therefore, a new loop goes back to line 11 and begins to explore the unused space with the second greatest number of flip-flops. The process will be repeated until all unused spaces with flip-flops have been investigated. Finally, the test index will be a value greater than but very closed to the threshold. It means LFSR can offer more inputs than input count of every BISA block, so there is no redundant gate in the BISA circuitry.

Algorithm 1 The dynamic BISA design procedure.

```
1: Read SC and UNSP files
2: Read polynomial primitive [30] for LFSR and MISR
3: Read dynamic programming filling results
4: Insert scan flip-flops to fill all unused spaces
5: Insert appropriate XOR gates, remove scan flip-flops if
   necessary
6: Insert BISA cells to fill remaining unused spaces
7: Count current BISA cell input # and BISA cell #
8: Calculate Test Index
9: while (Test Index > threshold) do
10:     Search an uninvestigated unused space with most flip-
   flops
11:     Clear the unused space (remove scan flip-flops, XORs
   and BISA cells in the unused space)
12:     Insert BISA cells to fill remaining unused spaces
13:     Count current BISA cell input # and BISA cell #
14:     Update Test Index
15:     if (Test Index > threshold) then
16:         Update inserted BISA cells
17:         Update scan flip-flops and XORs
18:         Update LFSR and MISR
19:         Update Test Index
20:     end if
21: end while
```

3.2.5 BISA Design in System-on-Chips (SoCs)

In this section, we extend the BISA framework to System-on-Chips (SOCs). SOC is typically a bottom-up hierarchical design, and the top module includes other pre-designed sub-modules, or what are known as intellectual property (IP) cores. We assume that each IP core in the SOC has been implemented with BISA already by the flow described in Section 3.2. In this case, each IP core can be simply treated like a standard cell, so BISA design in the top module of a SOC is nearly the same as that in a single-module design. The main difference is that IP cores are much larger than standard cells and cannot be placed in a row. In addition, we also have to determine how to organize the LFSR/MISR. For an SOC design, therefore, we just need to make some small changes on the step of unused spaces identification, and other steps are exactly same.

Since BISA will be applied to all modules in a hierarchical design, three topological structures (distributed, centralized, and hybrid) to organize LFSR/MISR are illustrated in Figure 11. “Distributed structure” means each IP core and top module has its own LFSR/MISR, as shown in Figure 11(a). Two additional pins are reserved in each sub-module for shifting data in shift mode. Although distributed structure has simple wire interconnections and requires fewer pins, having these independent LFSRs/MISRs in every module could result in the larger area overhead. In “centralized structure”, only one centralized LFSR and one centralized MISR (in the top module or one of sub-modules) are used to generate test patterns and compress responses for BISA blocks in both the top module and other IP cores. Centralized structure can potentially increase the complexity of BISA cells routing, requires more pins than others and could potentially increase area. Each module will use *N+M* pins to receive N-bits test vectors and output M-bits

responses. “Hybrid structure” is, as the name suggests, an amalgam of the two above structures. If some sub-modules have dedicated LFSR/MISR and some do not have due to limited IP core resources, “Hybrid structure” is more flexible.

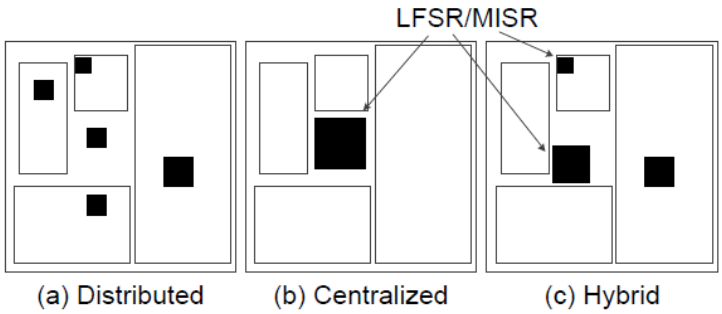


Figure 11: Three possible structures used to organize LFSRs/MISRs in an SoC design.

3.3 Feasibility and Reliability of BISA

3.3.1 ECO

In chip design, Engineering Change Order (ECO) [19] is the process of making some small changes in the processed design. Before the chip masks are made, ECOs are usually done to save time by avoiding the need for full ASIC logic synthesis, technology mapping, layout design, layout extraction, and timing verification. After masks have been developed, ECOs could be performed to save money. If a change can be implemented by modifying only a few of the layers of a design, then the cost is much less than it would be if the design was rebuilt from scratch. Our proposed BISA technique does not impact either pre-mask or post-mask ECOs. All inserted BISA cells can be used as spare logic gates. When a BISA cell is selected for ECO, the BISA cell needs to disconnect from its BISA circuits. Then another two wire connection changes are required to skip the selected BISA cell and reroute the original BISA circuits. Since BISA circuits do not have timing constraints, these two routing changes can be implemented using limited space. Moreover, our BISA design algorithm will fill unused spaces with different types of gates and ensure the variety of BISA cells.

3.3.2 Filler Cell and Decoupling Capacitor Cell

In conventional layout design, after placement and routing, unused spaces will be filled with filler cells or decoupling capacitor (decap) cells in order to reduce the design rule check (DRC) violations created by the base layers and to ensure power rail connection. Moreover, filler cells and decap cells are able to provide power and ground capacitances, which can effectively minimize power supply noise. However, filler cells and decap cells do not have functionality, so they cannot be tested and can be easily identified and removed by attacker. Although no space is left for decap cells after applying BISA, the BISA cells are able to provide decoupling capacitance to some degree. For decap cell, the decoupling capacitance comes entirely from dedicated gate oxide capacitance sources, but BISA cell utilizes its diffusion capacitance for the decoupling capacitance. Typically, diffusion capacitance is less than gate capacitance. However, N. Weste and D. Harris [20] pointed out that the diffusion capacitance C_{sb} and C_{db} of contacted source and drain region was comparable to the gate capacitance. In the BISA design, BISA cells are connected together to form a combinational circuit, so most source/drain diffusions are contacted for making connections. Therefore, BISA cells are able to supply decoupling capacitance to functional cells nearby. To illustrate their power and ground capacitance, we use the cell characterization function in HSPICE to report the node capacitance of standard cells. Table 3 lists the power (C_{VDD}) and ground capacitance (C_{VSS}) of functional standard BISA cells and

filler cells in two academic standard cell libraries. Top portion for each library corresponds to the BISA cells while the bottom portions correspond to the filler and decap cells. The last column in the table shows the power and ground capacitance per unit area. We observe that standard cells do not lose much power/ground capacitance, and even some BISA cells have larger capacitance. For example, in the NANGATE 45nm standard cell library, NAND2_x1 and NOR2_x1 can offer larger average capacitance than its filler cells. Note that the source/drain capacitor is operation mode dependent [20]. When the system is in functional mode, the BISA circuit will be idle and all BISA cells are in a pre-specified mode depending on the value in the LFSR. Some cells are in the mode where they can provide the maximum decaps, while others are may not be. Since different types of BISA cells are inserted at the BISA placement step, a large number of inserted BISA cells and their diversity can ensure that average decoupling capacitance is relatively stable. Analyzing this shall be part of our future research plan.

Table 3: The power and ground capacitance of filler cells and a portion of functional standard cells in two academic standard cell libraries.

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

3.3.3 Potential Attacks

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

1) **Removal attack**: Removal attack is the most direct and simplest way to create space for Trojan gates. Simply removing BISA cells will change the functionality of BISA blocks. Test patterns will test the functionality of BISA blocks, and BISA signatures can tell if the BISA cells have been tampered with. If some functional standard cells from the original design are

removed by adversaries, the original functionality will be altered. Both functional and structural tests are able to detect removal attacks. We recommend using as few unnecessary non-functional standard cells, such as buffers, as possible. Even if some buffers are deleted and cannot be detected, the limited space left from removing buffers would still hamper Trojan insertion.

2) **Redesign attack**: If the adversary changes one or more gates that alter the functionality, it will be detected by functional tests regardless in BISA circuit or the original circuit [21]. If the adversary is forced to redesign the layout in order to achieve the same functionality and insert Trojan gates as well, then the chip dimensions will probably alter. In addition, these alterations could result in changes on placements and routings for some or all design components and their interconnections. Modifications to original circuitry would impact the circuit timing that could be detected using side-channel based techniques [5] [6]. Therefore, the redesign attack could be more effective on BISA than original circuit, because BISA is redundant circuit in functional mode and does not impact circuit performance. However, there are major difficulties making it practically impossible. First, it is nontrivial for the foundry to differentiate the original circuit from BISA in order to create the space for hardware Trojan insertion. BISA cells are the same standard cells as in the original design. BISA cells form combinational circuits and are connected to scan flip-flops of LFSR/MISR, which look like a standard BIST design. The foundry would need to analyze the entire layout to identify the BISA cells. While, this is not necessarily impossible, it does require a significant amount of effort and expertise on the part of the attacker to insert a Trojan compared to conventional approaches that do not use BISA. Second, should an adversary actually identify BISA cells, they must then analyze the BISA circuitry and find a functionally equivalent logic with a smaller area. Finding a feasible solution for a tree-structured circuit is another challenge. For example, two 2-input NAND gate converging in another 2-input NAND gate can be replaced by a 2-input OR gate and two 2-input AND gate. The area of the NAND gate implementation is 16.59 (μm^2) using the SAED90nm standard cell library, but the OR gate plus AND gates need a space of 22.12 (μm^2). Thus, it cannot be used to create unused spaces for inserting hardware Trojans. The most possible way is that some certain gates can be function-equivalently replaced by an available complex standard cell in standard cell library, such as AO22x1, AOI221x1 and etc. Since these complex cells are well designed and compacted as much as possible, such as sharing diffusion areas and sharing connection wires. Usually, the complex cell has a smaller area than its functionally equivalent circuit comprised of basic cells (like NAND, NOR, AND, and OR). For instance, the area of an AO22x1 cell (11.98 μm^2) is smaller than the area of a 2-input AND cell plus a 2-input OR cell (14.74 μm^2). Therefore, these logics that complex cells have should be avoided in a tree-structured BISA block. If found, a new logic can be formed by slightly changing cell's order in the BISA block. Moreover, this attack can also be made more challenging if the BISA cells that are connected are not directly neighboring. Even if there is an equivalent circuit, the area might be larger than any unused space created by removing any one of these distributed BISA cells. Taking the layout in Figure 3 for example, we suppose that a yellow cell can replace two blue cells without change in functionality and its size is smaller than total size of two blue cells. Since a yellow cell is larger than one blue cell, the new yellow cell cannot be inserted into any places where two original blue cells locate without touching original circuit. In addition, some techniques can be used to prevent the logic minimization at the BISA design stage. Synthesis tools have the ability to minimize logic and can be used to verify if a BISA block is the implementation with the minimum area. If not, one can reorder the sequence of gates to form a new tree structure BISA block until it cannot be optimized. This step does not change the placement of BISA cells, but it needs a longer processing time and modifies routing in the BISA block. Third, if the attacker cannot simply

replace the BISA cells with a functionally equivalent cell with smaller area, the only alternative is to begin modifying the original design as well. As mentioned in the work, BISA does not address the issue of redesigning the original circuit. However, modifications in the original circuit could impact the performance of the original design and change its side-channel (delay, power, etc.) behavior. While an attacker could overcome the obstacles above with enough effort, Trojan insertion is still significantly hampered with BISA compared to other approaches.

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

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

3.3.4 Yield

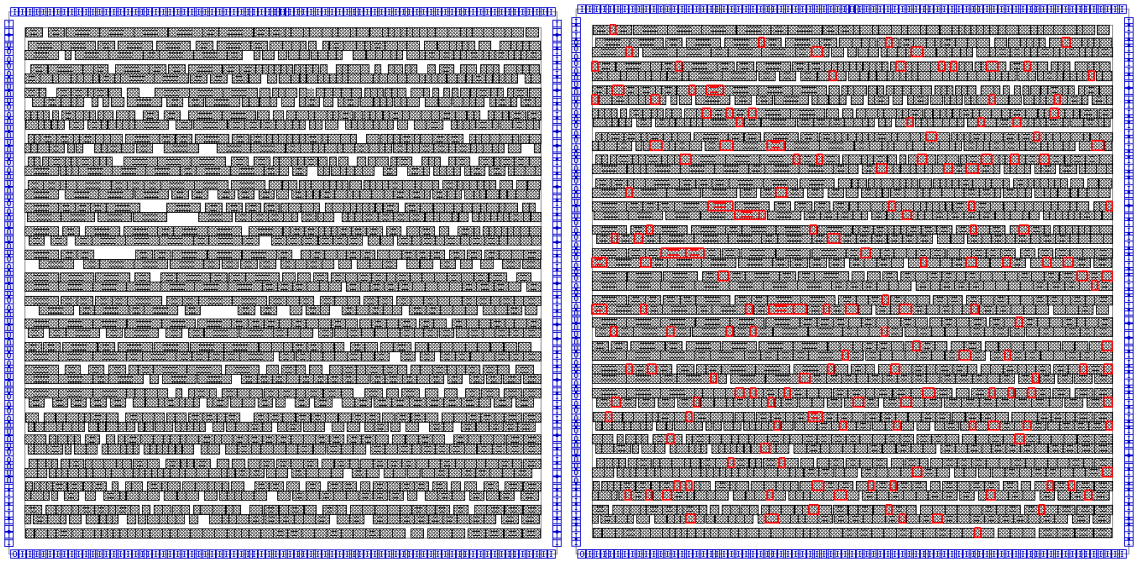
The discussions above rely on the fact that BISA is genuine and working without manufacturing defects. However, BISA contains LFSR, MISR and many BISA cells which may also be affected by silicon defects during fabrication. One chip producing a faulty signature may contain hardware Trojans, but it also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

3.4 Results and Analysis

3.4.1 BISA Implementation

In this section, we implement BISA on 15 designs from various benchmark suites. Four of them (OpenSparc, Leon3mp, Netcard and VGA_lcd) are SOCs containing hierarchical sub-modules and the others are single-module designs. All benchmarks are synthesized using Synopsys Design Compiler with Synopsys 90nm technology library. Overall, 32 standard cells

were selected as BISA cells from the 320 in the library and each cell has a unique function. Synopsys IC Compiler has been employed for layout design. For our implementations of BISA, a series of programs have been developed to extract standard library information, to identify unused spaces, and to generate placing & routing scripts for the IC Compiler. Figure 12 shows the layouts of benchmark DES3_area with the core utilization of 70%. Before applying BISA, we can see many unused spaces in Figure 12(a). In Figure 12(b), all these unused spaces are filled with BISA cells, highlighted in red, after BISA insertion. Although there still are some spare spaces left, these remaining spaces are too small to place a cell. Another benefit of these left spare spaces is that they can be routing channel for low-layer metals, especially for critical paths.



(a)Before BISA Insertion (b) After BISA Insertion

Figure 12: Implementation of a signal module design.

Figure 13 shows the implementation of one Sparc core in OpenSparc T1 benchmark. The Opensparc T1 is the first 64-bit open-source microprocessor, released in 2006 by ORACLE. The Opensparc core is composed of seven sub-modules (hard macros): exu, ffu, ifu, lsu, mul, spu, and tlu. The layout floorplan and proportion of every sub-module are depicted in Figure 13(a) and (b), respectively. We try to compact the floorplan as much as possible in the layout, but gaps still inevitably exist between sub-modules. With the implementation of BISA, BISA circuit can perfectly fill all these gaps. Since the OpenSparc benchmark has more than half million gates, it is very difficult to see those inserted BISA cells. To show the BISA circuitry between sub-modules, three screenshots at different locations in the layout are shown in Figure 13(c)-(e). The black parts are cores (sub-modules) and the red components represent the inserted BISA cells.

Similarly, BISA has been applied to all benchmarks, and their implementation results are presented in Table 4. Compared to single-module designs, SOC designs need many more BISA cells. Therefore, in these SOC designs, a large dynamic LFSR and a MISR are generated in these unused spaces to ensure a very high test coverage without any area overhead. It is worth nothing that the size of the BISA circuit is not proportional to the original circuit and it is related to the circuit structure and its timing constraints. For example, the AES_core benchmark is half the size of the DES_perf benchmark, but its BISA circuit is more than twice as large as that in the DES_perf using the same core utilization.

Figure 13: Implementation of the OpenSparc microprocessor core.

Table 4: BISA implementation.

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

3.4.2 Filling Approach

A filling approach using the dynamic programming algorithm has been developed in Section 3.2.5. Theoretically, this algorithm can produce an optimal filling solution with the fewest number of inputs. Four randomly picked unused spaces have been investigated. The greedy algorithm (one classic heuristic algorithm) used in [18] has also been employed to fill these four spaces with the same set of BISA cells. Table 5 is the comparison results between the two algorithms under the same condition. Rows 5 and 6 show the total input count of inserted BISA cells and the leftover space after BISA insertion, respectively. In all four cases, the dynamic programming algorithm (DP) outperforms the greedy algorithm in the total input count and therefore should obtain lower LFSR overheads and higher test coverage more easily. In all but one case (case 3), the dynamic programming algorithm leaves the same amount of spare space as the greedy algorithm. In case 3, it leaves significantly less space. Therefore, the dynamic programming algorithm is at least as good as the greedy algorithm in filling unused spaces, if not better.

Table 5: Algorithm performance.

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

3.4.3 Test Coverage Analysis

As described in Section 3.2.4, two factors can influence the eventual test coverage of a BISA circuit: the testability of the BISA circuit and test patterns. The testability of the BISA circuit is determined by the number of redundant gates at the BISA design phase. As LFSR and MISR become larger, their testing capabilities enhance and the number of redundant gates goes down. Figure 14 shows testability of the BISA circuit in benchmark USB_funct by performing fault simulation in Synopsys TetraMax. Different sizes of LFSRs, such as 32-bit, 16-bit and 8-bit are investigated. As we expected, the test coverage goes up gradually as LFSR and MISR become larger. The testable coverage is a theoretical test coverage that can be achieved, but the actual test coverage also is dependent on test patterns, i.e. the number of random patterns generated from LFSR.

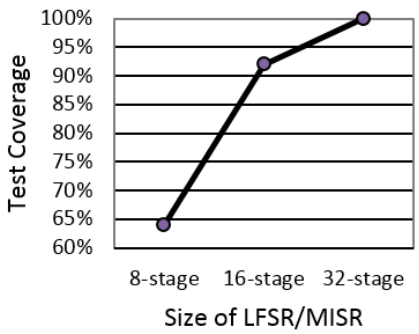


Figure 14: Testability of BISA circuit with three types of LFSR/MISR.

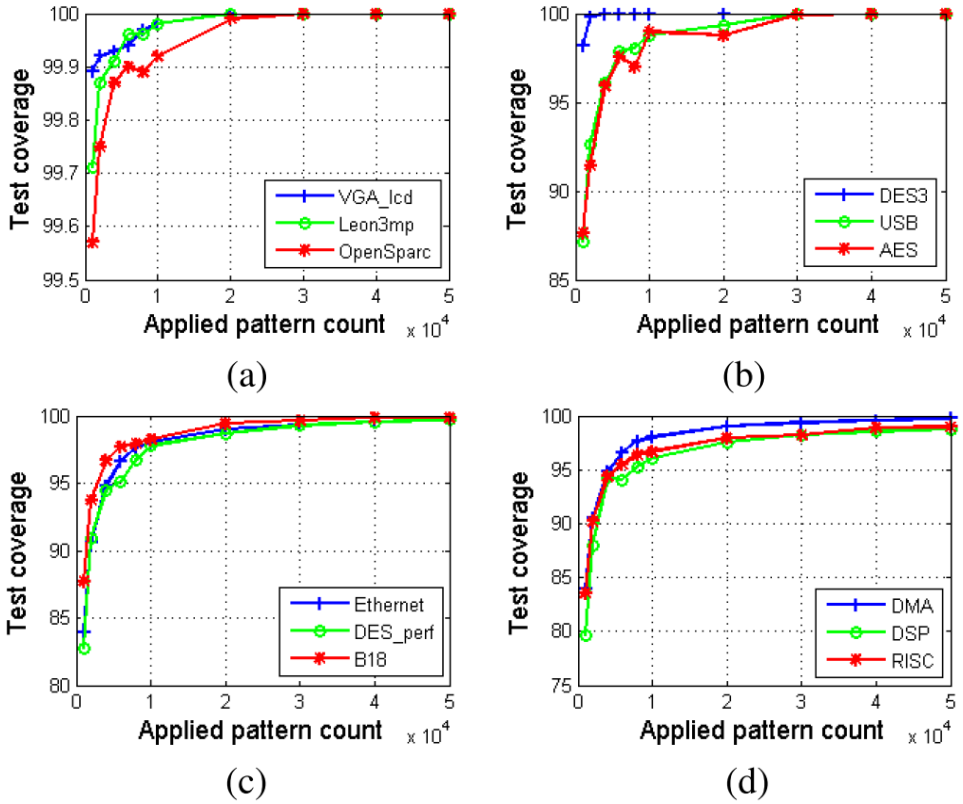


Figure 15: Test coverage over a different number of test patterns.

The test coverage over a different number of test patterns are illustrated in Figure 15. As more random test patterns have been applied, the test coverage increases and then gradually goes to saturation in all 12 benchmarks. The top left figure shows the results of three SOC designs. Although these SOC designs have more BISA cells and larger BISA circuits, compared to other single-module benchmarks, their test coverage can be 100% after applying 30,000 random patterns. For other single-module designs with smaller BISA circuits, their test coverage is lower than the SOCs', which defies our intuition. This is because, for the BISA circuits in SOCs, XOR gate is much more than other types of gates, such as AND, OR, NOR, and NAND. A XOR gate can produce half of ones and half of zeroes with neutral input values. In other words, XOR is more controllable and observable than other types of gates. Therefore, these BISA circuits with much more XORs can have a higher test coverage. Three examples (DMA, DSP and RISC) are shown in

in the bottom right figure. Their test coverage is around 98% after applying 50,000 test patterns--lower than other benchmarks. If we further apply a number of random patterns, their test coverage still keeps increasing, but very slowly. We find that the remaining faults can be tested by some specific patterns with a large portion of deterministic bits that have a very small probability of occurring in random patterns. However, in order to get the full test coverage in a short time, certain seeds for LFSR can be shifted in so that these specific patterns can be generated as well.

3.4.4 Dynamic BISA vs. Static BISA

For the static BISA design, the original design must contain LSFR and MISR. The LFSR and MISR have been designed before layout design, so it is difficult to accurately predict the size of the BISA circuit at that time. If the pre-designed LFSR and MISR are not large enough to effectively test the entire BISA circuit when it is implemented in layout, the test coverage will not be very high, especially for a low-utilization design or SOC design. The dynamic BISA design approach described in Section 3.2.5 makes up the shortcoming of the static BISA design, which is more suitable for designs with large unused spaces. All designs have been revaluated to demonstrate if the dynamic BISA design can enhance the testability of BISA circuitry. Figure 16 shows the results of a portion of benchmarks using the static and dynamic BISA design approaches. In comparison, the dynamic BISA design algorithm can avoid redundant gate in BISA circuit, so their test coverage is always 100%. The test coverage in fault simulation for the static BISA design is around 95%, and the AES_core is even less than 80\%. We can make an obvious conclusion that the dynamic BISA insertion can enhance the testability of large BISA circuits significantly.

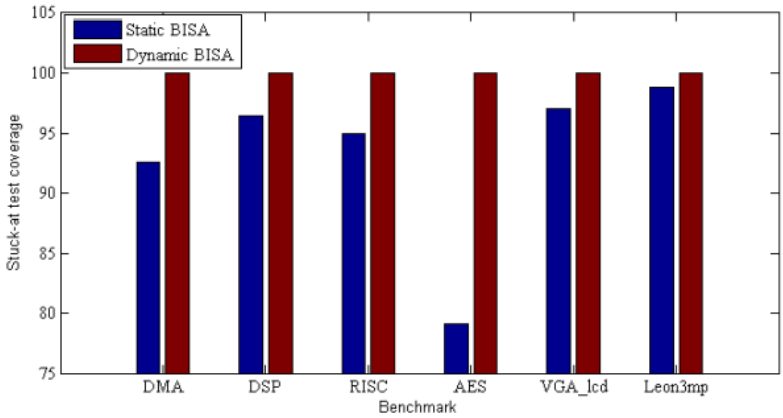


Figure 16: Test coverage using static and dynamic BISA designs.

3.4.5 Compaction Ratio

Although the dynamic BISA design can realize a higher test coverage than the static design, the dynamic method requires more large unused spaces to place sufficient scan flip-flops. Therefore, the dynamic BISA design is not suitable for very compact designs. In a layout, the number and size of unused spaces are dependent on the layout compaction ratio (core utilization). If the core utilization is very high, standard cells are placed tightly in each row. This will result in fewer and smaller unused spaces. There will not be enough scan flip-flops to construct large entough LFSR and MISR, so the LFSR and MISR cannot fully test the BISA circuits. Table 6 shows the results using various core utilizations in the two largest OpenCore benchmarks.

Table 6: Compaction Ratio.

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

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

3.4.6 Timing Overhead

Although the BISA circuits have no interconnections to the original circuits, there may still be a concern that the BISA cells will negatively impact timing of the original design. The additional metal interconnections introduced by BISA circuits will result in less routing space and more cross-talk. These effects could potentially make critical paths violate timing constraints that were verified without the BISA circuits. We have conducted experiments to explore how much extra delay will be introduced by the BISA circuits. Benchmark b18 has been implemented with and without BISA. 200 critical paths have been extracted from designs with BISA and without BISA by Synopsys Primetime, and the HSPICE has been employed to simulate their propagation delays. To obtain information on path delay changes with and without BISA insertion, the same paths are used for comparison. Since the BISA circuit size is dependent on the core utilization of a design, different core utilizations for one design are investigated as well. Table 7 shows results from the benchmark b18. The third column presents the average increased delay after BISA insertion across 200 critical paths. The fourth and fifth columns indicates the average percentage of the increase and largest increase respectively. As these results show, more unused spaces require more BISA cells as the core utilization goes down, thus increasing the impact on path delay. We can see that the impact is much less than the wafer-to-wafer process variation, because the largest impact is less than 2%. Therefore, the impact on timing is not a critical issue for the proposed BISA technique.

Table 7: Timing analysis on 200 critical paths in designs with and without BISA circuits.

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

4 Efficient and Secure Split Manufacturing via Obfuscated Built-In Self-Authentication

Recently, Split manufacturing, or split fabrication (used interchangeably in this section), has been proposed as an approach to enable use of state-of-the-art semiconductor foundries while minimizing the risks to IP [22]. Split manufacturing divides a design into Front End of Line (FEOL) and Back End of Line (BEOL) portions for fabrication by different foundries. An untrusted foundry performs (higher cost) FEOL manufacturing, then ships wafers to a trusted foundry for (lower cost) BEOL fabrication.

Two types of split manufacturing have been proposed in prior work: 2D integration and 3D integration based split fabrication. [23] first proposed the use of 3D integration of two tiers (the computation plane and the control plane) manufactured in separate foundries to ensure the performance of the computation plane and the security of the control plane. One tier is stacked on the top of another tier and conventional 3D stacked integration techniques are required to merge two tiers with vertical interconnections called through-silicon-vias (TSVs). Unfortunately, the semiconductor industry has not adopted 3D ICs as quickly as many in the industry expected. Given the barriers to 3D, 2D and 2.5D based split manufacturing are discussed more, such as those in [24] [25] [26] [27]. [22] developed a technique that makes FEOL and BEOL fabricated separately in different foundries and can make connections between them with wafer-bonding at a fine enough pitch similar to TSVs. [25] [26] demonstrated the feasibility of split fabrication after metal 1 (M1) on test chips and evaluated the chip performance. Although the split after M1 attempts to hide all inter-cell interconnections and can obfuscate the design effectively, it leads to high manufacturing costs. [26 [27] employed another integration approach. The back-end layers can be manufactured directly on top of the front-end layers with mask alignment techniques in a trusted foundry, which was studied on an FPGA chip fabricated in a split manufacturing process [27]. Finally, [28] presents a *k*-security metric to select wires to be lifted to a trusted tier (BEOL) to ensure the security when split at a higher layer. However, lifting a large number of wires in the original design will introduce large timing (≥73%) and power (≥54%) overhead and significantly impact chip performance [28], since delay and power are strong functions of wire length. In addition, though *k* similar elements can be created by lifting sufficient wires, it cannot prevent adversaries from tampering all these similar elements with untargeted Trojans.

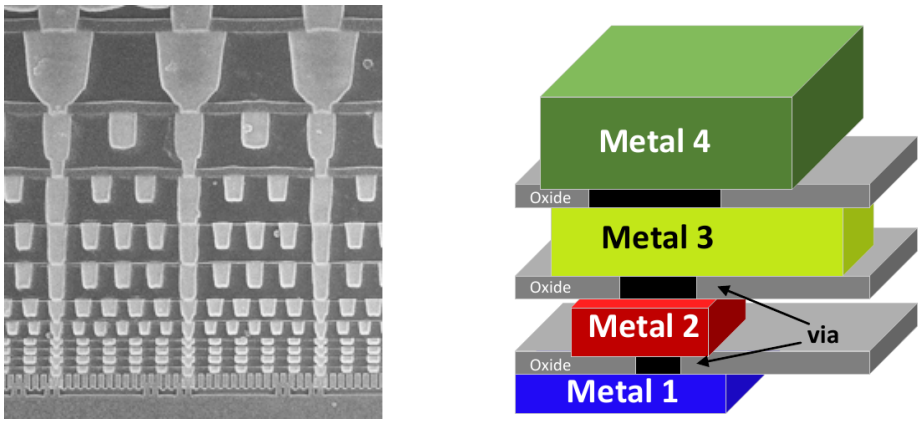
In this section, we propose a new design methodology that can effectively prevent reverse engineering of the chip functionality and further prevent hardware Trojan insertion with split fabrication process. Our technique allows FEOL and BEOL to be separated at higher layers (≥ M4) to reduce cost. In order to enhance effectiveness of obfuscation, all unused spaces in layout will be filled with additional functional cells or circuitry called obfuscated built-in self-authentication (OBISA) instead of non-functional filler cells during layout design. If any of the OBISA cells are replaced by a Trojan cell during FEOL, OBISA will be able to detect the modification. We propose to make connections between OBISA added into unused spaces and the original circuit, especially in its critical parts that are to be protected. The OBISA circuit not only makes it

extremely difficult for adversary to identify the original design, but also thwarts hardware Trojan insertion by filling unused spaces in layout. In addition, several design-for-security methods are proposed to minimize timing and power overhead introduced by OBISA circuit while maintaining a high test coverage for OBISA.

4.1 Low-Layer Split vs. High-Layer Split

4.1.1 Cost Issues

Reverse engineering requires analysis of local standard-cell types and their interconnections in order to identify structures or some targeted logics within a circuit. Split fabrication can prevent reverse-engineering the complete design or macro in a layout design by hiding a portion of wires. However, an adversary still could identify some sub-circuits, such as adder, decoder, cryptographic logic, etc., based on FEOL mask-layer information. The strength of obfuscation depends on the split layer, i.e., the layer that ends at the FEOL, since it determines how much layout information will be exposed to untrusted foundries. Figure 17 (a) shows a cross-section of a 14nm Intel chip [29] [30]. A high-layer split leaks more interconnections while low-layer split leaks much less. Split after M1 provides exceptional circuit obfuscation, because adversaries at untrusted foundry only see unconnected gates in layout and no inter-cell connections at all [24] [25]. However, such a low-layer split also brings challenges in split manufacturing process.



(a) Intel's 14nm Chip (b) Metal layers

Figure 17: A cross-section of an IC.

First, high metal layers are thicker and have a larger pitch than low metal layers, as shown in Figure 17 (a) and (b). Table 8 shows the pitches of different metal layers for a 45nm technology [30]. The integration process of FEOL and BEOL wafers requires precise alignment using either electrical, mechanical, or optical alignment techniques. A via requires a certain amount of space surrounding in order to satisfy design rules. If alignment is not perfect, the misalignment defects could influence circuit performance or even produce malfunction. However, the tolerance for misalignment improves greatly if the split occurs at higher layers. Unfortunately, misalignment has a higher probability to occur in split fabrication due to different process technologies or facilities at two different foundries (FEOL and BEOL), the yield for low-layer split could be relatively low.

Table 8: Pitch length of different metal layers in 45nm CMOS technology [30].

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

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

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

4.1.2 Security Issues: Obfuscation and Tampering

Split at a higher layer has many pros as described above, but it also results in significant interconnections for circuit blocks information leakage. Although adversaries cannot reverse engineer the entire design due to lack of connections in BEOL, it still provides adversaries opportunities to identify some sub-circuits (such as adder, decoder, and FSM) and tamper them with hardware Trojans. [24] and [26] proposed to insert additional ``dummy'' cells as spare cells for obfuscating the composition of a circuit, but the inserted cells have no interconnections between themselves or between them and the main design. Hence, they are easy to identify.

4.2 Split Manufacturing with OBISA

4.2.1 Proposed Approach

Based on the previous techniques, there is a tradeoff between cost and security for split manufacturing. However, our objective is to develop a methodology that allows high-layer splits, M3 or higher, while lowering the cost and at the same time providing a high level of security. We propose to insert *functional standard cells* into unused spaces of layout instead of filler cells or dummy cells. The inserted cells are connected together to form a circuitry, called obfuscated built-in self-authentication (OBISA). Note that this is different from the previously proposed built-in self-authentication (BISA) [18] for preventing Trojan insertion both from design and objective standpoints. As shown in Figure 18 and 19, OBISA is connected to the original circuit it is trying to protect, which makes it extremely difficult for adversaries at untrusted foundry to separate the OBISA design from the original design. However, OBISA circuit would result in dynamic power and timing overhead when the original circuit is operating. Therefore, a gating mechanism and a net selection method are proposed to minimize the negative impact on the original design. Additionally, we attempt to maintain a high test coverage for OBISA circuit with tree-structure circuits as in Section 3. Its built-in self-test (BIST) like structure can detect potential malicious modifications by test patterns that target stuck-at faults. However, the tree-structure circuit and test structure components could become a target for the adversary to identify them. In this work, an approach is presented that can create fan-outs to further obfuscate tree-structure circuit in OBISA without impacting its original high test coverage. Moreover, lifting critical wires to the trusted tier (BEOL) can prevent identifying the test structure within OBISA.

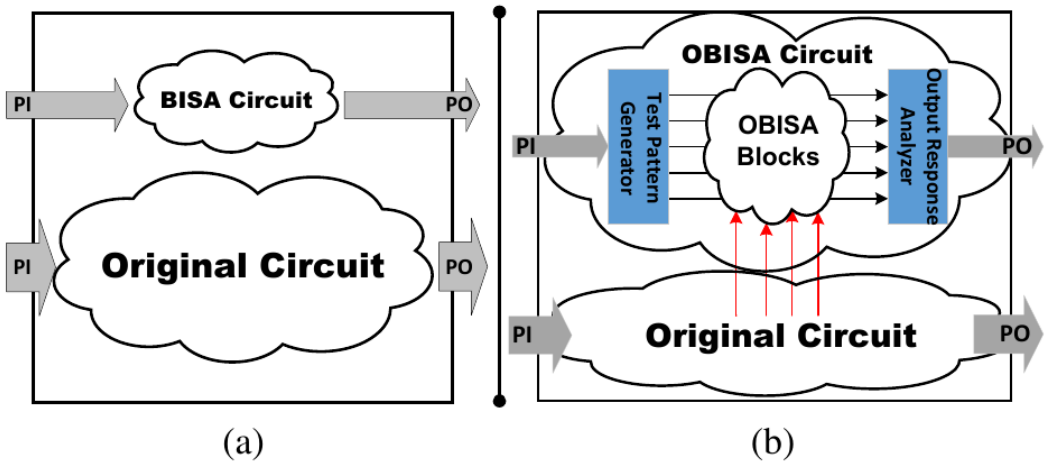


Figure 18: (a) BISA structure and (b) the proposed OBISA structure for split manufacturing in this work.

4.2.2 OBISA Structure

Figure 18(b) shows the structure of the proposed OBISA. It consists of a test pattern generator (TPG), an output response analyzer (ORA) and OBISA blocks under test. The entire OBISA circuit is implemented in unused spaces of the layout. In this work, we use a linear feedback shift register (LFSR) as TPG and multiple input shift register (MISR) as ORA. They are used to generate random test patterns and compress responses to generate a final signature, respectively [16].

Two operating modes are associated with the proposed technique. In *functional mode*, the original circuit is operating normally, but clocks for BISA circuits are disabled and LFSR and MISR are reset to their pre-defined state. A gating mechanism presented in Section 4.3.3 will block signals from activating original circuits to OBISA circuit. Therefore, OBISA circuit stays quiet and does not consume any dynamic power. In addition, these idle OBISA cells can fulfill the role of decap cells. Note that OBISA circuitry will, however, consume leakage power. The other mode, *authentication mode*, is used to authenticate a fabricated chip in the field. In this mode, LFSR generates a random test pattern at every clock cycle and the test pattern is shared by all OBISA blocks. At the same time, MISR collects responses from OBISA blocks and eventually produces a signature, as shown in Figure 18(b). The test phase ends when a sufficient number of test patterns have been applied.

Since inputs to OBISA come from LFSR and original circuits, thus test patterns for OBISA are generated depending on the LFSR and the state of the original circuit. We can keep the state of original circuit and run a large number of clock cycles on LFSR to test the inserted circuit as one iteration. In the next iteration, we change state of the original circuit and perform the tests once again. Test coverage will increase as more iterations are performed. Functional simulation at the OBISA design phase could help us find an efficient combination of the circuit state and the seed in the LFSR.

Note that clock is provided externally either in normal mode or in authentication mode. Typically, authentication test clock is much slower than functional clock [16]. The authentication clock frequency is dependent on the number of gates in an OBISA block and the gate types. The timing constraint can be obtained using post-layout timing analysis tool to find the longest path after OBISA inserted circuit. In the authentication mode, a very slow test clock can be used to ensure no path fails in OBISA circuits.

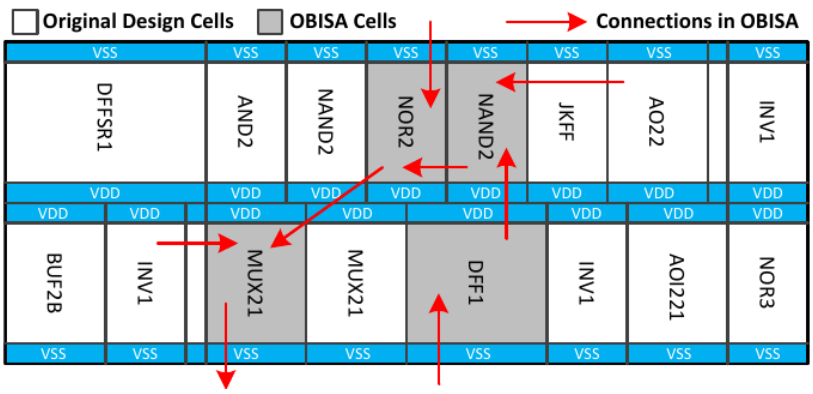
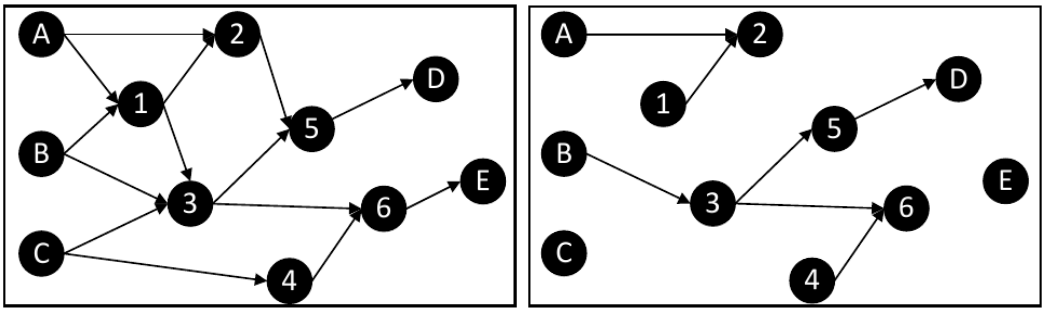


Figure 19: Layout after OBISA insertion.

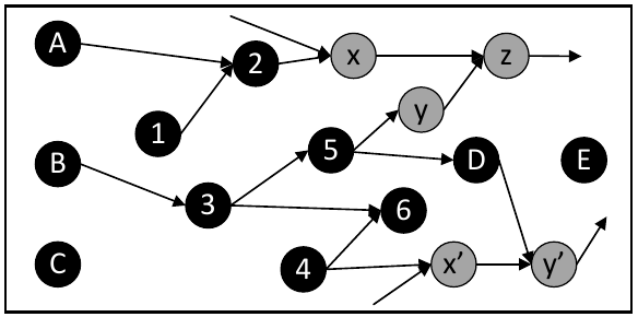
4.2.3 Security Analysis

Split manufacturing obfuscates the design by preventing an untrusted foundry from gaining a full view of the layout. Our proposed OBISA structure can improve obfuscation through the following features:

- Functional filler cells are inserted into unused spaces during layout design, as shown in Figure 19. Original cells and additional OBISA cells are all standard cells from the same technology library.
- Additional cells are connected together to construct combinational circuits. There is no difference in routing between original circuits and inserted OBISA circuits. While tree-structure circuit (no fan-out) is required for a BISA block in order to ensure a high test coverage, fan-out is allowed in this new OBISA technique. An approach for fan-out creation in OBISA blocks is proposed in Section 4.3.3.
- Interconnections between OBISA circuits and original circuits (obfuscation connections) are allowed. The additional logics can further obfuscate the original design, especially for the security-critical parts. For example, Figure 20(a) shows a design. Split fabrication can hide some wires in FEOL, but some sub-circuits (cell 3, 4, 5 and 6) could be identified. In Figure 20(c), the proposed technique can add OBISA cells to protect this sub-circuit from reverse-engineering.
- The inserted OBISA includes LFSR and MISR. We understand that LFSR and MISR have a unique structure and are controlled by mode select port, so they could be a target by an adversary to identify the additional OBISA circuit. Split manufacturing can hide critical wires in LFSR and MISR and effectively obfuscate LFSR and MISR.
- Additional local/global connections and various gates introduced by OBISA circuits can hinder neighbor connectedness analysis and standard-cell composition bias analysis [26]. OBISA blocks have not only local connections to connect cells nearby, but also long connections to other OBISA blocks, LFSR and MISR. A measure of spatial connectedness will be influenced by OBISA circuits. Similarly, additional cells with different types can change the types and proportions of cells of design presented in a small region in layout. OBISA cells can obfuscate the cell composition analysis.
- The proximity attack is based on the heuristic that floorplanning and placement (F&P) tools place the partitions close by and orient the partitions so as to reduce the wiring (delay) between the pins to be connected. [30] shows that the proximity attack could be a threat of connecting the missing nets correctly. However, OBISA circuit is able to produce additional tier-to-tier connections which can mitigate the threats of proximity attacks.



(a) Original Design (b) Design in FEOL



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

Figure 20: Circuit graphs: OBISA circuit is used to obfuscate the original design.

4.3 Implementation Strategy

4.3.1 Implementation Flow

Figure 21 presents our proposed OBISA implementation flow. The flow fits within the conventional ASIC design flow and is computable with current commercial physical design tools. OBISA insertion procedure begins after clock tree synthesis. At that point, the whole original circuit has been placed and no more cells will be added in conventional flow (the most left column in Figure 21). The unused spaces would be identified in DEF file and various standard cells are inserted depending on size of each unused space. More information regarding OBISA cell insertion will be described in Section 4.3.2. Once all unused spaces are filled with OBISA cells, we will begin to connect all OBISA cells in a region to construct a number of tree-structure combinational circuits (referred to as OBISA blocks). These steps as shown in the middle column were developed in Section 3. The steps in the third column are proposed to strengthen obfuscation, including fanout creation in tree-structure OBISA blocks, adding obfuscation connections, and lifting secure-critical paths within OBISA. Detailed connection strategies for each of them will be presented in Section 4.3.3. After the OBISA process, the flow resumes the procedure in conventional design flow. The physical design tool will perform routing for the entire design including original circuit and OBISA circuit. All constraints for the original design can be taken care of by the physical design tool during routing process. Once the timing and sign-off of the design are successful, the last step involves the generation of a GDSII format of the design for final tape-out.

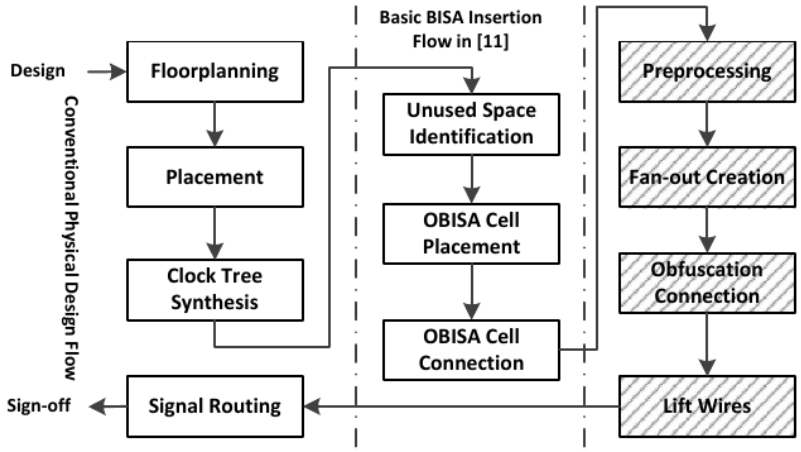


Figure 21: The OBISA implementation flow.

4.3.2 OBISA Cell Insertion Strategy

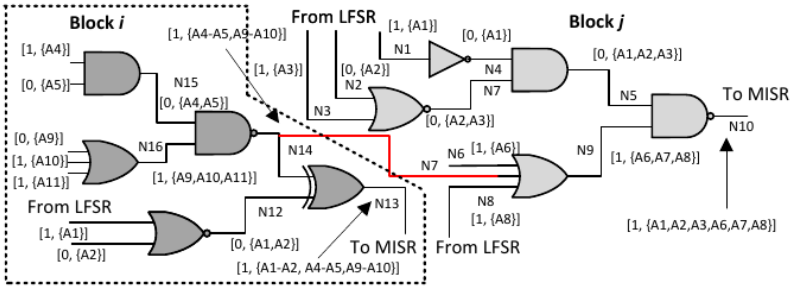
Several filling algorithms are proposed to fill every unused space as much as possible and no more cells could be added without changing the layout. Cells with different types and sizes are inserted to ensure a variety of OBISA cells due to the following reasons:

- A greater variety of OBISA cells can effectively thwart the cell composition analysis [26]. This is analogous to ``dummy'' cells in prior work.
- Our OBISA also supports either pre-mask or post-mask Engineering Change Order (ECO). Unused spaces are filled with a variety of standard cells and all these cells can be treated as spare logic gates. Moreover. When an OBISA cell is selected for ECO, a few modifications are needed to bypass this cell in OBISA block. Since OBISA circuits do not have a certain timing constraint, routing for OBISA could be very flexible.

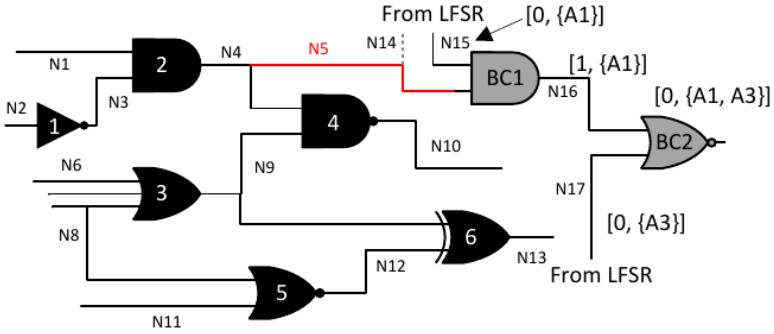
4.3.3 OBISA Cell Connection Strategy

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

1) **Preprocessing**: Two pieces of information are required to collect for each net in OBISA blocks: idle state (IS) and related inputs (RI) in LFSR. As we described earlier, OBISA circuits are working in authentication mode only, so LFSR will be set to a specific state if the chip is in the other mode. We propose to set the state of LFSR to a vector that has alternative `1' and `0' to avoid biasing values in OBISA circuit. If an input is connected to the original design because of the obfuscation connection, its IS is undetermined (`X'). Therefore, the IS for each net can be `1', `0', or `X', based on input values and their interconnections. IS can determine if a cell could be a gated cell. Another required information is the RI for each net. Taking the OBISA circuit in Figure 22(a) for example, the RI for the net N7 are A2 (N2) and A3 (N3), because nets N2 and N3 determine the value on net N7, while the related inputs for net N5 are A1, A2, and A3. Information regarding RI can help us decide how to create fan-out without losing any test coverage. In this work, we use a symbol, [x, {A,B,C}], to represent IS (x) and RI (A,B,C) of a net. One can see the IS and RI for the nets in OBISA circuits are labeled in Figure 22.



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



(b) An obfuscation connection is made.

Figure 22: Additional connections for improving obfuscation.

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

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

If these two conditions are satisfied, the net and the input pin can be a candidate pair for a fan-out. During the fan-out creation, the original connection on the input pin in block *j* from LFSR should be removed, so the pin is available for the new fan-out connection. These two tree-structure OBISA blocks *i* and *j* share a sub-tree circuit so that they are still tree-structure blocks in nature. Therefore, the added fan-out will not result in any extra redundant gates in OBISA blocks. Each net is still controllable, so a high test coverage of OBISA circuit will be achieved. Figure 22(a) shows an example of the fan-out creation. The net N14 in OBISA block *i* has completely different RI from the root output N10 in OBISA block *j*, so N14 can have a fan-out to connect any input in OBISA block j. In Figure 22(a), the pin for the net N7 is selected. Note that a fan-out cannot be made on the net N13, because the net N13 and N10 have shared RIs, A1 and A2.

3) **Gated Cells**: For logic gates, the dominant value on one input can result in a deterministic output regardless of logic values on other inputs, so those gates are gated by their dominant values. For example, '1' and '0' are used to gate OR and AND cells, respectively. We will take advantage of this for preventing dynamic power consumption within OBISA circuitry. An obfuscation connection must connect to a gated cell in OBISA circuit. For a gated cell that acts as an interface, at least one input's IS should be its dominant value during the normal operation. Other inputs of the gated cell can connect any net in the original circuit. In order to minimize modifications caused by the obfuscation connections, we prefer to choose a leaf cell for gated cell, because its input connections from LFSR can be removed without changing

existing OBISA blocks. In Figure 22(b), cells BC1 and BC2 are both gated in idle state. BC1 is selected since it is a leaf cell in the tree-structure OBISA block. All gated cells will be identified at this step, and they could be selected for obfuscation connections, as described in the next subsection.

4) **Obfuscation Connection**: Obfuscation connection also introduces additional interconnections between OBISA circuits and original circuits. Figure 22(b) shows an example of an obfuscation connection. It will cause inevitable increased capacitance on connected nets. Although we do not worry about the timing in OBISA circuits because of the slower frequency used during the authentication for OBISA, the added capacitance could potentially fail paths in the original design. Thus, we must select connection nets very carefully to avoid timing violations. Here, we propose an approach to select nets in the original circuit for obfuscation connection based on delay estimation by the static timing analysis (STA) tool.

- We define a parameter C_0, which is a threshold for dividing paths into critical paths and non-critical paths, as shown in Figure 23.
- Given the C_0, the STA is able to find all critical paths in original design. All nets on critical paths will be excluded from obfuscation connection. All remaining nets will be assigned a virtual path delay C_0. We call it virtual path delay, because it is not a real delay. Many non-critical paths have much smaller delay than C_0, but we treat them as the same length to simplify the problem.
- Another parameter C_1 is defined as a threshold to select net for obfuscation connection. The C_1 should be smaller than functional clock period C_2. The difference of C_1 and C_2 is a safe margin that ensures no timing violation is produced due to our rough estimation.
- If an obfuscation connection is made on a net, its virtual path delay will add an increased delay *D*. The increased value *D* will vary depending on the technology libraries. It can be estimated by averaging some samples in simulations. A large enough safe margin (C_2- C_1) can tolerate such a rough calculation.
- A net can be considered for an obfuscation connection if its virtual path delay plus *D* is still smaller than the threshold value C_1. For example, in Figure 23, path P0's added *D* is still smaller than C_1. In the example of Figure 22(b), the net N4 is selected for the obfuscation connection at this step.
- If there is an available gated cell nearby (BC1 in Figure 22(b)), an obfuscation connection can be made from the net (N4) in original design to one input (N14) of the gated cell.
- The increased capacitance not only influences one net, but also affects all nets on paths that pass this net. Therefore, the virtual delay of these relevant nets will be updated by adding *D*. For the circuit in Figure 22(b), the net N1, N2, N3, N5, and N10 need to be updated. For the net in Figure 23, there are two obfuscation connections that push its virtual delay from P0 to P2.

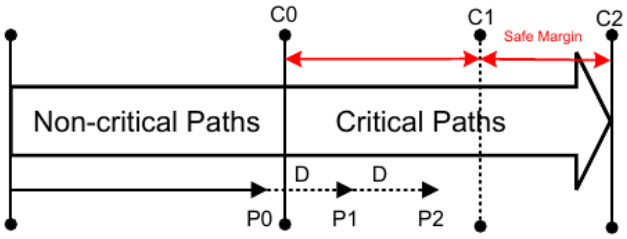


Figure 23: Net selection for obfuscation connection.

The value of increased delay *D* depends on the length of added wire. Experiment results in Section 2.1.1 shows that adding a short connection can bring about 30ps increased delay with 90nm technology. Since the obfuscation connection is made between nets not far away from

each other, the increased delay *D* will not be very large. We can obtain a conservative D value in simulations.

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

4.4 Experimental Results

4.4.1 OBISA Implementation

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

Table 9 shows the implementation results of five Opencore benchmark circuits with various scales. The number of OBISA cells for each circuit is listed in the third row of the table. Those OBISA cells do not introduce any area overhead because they are placed in the unused spaces in layout. Fan-outs and obfuscation connections are created between OBISA blocks to obfuscate their tree structures and increase the difficulty for adversaries to identify them in layout. An average 5% of nets go through trusted tier that is above M3 layer, as shown in the seventh row of the table. However, a small number of obfuscation connections are sufficient to make the inserted OBISA circuits look like a part of original design, to enhance obfuscation in FEOL. Since the test vector from LFSR has alternative `1' and `0', almost all leaf cells could be a gated cell. In our implementations, we use percentage to quantify how many inputs are altered for obfuscation connection in an OBISA circuit. Table 9 shows results with around 5% obfuscation connections (OCs). Thus, for an OBISA block with 80 inputs, there are 4 inputs connected to original circuit as obfuscation connections. Since all the OCs are made on short paths in the original circuit, no timing violations are introduced by OCs.

Table 9: Implementation results on different benchmark circuits.

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

4.4.2 Authentication Test Coverage Analysis

The authentication test coverage is a metric to assess the security level of the circuit. A higher stuck-at test coverage for OBISA circuits indicates that more OBISA cells could be verified by structural test patterns. The target coverage is 100%. According to our proposed flow in Section 4.3.1, all inserted OBISA cells will be connected in a tree-structure manner first. Then fan-outs and obfuscation connections are added based on the existed OBISA blocks. We take the OBISA circuitry in the Ethernet benchmark circuit as a running example to compare the test

coverage across different test patterns in five scenarios: *only tree-structure OBISA blocks*, *OBISA blocks with fan-out*, *BISA blocks with fan-out and three different proportions (5%, 15%, and 25%) of obfuscation connections*. For each scenario, four kinds of test patterns, 50,000, 100,000, 10,000 random patterns with different iterations, and ATPG (automatic test pattern generation) patterns, are applied to the OBISA circuits separately, and the results are shown in Figure 24. From the two left columns, we can see tree-structure OBISA circuit with and without fan-outs have the same test coverage using either random patterns or ATPG patterns. It demonstrates that the proposed fan-out creation method will not affect test coverage. The ATPG patterns can achieve almost 100% test coverage. For the random patterns from LFSR, the test coverage goes up as more patterns are applied. Their test coverage for 50,000 and 100,000 random patterns are 99.81% and 99.97%, respectively. The remaining three columns illustrate that test coverage will be impacted by obfuscation connections. This is because signals on obfuscation connections are tied to a constant value in one iteration and thus result in the controllability loss of gated cells. The last column shows an extreme scenario that 25% of inputs in OBISA blocks are connected to original design. The test coverage within 1-iteration is not high enough for the authentication test. However, as we described in Section 4.2.2, if we change the status of original circuits with multiple iterations, i.e. the values on obfuscation connections could change, the controllability of gated cells can be compensated to some degree depending on how many iterations can perform. Results in Figure 24 show that multiple iteration offers much better test coverage than the 1-iteration with the same number of test patterns, while using 10-iteration test leads to a higher test coverage than that with 5-iteration tests. The test coverage can improve further if more random patterns are applied or more iterations are performed.

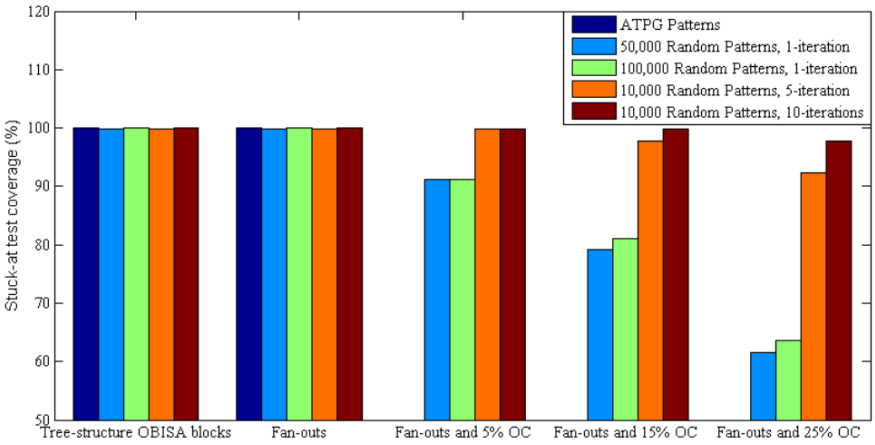


Figure 24: Authentication test coverage for an OBISA circuit.

5. Conclusions and Research Plans

In this report, we present a clock sweeping technique, a novel BISA technique, an OBISA technique to address IC trust issues by untrusted foundry. Clock sweeping is used to obtain the critical and non-critical path delay and then generate signatures for ICs for the purpose of detecting hardware Trojans. Statistical analysis methods have proved to be effective at identifying Trojan-inserted ICs in the presence of process variations, as demonstrated by both our simulation and FPGA implementation results. BISA will fill all unused spaces to prevent or hamper Trojan insertion process after completing layout design by leaving no space for Trojan

gates. BISA cells are connected to form a certain functionality. BISA has no impact on original design, since BISA and original circuits work independently. Additionally, different kinds of attacks can be detected, in order to ensure BISA's result is trustworthy. By comparing signatures, designers would know whether the chip has been tampered or not, as demonstrated by the implementation of different attacks. OBISA can effectively prevent reverse engineering of the chip functionality and further prevent hardware Trojan insertion with split fabrication process. Our technique allows FEOL and BEOL to be separated at higher layers (≥ M3) to reduce cost. We propose to make connections between OBISA added into unused spaces and the original circuit, especially in its critical parts that are to be protected. The OBISA circuit not only makes it extremely difficult for adversary to identify the original design, but also thwarts hardware Trojan insertion by filling unused spaces in layout. In addition, several optimization approaches are proposed to minimize timing and power overhead introduced by OBISA circuit while maintaining a high test coverage for OBISA.

The **future directions** of our work will be:

1. Evaluate and analyze the security of the OBISA technique against reverse engineering.
2. Apply OBISA to 3D IC whose dies are fabricated by trusted and untrusted foundries.
3. Develop programmable BISA cells and insertion methodology to improve testability of BISA circuitry
4. Develop a novel and effective OBISA technique without split manufacturing

References

[1] M. Tehranipoor and C. Wang, ``Introduction to Hardware Security and Trust," Springer, August 2011.

[2] M. Tehranipoor and F. Koushanfar, ``A Survey of Hardware Trojan Taxonomy and Detection," IEEE Design and Test of Computers, pp. 10-25, January-February 2010.

[3] M. Banga and M. Hsiao, ``A novel sustained vector technique for the detection of hardware trojans," IEEE VLSI Design, pp. 327 –332, Jan. 2009.

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

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

[6] J.Aarestad, D. Acharyya, R. Rad, and J. Plusquellic, ``Detecting Trojans through leakage current analysis multiple supply pad IDDQs," IEEE Transactions on Information Forensics and Security, vol. 5, issue 4, pp. 893-904, 2010.

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

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

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

[10] R.Karri, J.Rajendran, K.Rosenfeld, and M.Tehranipoor, ``Trustworthy Hardware: Identifying and Classifying Hardware Trojans," IEEE Computer Magazine, vol.43, pp.39-46, 2010.

[11] F.Wolff, C.Papachristou, S.Bhunia, and R.Chakraborty, ``Towards Trojan-free Trusted ICs: Problem Analysis and Detection Scheme," Design, Automation and Test in Europe (DATE), pp.1362-1365, 2008.
[12] R.Rad, M.Tehranipoor, J.Plusquellic, ``A Sensitivity Analysis of Power Signal Methods for Detecting Hardware Trojans under Real Process and Environmental Conditions," IEEE Trans. in VLSI, vol18, no.12, pp.1735-1744, 2009.
[13] Y.Jin and Y.Makris, ``Hardware Trojan Detection using Path Delay Fingerprint," IEEE Int. Workshop on Hardware-Oriented Security and Trust(HOST), pp.51-57, 2008.
[14] H.Ren, Z.Wang, W.Shi, and Doug Edwards, "Critical Path Analysis in Data-Driven Asynchronous Pipelines" Int. Conf. on Computer Communication and Informatics (ICCCI), pp. 1-9, 2012.
[15] I.Borg and P.Groenen, "Modern Multidimensional Scaling, Theory and Applications," New York, Springer-Verlag, 1997.
[16] M. Bushnell and V. Agrawal, "Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits," Springer, 2000.
[17] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, "Introduction to Algorithms," The MIT Press (third edition), 2009.
[18] K. Xiao and M. Tehranipoor, "Built-In Self-Authentication for Preventing Hardware Trojan Insertion," IEEE Int. Symposium on Hardware Oriented Security and Trust (HOST), pp. 45-50, 2013.
[19] H. Xiang, K. Chao, and M. Wong, "An ECO Routing Algorithm for Eliminating Coupling-Capacitance Violations," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, issue 9, Spet. 2006.
[20]N. Weste and D. Harris, "CMOS VLSI Design: A Circuits and Systems Perspective," 4th edition, Addison-Wesley, 2010.
[21]M. Banga and M. Hsiao, "A Novel Sustained Vector Technique for the Detection of Hardware Trojans," International Conference on VLSI Design, pp. 327-332, Jan. 2009.
[22]R. Jarvis and M. McIntyre, "Split Manufacturing Method for Advanced Semiconductor Circuits," US Patent 10/305,670, 2007.
[23]J. Valamehr, et al, "A 3-D Split Manufacturing Approach to Trustworthy System Development," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 32, No. 4, April, 2013.
[24]K. Vaidyanathan, et al, "Building Trusted ICs using Split Fabrication," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2014.
[25]K. Vaidyanathan, et al, "Efficient and Secure Intellectual Property (IP) Design with Split Fabrication," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2014.
[26]M. Jagasivamani, et al, "Split-Fabrication Obfuscation: Metrics and Techniques," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2014.
[27]B. Hill, et al, "A Split-Foundry Asynchronous FPGA," Proceedings of the IEEE Custom Integrated Circuits Conference (CICC), September, 2013.
[28]F. Imeson, et al, "Securing Computer Hardware Using 3D integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation," in the Proceedings of the 22nd USENIX Security Symposium, Aug., 2013.
[29]Chipworks Inc. http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/intels-14-nm-parts-are-finally-here/
[30]J. Rahendran, et al. "Is Split Manufacturing Secure?" IEEE Design, Automation & Test in Europe, 2013.

[31]J. Roy, F. Koushanfar, and I Markov, “EPIC: Ending Piracy of Integrated Circuits,” Design, Automation and Test in Europe (DATE), 2008.
[32]R. Cocchi, P Baukus, L. Chow, and B. Wang, “Circuit Camouflage Integration for Hardware IP Protection,” 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), 2014.