Integrity protection and authentication of integrated circuit intellectual property cores

Zhang, Li

2015

Zhang, L. (2015). Integrity protection and authentication of integrated circuit intellectual property cores. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/62512>

<https://doi.org/10.32657/10356/62512>

Integrity Protection and Authentication of Integrated Circuit Intellectual Property Cores

Zhang Li

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2015

Acknowledgements

First of all, I would like to express my deep appreciation to my supervisor, Associate Professor Chang Chip Hong. Without his intellectual guidance, encouragement and support, I could never finish this thesis in four years of my PhD pursuit. His research attitude of striving for the best and being precise, enthusiasm for work and responsible and kind personality have not only shown me how to be a good researcher, but also been an excellent model for me to follow in my future career and life.

I am also very grateful to my parents, my younger brother and my girlfriend. Their love, consideration and company were and will continue to be what drives me to move forward and become my better self.

My gratitude also goes to my friends in Singapore, Pengfei Zhang, Hanhe Geng, Gao Wang, Chu Qin, Xu Xiao, Ansen Zhang, Lanlan Dong etc. Together, we exercise, we travel, and we enjoy our spare time. Only with you can I experience the wonderfulness of this world so much. You Zou, you are not forgotten. The distance between us cannot change the fact that you are one of my best friends.

Last but not least, I would like to thank my colleagues in the Center for High Performance Embedded Systems and acknowledge the School of Electrical and Electronic Engineering of Nanyang Technological University for providing me such a friendly and helpful research environment. Special thanks go to the Chinese library, in which I spent most of my time after lunch to rest my mind and roam in the sea of Chinese history and philosophy.

Abstract

With the advancement of semiconductor processing technology, the capacity and versatility of an integrated circuit (IC) have been growing perpetually. Highly sophisticated systems can now be resided in a single IC chip. Accompanied with the increased integration density is a widening of design productivity gap, which can never be closed by using existing design automation tools with conventional design methodology. Today, reuse-based or intellectual property (IP) based design methodology has prevailed as the most effective means to increase the design productivity. The downside of it is the IP cores are now targets of abuse and have become more vulnerable than ever to various infringement attacks. Protection of IP cores by legal means is passive and feeble, and must be fortified by technology and forensic approaches. This thesis presents several new techniques for the integrity protection and authentication of the IP cores. The three facets to this research towards trusted hardware IPs are the identification of IP owner and IP buyer, the usage control of an IP core, and the integrity verification of the IP content. One major hardware security problem in each of these three facets is independently investigated. The techniques developed can be combined to form an augmented solution to deal with multiple security issues arising from the design of an IP to its integration into chips or any other vulnerable channels in the transition of the chips containing the IP cores.

A dynamic fingerprinting scheme for the protection of sequential circuit IPs is first proposed. The inserted fingerprint is an oblivious ownership watermark independently endorsed by each user through a blind signature protocol. The fingerprint can be conveniently detected from the output response off-chip by injecting a specific input sequence. From the fingerprint, the IP provider can easily prove his ownership of the IP core and identify the buyer. The proof is indisputable as the buyer has endorsed the concealed watermark of the IP owner in the process of fingerprint generation. The security analyses and experimental results show that the fingerprinting scheme is able to generate a large number of high quality fingerprinted instances that are robust against all perceivable attacks.

A pragmatic per-device licensing scheme is also proposed for the IP cores used in field-programmable gate array (FPGA) chips. The scheme enables the licensing of IP cores on a per-device basis by exploiting only existing hardware primitives on commercial FPGA chips and without the need for an external trusted third party. Besides guaranteeing the secrecy and integrity of the licensed IP cores, the scheme also prohibits the implementation of the protected IP cores on unscreened excess devices and counterfeit chips sold in the gray market. The scheme blends well with the IP fingerprinting scheme, where the latter will deter the IP licensees from abusing their IP instances. Using the self-reconfiguration feature of FPGAs today, the resources consumed by the control module used for the implementation of the protected IP cores on the authorized chip are temporary and marginal.

Finally, a technique for verifying the integrity of fabricated chips is presented. It is capable of detecting small hardware Trojan (HT) in the circuit even if it remains dormant through a fast gate-level characterization (GLC) process of the leakage current. The GLC process is efficiently performed by the normal equation of linear regression model. Based on the discrepancies in the bias parameter of the linear regression and the accurately estimated scaling factors of a subset of gates, the HT-infected chip can be distinguished without the need to compare with the parameters from a Trojan-free golden chip, which can only be obtained through expensive destructive reverse engineering.

Table of Contents

Acknowledgements.....	iii
Abstract.....	v
List of Tables	v
List of Figures	viii
List of Abbreviations	x
Chapter 1 Introduction	1
1.1 Motivation.....	1
1.1.1 Semiconductor IP Market and IP Categories.....	2
1.1.2 IP Infringement and Protection.....	4
1.2 Research Objective	6
1.3 Major Contributions.....	6
1.4 Organization of the Thesis.....	8
Chapter 2 Preliminaries and Literature Review.....	10
2.1 Integrated Circuit IP Protection	10
2.2 Hardware IP Watermarking and Fingerprinting	11
2.2.1 Hardware IP Watermarking.....	12
2.2.1.1 Watermark Embedding	14
2.2.1.2 Watermark Extraction	16
2.2.2 Hardware IP Fingerprinting	18
2.3 Active Control of the IP Usage.....	24
2.3.1 The FPGA IP Market.....	25
2.3.1.1 Principals in the Market	25
2.3.1.2 Common Attacks to FPGA based IPs	27
2.3.1.3 Desiderata of FPGA IP Licensing Scheme	28
2.3.2 Commonly Employed FPGA Features in IP Licensing Schemes	29
2.3.3 Conventional Crypto Primitives based Licensing Schemes	30

2.3.3.1	A Licensing Scheme for IP cores at a high abstraction level.....	31
2.3.3.2	A Licensing Scheme with Asymmetric Crypto based Key Derivation.....	33
2.3.3.3	A Licensing Scheme with TTP and Symmetric Crypto based Key Derivation.....	34
2.4	Hardware Trojan Detection.....	37
2.4.1	Hardware Trojan Taxonomy.....	38
2.4.2	Hardware Trojan Detection Methods.....	41
2.4.2.1	Logic testing based approaches for HT detection in fabricated chips	42
2.4.2.2	Side-channel based approaches for HT detection in fabricated chips	43
2.5	Chapter Summary	49
Chapter 3	A Blind Dynamic Fingerprinting Technique for Sequential Circuit Intellectual Property Protection	51
3.1	Introduction.....	51
3.2	Preliminaries on Test Machine Insertion	52
3.2.1	Test Machine Fundamentals	52
3.2.2	Test machine Insertion.....	54
3.3	The Proposed Fingerprinting Scheme.....	59
3.3.1	Fingerprint Generation.....	59
3.3.2	Fingerprint Insertion	60
3.3.3	Partitioning for Efficient Fingerprinting.....	63
3.3.4	Fingerprint Detection and Authorship Verification.....	69
3.3.5	Augmented Fingerprint.....	69
3.4	Security Analysis of the Proposed Scheme	70
3.4.1	Fingerprint Credibility	70
3.4.2	Fingerprint Robustness	70
3.4.2.1	Collusion attack.....	70
3.4.2.2	Combinational logic resynthesis	71
3.4.2.3	Circuit retiming	72
3.4.2.4	State re-encoding.....	73
3.4.2.5	State reduction.....	74

3.4.3 Fingerprint Feasibility	74
3.5 Experiment Results	75
3.6 Chapter Summary	84
Chapter 4 Pragmatic and Secure Per-Device Licensing Schemes for FPGA IP Cores	86
4.1 Introduction.....	86
4.2 Prerequisite and Assumptions.....	87
4.3 The Proposed Scheme.....	88
4.3.1 Core Installation Module (CIM)	88
4.3.2 Protocol Execution.....	89
4.3.2.1 Device Enrollment.....	89
4.3.2.2 IP Core Enrollment.....	89
4.3.2.3 IP Core Licensing.....	90
4.3.3 Implementation of Multi-IP Core System.....	92
4.4 Evaluation and Discussion.....	95
4.4.1 Security Analyses.....	95
4.4.1.1 Evasion of license fee by SD.....	96
4.4.1.2 Attacks by MA	96
4.4.1.3 Breaching of trust by FV	97
4.4.2 Possible Security Enhancements and Variants	97
4.4.2.1 Adding an ancillary establishment module	98
4.4.2.2 Use asymmetric crypto for IP key establishment.....	99
4.4.3 Practicality of the Scheme	100
4.4.4 Implementation Considerations	101
4.5 Chapter Summary	105
Chapter 5 Hardware Trojan Detection with Linear Regression based Gate-level Characterization	106
5.1 Introduction.....	106
5.2 Preliminaries on Problem Formulation.....	107
5.2.1 Process Variation	107
5.2.2 Gate-level Characterization of Leakage Current	108
5.2.3 Linear Regression with Multiple Variables	109
5.3 The Proposed HT Detection Method	110

5.4	Experimental Evaluations	115
5.5	Chapter Summary	120
Chapter 6	Conclusions and Future Work.....	121
6.1	Conclusions.....	121
6.2	Recommended Future Work.....	123
6.2.1	Fragile IP tagging for HT detection.....	123
6.2.2	FPGA IP licensing schemes based on public-PUF	124
6.2.3	HT detection with segmentation techniques	125
AUTHOR'S PUBLICATION		126
BIBLIOGRAPHY		128

List of Tables

Table 2.1	Features and drawbacks of existing core-level IP protection schemes.	37
Table 3.1	Synchronizing sequences and distinguishing responses of 3-FF test machine.	54
Table 3.2	Synthesis results for S27 after test machine insertion.	58
Table 3.3	Synthesis results (area in μm^2) for test machine embedded components of S344.....	67
Table 3.4	Fingerprinting overheads with state re-encoding for minimum area. $\#FF$ is the number of FFs. A_s and A_m , D_s and D_m , P_s and P_m are areas in μm^2 , delays in ns , and powers in mW of the non-fingerprinting optimized seed design and fingerprinted design, respectively. ΔA , ΔD , and ΔP are their respective percentage increments.	76
Table 3.5	Fingerprinting overheads with state re-encoding for minimum area-delay product. The definitions and units of all notations are the same as those in Table 3.4.....	77
Table 3.6	Area and timing overheads of fingerprinted designs implemented on FPGA. A_s and A_m , D_s and D_m are the areas in number of slices and delays in ns of the seed design and fingerprinted design, respectively	78
Table 3.7	Synthesis results for 100 different fingerprinted instances of each circuit. A_m and D_m denote area in μm^2 and delay in ns , respectively. $\%A_{err}$ and $\%D_{err}$ are obtained by calculating the margins of error A_{err} and D_{err} for the area and delay for 95% confidence interval and normalizing them by the average area and delay, respectively.	80

Table 3.8	Comparison of synthesis results with FSM watermarking [43]. A_s and A_m , and D_s and D_m are the areas in μm^2 and delays in ns of the seed design and fingerprinted design, respectively.....	81
Table 3.9	Comparison of area and delay overhead with [39]. A negative overhead means an improvement in the corresponding design attribute.	82
Table 3.10	Comparison of mark robustness with [43].....	84
Table 4.1	Overheads of Boolean and multiplicative masking [135], masked S-box [136], masked S-box optimized by larger LUT in recent FPGA [137] and dual-rail precharge logic [138] methods of masked AES implementation.	103
Table 4.2	Resource consumptions (in # equivalent slices) on Xilinx Vertex-6 devices of the unprotected, side-channel attack (SCA) resistant, fault attack (FA) resistant, and final physical attack (PA) resistant versions of CIM.....	104
Table 5.1	(a) The scaling factor for each gate in the simulated circuit and (b) The modulated mean and variance of calculated θ_j for the HT-free and HT-infected cases	114
Table 5.2	Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C17 associated with different PV. The HT used is a 2-input NAND gate.	116
Table 5.3	Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C432 associated with different PV. The HT used is a 2-input NAND gate.	117

Table 5.4	Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C499 associated with different PV. The HT used is a 2-input NAND gate.	117
Table 5.5	Comparison of the averaged discrepancy between HT free and infected circuits in [101] and the proposed method.....	116
Table 5.6	Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C880a associated with different PV. The HT used is two 2-input NAND gates.....	118
Table 5.7	Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C17 associated with different PV for correlated HT case. The HT gate is inserted to receive the same inputs as gate 5 of the circuit.	119

List of Figures

Figure 1.1 Three main types of IP cores for the hardware design part in the SoC flow	4
Figure 2.1 Illegal usage of an IP core and tracking the source of misuse by fingerprint.....	11
Figure 2.2 A generic model for IP watermarking	12
Figure 2.3 A generic pre-processing constraint-based IP watermarking procedure	14
Figure 2.4 Four instances of the same function in a tile with fixed interface [52]	20
Figure 2.5 Pseudo-code of the iterative fingerprinting approach on standard-cell placement.	21
Figure 2.6 Duplicating vertex A to generate a solution from which two solutions can be derived for the original graph.	23
Figure 2.7 Interactions and interests of the four main parties in the FPGA IP market	26
Figure 2.8 FPGA IP licensing for high-level IP cores [78].	32
Figure 2.9 FPGA IP licensing with asymmetric crypto based key derivation [79]	34
Figure 2.10 FPGA IP licensing with symmetric crypto and TTP [80].	35
Figure 2.11 The symmetric cryptography based core installation module.	36
Figure 2.12 A simple Trojan with the triggering and payload logic.....	38
Figure 2.13 HT taxonomy proposed in [84].	39
Figure 2.14 Path delay characterization with a negative-skewed shadow register [94].....	44

Figure 2.15 Flow of thermal conditioning for breaking gate correlations in GLC [102].	47
Figure 2.16 Circuit segmentation technique to enable scalable GLC [103].	47
Figure 2.17 The iterative HT detection algorithm in [104].	48
Figure 3.1 STG of a 3-FF test machine.	54
Figure 3.2 Minimum dependency graph of 3-FF test machine.	56
Figure 3.3 Gate-level schematic of S27.	57
Figure 3.4 State variable dependency graph of S27.	57
Figure 3.5 Schematic of S27 after test machine insertion.	58
Figure 3.6 Fingerprint insertion algorithm.	62
Figure 3.7 Dependency-directed partitioning algorithm.	65
Figure 3.8 Dependency graph of S344 excluding self-dependencies.	66
Figure 3.9 STGs of (a) 1-FF and (b) 2-FF test machines.	72
Figure 4.1 Block diagram of the core installation module (CIM).	89
Figure 4.2 The proposed pay-per-use IP licensing protocol.	90
Figure 4.3 Queue of design modules in external NVM of device k in batch i .	93
Figure 4.4 Configuration steps for protected IP_1 on device k in batch i .	94
Figure 4.5 Communication of IP core pertinent data among FV, CV and SD.	95
Figure 4.6 Structure of the CIM using asymmetric-crypto for key establishment	100
Figure 5.1 The generic forms of (a) the SLE with M equations and N variables, (b) the vector y with M elements, and (c) the M -by- $(N+1)$ Matrix X .	110
Figure 5.2 Flow chart of the proposed 2-phase HT detection method.	113
Figure 5.3 The example circuit of 7 NAND gates with a 1-NAND HT.	114

List of Abbreviations

AES	Advanced Encryption Standard
ATPG	Automatic Test Pattern Generation
CIM	Core Installation Module
CLB	Configurable Logic Block
CRC	Cyclic Redundancy Check
CRP	Challenge-Response Pair
CUT	Circuit Under Test
CV	Core Vendor
DC	Design Compiler
DfT	Design for Testability
DG	Dependency Graph
DS	Distinguishing Sequence
DR	Distinguishing Response
ECC	Error Correction Code
ECDH	Elliptic Curve Diffie-Hellman
EM	Establishment Module
FA	Fault Attack
FF	Flip-Flop
FIB	Focused Ion Beam
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
FV	FPGA Vendor
GLC	Gate-Level Characterization
HDL	Hardware Description Language

HM	Hardware Manufacturer
HT	Hardware Trojan
IC	Integrated Circuit
ICAP	Internal Configuration Access Port
IP	Intellectual Property
IUA	IC Under Authentication
KDF	Key Derivation Function
LSB	Least Significant Bit
MA	Malicious Attacker
NVM	Non-Volatile Memory
PA	Physical Attack
PNG	Pseudorandom Number Generator
PUF	Physical Unclonable Function
PV	Process Variation
SAT	Satisfiability problem
SCA	Side Channel Analysis
SCC	Strongly Connected Component
SD	System Developer
SEM	Scanning Electron Microscope
SfT	Synthesis for Testability
SHA	Secure Hash Algorithm
SLE	System of Linear Equations
SoC	System on a Chip
SS	Synchronizing Sequence
STG	State Transition Graphs
SVD	Singular Value Decomposition
TCR	Trojan to Circuit Ratio

TE	Test-Enable
TLS	Transport Layer Security
TTP	Trusted Third Party
UART	Universal Asynchronous Receiver/Transceiver

Chapter 1

Introduction

1.1 MOTIVATION

Electronic equipment has been pervasive in our daily life and used across almost every industry. In virtually all the electronic equipment, the integrated circuit (IC) is a fundamental building block. An IC is a group of electronic circuits that are integrated on a small plate (i.e., chip) of semiconductor material (typically silicon). Since its invention in 1950s, ICs have experienced rapid increase in complexity and followed closely the Moore's law, which states that the number of transistors in an IC doubles every 18 months. In 1971, the number of transistors in a microprocessor IC was around 2300. Till 2013, there had been 5 billion transistors in a microprocessor IC. The current world record for the number of transistors per chip is held by a Xilinx field-programmable gate array (FPGA) chip, which contains more than 20 billion transistors [1]. The increase in complexity of the IC has made it possible for an entire sophisticated system to be implemented on chip, i.e., the system on a chip (SoC). Indeed, the advancement in the fabrication technology that enables the exorbitant growth in new functionalities for an IC has been the important driving force behind the continuous proliferation of versatile electronic devices such as personal computers, smart phones and tablets.

However, the design productivity does not increase at the same rate as the rapid increase in the integration density of the IC chip, which creates a continuously widening gap between what can be built in a chip and what can actually be designed. In other words, there are more transistors available in a chip than the designer could use them meaningfully. With the declining product life cycle and time to market, it is no longer viable to design an advanced system from scratch. To shorten the design turnaround time, reuse-based design methodology has been advocated and is prevailing. In this contemporary design paradigm, reusable design blocks, also known

as intellectual property (IP) cores, are purchased from external entities and incorporated into the system to be designed. Examples for the pre-designed IP cores are the bus controller, microprocessor, memory subsystem, and specialized peripheral functions like the universal asynchronous receiver/transceiver (UART), etc. Besides shortening the system development time, reuse-based design methodology also allows the system designer to focus on the critical blocks of the system to differentiate them from the competitor's systems.

1.1.1 Semiconductor IP Market and IP Categories

The popularity of the reuse based design methodology has boosted the transactions of IP cores among different entities, resulting in a flourishing semiconductor IP market. The new entity that is brought into the ecosystem of IC development is the design houses (namely IP vendors), who are also the key drivers and beneficiaries of this growing IP market. They develop novel and highly optimized IP cores and gain revenues from licensing the use of the IP cores to system designers. According to a market research report by Markets and Markets [2], the semiconductor IP market is expected to grow at a compound annual growth rate of 12.6% from 2014 to 2020 and reach \$5.63 billion in 2020.

There have been an extensive number of different IP cores available in the semiconductor IP market, which can be used in different abstraction levels of the SoC design flow. A typical SoC design flow is shown in Fig. 1.1, which consists of both software and hardware development (software/hardware co-design). The IP cores produced in the hardware development flow can be classified into three major types, i.e., the soft, firm and hard IP cores [3]. The main features of these IP cores are summarized as follows:

The soft IP core is used in the behavioral level. These IP cores are usually delivered in the form of synthesizable hardware description language (HDL). This type of IPs provides excellent flexibility to match the requirements of a specific system. The drawback of these IP cores is that their performance is highly dependent on the optimization effort of the system integrator, which is less predictable than the other two types of IP cores.

The hard IP core is used in the physical level. These IP cores target a specific technology and are delivered in the form of fully optimized netlist or the corresponding physical layout. This type of IP cores offers the best performance for the chosen technology library; but due to process dependencies, they have much less flexibility and portability than the soft IP cores. Without requiring any further optimization, the hard IP core is generally released as a drop-in replacement in the physical level design of the system.

The firm IP, or semi-hard IP, refers to IP cores that are in an intermediate form between the soft IP and hard IP. They are usually delivered in the form of gate-level netlist in the structural level. This type of IP cores has more predictable performance than the soft IP and greater flexibility and portability than the hard IP. They may also be optimized using a generic technology library, including even physical synthesis steps like floor planning and placement. Nonetheless, as no routing is performed, the firm IP remains relatively technology independent. After integrating the firm IP into the system, the system integrator still needs to perform the physical synthesis for its optimization in a specific technology.

Figure 1.1 Three main types of IP cores for the hardware design part in the SoC flow.

1.1.2 IP Infringement and Protection

The creation of IP cores involves substantial time, effort and monetary investment. However, the reuse-based design environment makes the shared IP cores more vulnerable than ever to various infringement attacks. For example, a legitimate buyer (a system designer) of one IP core may misuse the IP core in an unauthorized application or illegally resell the IP core to other system designers. The attacker may also tamper with the distributed IP core to include a malicious component, namely hardware Trojan (HT). If the HT-infected IP core is integrated with other genuine IP cores in a system, functionality of the system may be changed or disabled and valuable information of the genuine IP cores or the system may be divulged. These two attacks can be termed as IP misuse and IP tampering, respectively.

The system designed by the system integrator can also be deemed as a monolithic IP. With the increase in the expense for building and maintaining the state-of-the-art manufacturing facility, many semiconductor companies like Qualcomm, Broadcom and AMD have outsourced the fabrication of their chips to external contract foundries [4]. There are now many opportunities in the fabrication steps for an attacker to implant the HT to the system, forming a variant of the IP tampering attack described above.

Prevailing means of IP protection includes external communications of legal protection, such as patents, copyrights and contracts, to deter illegal act, and licensing agreement and encryption to prevent unauthorized use of IPs. Such approaches, while effective against the benign users, are inadequate to suppress IP infringement by aggressors, malicious clients, competitors and curious contractors. Owing to the easily accessible, easily integratable and uncommitted physical manifestation of reusable IPs, the fraudsters can easily dispute or challenge the validity of the patent or copyright behind an IP, especially if it is not broad, fundamental or strong with regards to the obviousness test, novelty of idea or uniqueness of construction over the prior art. The lengthy and costly legal processes, together with the high degree of uncertainty over the outcome of IP litigation, tend to discourage the pursuit of IP infringements. As cases of IP violations by customers are more difficult to prove, IP vendors usually do not take legal action against their customers, resulting in the buoyant IP theft.

As long as IP infringement remains easy and perpetration is difficult to prove, sizeable portion of the investment in IP development will continue to be extorted. It is estimated that the annual revenue loss due to IP infringement in the IC industry is in excess of \$5 billion in 2006 and the loss continues to escalate [5]. The IP market is in dire need for a secure transaction environment, wherein the IP vendors get assurance that their IP cores cannot be abused and no design information will be divulged and the system developers get assurance that the received IP cores are authentic and uncompromised.

1.2 RESEARCH OBJECTIVE

The primary objective of this research is to develop new technical means that can be used to fortify the protection of IP cores used in an IC in order to foster a secure IP transaction environment. In particular, measures for protecting and verifying the integrity and authenticity of the IP cores will be investigated. The research consists of three facets, i.e., identification of IP ownership and IP buyer, active control of IP usage and integrity verification of IP content. IP watermarking and fingerprinting techniques will be studied in the first instance. These techniques help to identify the IP ownership (and the IP user for fingerprinting) but the protection is limited to passively confirming the occurrence of IP infringement. Secure transaction of IP cores require advanced protocol and ingenious exploitation of cryptographic primitives to meter the usage and control the integration of different IP cores onto a chip to prevent IP infringement or make it more difficult or costly for unauthorized IP cores to sneak into an integrated system. Secure verification of the integrity and authenticity of IP cores will be investigated to realize such an active IP tagging and monitoring mechanism. Finally, if all countermeasures fail, it is important that there is a means to efficiently screen the post-production chips to identify potentially dubious chips. In this respect, improved approaches to more effectively detect malicious hardware Trojan implanted in a fabricated chip will also be investigated.

1.3 MAJOR CONTRIBUTIONS

The following contributions have been made in this research towards achieving the abovementioned objectives.

- 1) A new fingerprinting technique for the protection of sequential circuit IPs has been developed. It was the first dynamic fingerprinting scheme in which the embedded ownership and buyer's identity of an embedded IP can be conveniently detected off-chip by injecting a specific input sequence. The fingerprint is inserted by constraining the state encoding of a test machine embedded into a sequential circuit IP. A test machine is a special FSM that can be used as an alternative to the scan chain for improving the controllability and observability of sequential circuit IPs. The inserted fingerprint is in the form of a single fused signature that carries the information of the IP owner

and the buyer, as opposed to the simple concatenation of the two separate signatures. The signature is generated through a message exchange in a blind signature protocol. From the fingerprint, the IP provider can easily identify the buyer without divulging any sensitive information about the fingerprint yet the buyer's endorsement of the detected fingerprint is indisputable.

- 2) A licensing scheme have been proposed for FPGA IP cores, which provides an active control over the use of IP cores and facilitates the adoption of pay-per-use IP licensing model. The distributed IP instance is encrypted, and the decryption key can only be generated on a specific chip with the aid of a corresponding license tokens. This guarantees that the licensed IP core is only permitted to be implemented in those contracted devices. The encrypted IP core and all the security sensitive data are communicated in the encrypted authenticated form so that only unimpaired IP core is allowed to be implemented on the target devices. The licensing scheme employs symmetric cryptography and is directly applicable to commercially available FPGA devices. The scheme can also be modified to use asymmetric cryptography to provide stronger protection to the IP cores with only minor modifications to the FPGA devices. By exploiting the self-reconfiguring feature of modern FPGAs, the secure core installation modules of both schemes only occupy a small portion of the fabrics temporarily for the enrollment and installation of the licensed IP cores.
- 3) A new approach to detect possibly HT-infected or tampered IC chips in post-fabrication diagnosis for the chips manufactured by an untrusted fab has been proposed. The method is based on gate-level characterization (GLC) and employs post-silicon leakage current measurement. The developed method requires the characterization of only a subset of the circuit gates. The characterization results are estimated by solving the GLC problem efficiently using the normal equation in linear regression analysis. Based on the disparity of the bias term of linear regression and a subset of accurately estimated scaling factors, the HT-infected chip can be easily distinguished from the genuine chip as long as the HT to circuit ratio exceeds the detection sensitivity of the method. This method does not needs a gold model which is expensive to

obtain reliably. As the gates of the HT circuit consume leakage power at all time, the method also does not require the HT to be activated in order to detect it, making it possible to detect different types of HT, including those that are hard to be randomly triggered.

The above contributions have led to the publications listed in the author's publications at the end of the thesis.

1.4 ORGANIZATION OF THE THESIS

The thesis is organized into six chapters. Chapter 1 presents the research motivation, objective, as well as the major contributions achieved through this research.

Chapter 2 reviews the existing proposals for protecting the hardware IP cores in the literature. The review consists of three parts. Firstly, the hardware IP watermarking and fingerprinting techniques for the identification of the IP owner and IP buyer are discussed. As IP fingerprinting shares most attributes with IP watermarking, existing IP fingerprinting methods are illustrated in details after a brief overview of the IP watermarking methods. Next, the methods for metering the usage of FPGA based IP cores and their controlled configurations are discussed. Existing pay-per-use licensing schemes, especially those based on conventional crypto primitives, are demonstrated and discussed with their features and limitations highlighted. Finally, a comprehensive review of existing HT detection methods, which includes a detailed study of the HT taxonomy, is presented.

From Chapter 3 to Chapter 5, the contributions of this research are presented in details. Chapter 3 presents the developed dynamic fingerprinting scheme. The properties of the test machine are explained, followed by the illustration of fingerprint generation, insertion, and detection. The illustration includes the blind signature protocol which is used to generate the fingerprint that contains both the identity information of the IP owner and IP buyer, the fingerprint insertion method which exploits the properties of the test machine, and the techniques employed to reduce the embedding overhead and detection cost. Then, detailed security analyses and experimental evaluations are performed to examine if the proposed fingerprinting

scheme is able to efficiently generate a large number of high quality fingerprinted instances that are robust against perceivable attacks.

Chapter 4 presents the developed licensing scheme for FPGA based IP cores. The hardware primitives employed in the scheme, the execution of the licensing scheme, as well as the configuration steps of the licensed IP cores are illustrated in details. Then, the security analyses of the scheme against perceivable attacks, two variants of the proposed scheme that provide better security at the cost of increased complexity, the practicality of the scheme, and the implementation considerations which include overhead evaluations of countermeasures against common types of physical attacks are provided.

Chapter 5 presents the proposed solution to hardware Trojan detection. Preliminaries for gate-level characterization, its modeling by the system of linear equations and the solution based on the normal equation are presented. Then, the two phases, i.e., the HT indicator extraction phase and HT detection phase, of our simplified algorithm to the HT detection problem by gate-level characterization are explained. Subsequently, experimental results for examining the detection sensitivity of the method are presented and discussed.

Finally, Chapter 6 concludes this thesis by summarizing the results achieved in the research. Some promising ideas that are worthy of further investigation are also discussed.

Chapter 2

Preliminaries and Literature Review

2.1 INTEGRATED CIRCUIT IP PROTECTION

As identified by the VSI Alliance IP protection development working group [6], there are three main approaches for protecting IP cores in an IC. The *deterrent* approach uses legal means, such as patents, copyrights and trade secrets, to deter an illegal act from occurring. The *protection* (prevention) approach tries to prevent the IP infringement e.g., by license agreements and encryption. The *detection* approach, such as IP watermarking and fingerprinting, is characterized of enabling the IP owner to determine whether an unauthorized use has occurred and then trace the source of malfeasance. However, these three approaches each have their own limitations. The deterrent approach does not provide any physical protection to the IP core, making it only effective against benign IP users. Naive use of the prevention approach like encryption can only ensure the IP core is securely delivered to the IP user, but cannot prevent the IP user from misusing the decrypted IP core. IP watermarking and fingerprinting techniques can only passively confirm the occurrence of IP infringement after it has taken place.

A trusted platform for secure IP development, distribution and usage is desired in a vibrant IC market. Unfortunately, none of the approaches described above or their combinations is close enough to fostering such a safe environment in which the IP vendors (design houses) can have the assurance that their IP cores cannot be abused and no design information will be divulged or maliciously manipulated and the IP users (system developers) can have the assurance that their received IP cores are authentic and uncompromised. Except the deterrent approach which is based solely on legal enforcement, the other two approaches are realized and aided by technology. Technology-based or technology-aided protection methods play an important role in

assuring the authenticity of the embedded IP cores before and after their integration and fabrication. In this chapter, IP watermarking and fingerprinting techniques to identify the owner and buyer of an IP core, pay-per-use licensing schemes for the secure metering and configuration of FPGA based IP cores, and hardware Trojan (HT) detection methods for verifying the integrity of IP cores, especially those that enable post-silicon diagnosis of the IC integrity, will be reviewed.

2.2 HARDWARE IP WATERMARKING AND FINGERPRINTING

These techniques embed a specific tag into the IP core. The tag may be the same for all IP instances and carry only the ownership information (i.e., watermarking), or it may also contain different user information in the IP instances distributed to different users (i.e., fingerprinting). As the watermarked IP instances distributed to different IP users are identical, the watermarking techniques are unable to trace the guilty buyer from the unauthorized resold copies. In contrast, the malicious buyer who illegally redistributes his IP instance to another entity or misuses his IP instance in an unauthorized application can be easily identified based on the embedded fingerprint, as depicted in Fig. 2.1.

Figure 2.1 Illegal usage of an IP core and tracking the source of misuse by fingerprint.

My research in this aspect (i.e., IP passive tag) focuses on IP fingerprinting. Nonetheless, as IP fingerprinting shares most of its desiderata with IP watermarking and quite some fingerprinting methods are developed from watermarking approaches, a brief review of IP watermarking techniques is presented below, after which IP fingerprinting proposals found in literature are explained.

2.2.1 Hardware IP Watermarking

Similar to watermarking techniques for multimedia applications (e.g., pictures, audio, video or 3D models), hardware IP watermarking is realized by inserting covert and indelible ownership information into the target design for ownership proof. Derived from the generic model for digital watermarking [7], a generic model for hardware IP watermarking is depicted in Fig. 2.2. The watermark insertion process inserts the watermark into an IP core at some chosen design abstraction level, while the watermark extraction process defines how the watermark is extracted from the watermarked IP core.

Figure 2.2 A generic model for IP watermarking.

The general requirements for IP watermarking are very similar to those of multimedia watermarking. Nonetheless, multimedia watermarking has more freedom to alter the cover media and insert the watermark. It exploits human auditory or visual imperfections to achieve watermark imperceptibility and robustness. Such alteration is

restricted in IP watermarking, since the watermarked IP must remain functionally correct. Based on the requirements for an IP watermarking scheme proposed in [8-10], the following desiderata are outlined:

Maintenance of functional correctness: The functionality of the IP core should not be altered after the insertion of the watermark.

Independence of the secrecy of algorithm: According to one of the oldest security tenets defined by Kerckhoffs [11], the security of any encryption or security technique lies not in the secrecy of the algorithm, but on the mathematical complexity of such algorithm. Thus, security of the watermark should not depend on the secrecy of the watermark insertion or extraction algorithm but on some system properties.

Strong authorship proof: The watermarking scheme should be capable of inserting enough data for identification of the IP owner. The data should be of sufficient creditability to be considered as evidence in front of court for proving the authorship.

Low embedding cost: The embedding of the watermark should be made transparent to existing design processes. The embedding cost, including both the computational cost and time needed for embedding the watermark, should be kept low.

High reliability: The reliability of a watermarking scheme can be evaluated with the robustness and probability of coincidence (P_c) of the watermark. The robustness measures the strength of the hidden signature against various attacks, while the probability of coincidence, sometimes called the false positive rate, is the possibility that the watermark is detected in a non-watermarked design. For non-repudiation, the probability of coincidence should be at least as low as the odd of finding a match fingerprint from two persons in forensic science.

Low implementation overhead: it is usually inevitable to introduce additional overhead to the IP core after watermarking. The performance overhead, usually measured in terms of area, power and delay, should be kept low for the IP core to remain useful.

Ease of detection and tracking: Tracking and detection is as important as watermark insertion. It is advantageous to ease the detection of watermark and enable the origin of fraudulence to be traced after possible attacks.

2.2.1.1 Watermark Embedding

Although there are IP watermarking proposals [12-15] where the watermark is inserted by adding circuitry that generates the watermark, the mainstream techniques embed the watermark in the form of additional constraints. The abundant NP-hard optimization problems in the IC design flow are too complex to be solved for the exact optimum solutions by employing exhaustive enumeration. Instead, quasi-optimal or near-optimal solutions are sought by heuristic algorithms with some design constraints. This is where constraint-based IP watermarking techniques come into play. The heuristic algorithm takes the design specifications and its performance constraints as inputs for design space exploration to select a good solution as the original IP core from a large solution space. To create a watermarked IP, the encrypted authorship message is first converted into a set of steganography constraints. These constraints are then used as either additional inputs to the optimizer (i.e., pre-processing) or imposed on the output of the optimizer (i.e., post-processing). The final result will be a watermarked IP core which satisfies both the original and the steganography constraints.

A generic representation of the pre-processing based watermarking procedure is shown in Fig. 2.3. The watermark is derived from the authorship information based on some encryption processes as depicted in Fig. 2.2. It is then converted into steganography constraints by the constraint generator that directs the mapping from the watermark to the constraints. With the steganography constraints added to the inputs of the heuristic solver, the solution space of the *original problem* is reduced to be much smaller.

Figure 2.3 A generic pre-processing constraint-based IP watermarking procedure.

As the synthesis problem for generating the watermarked IP is non-linear and complex, the watermark inserted using the pre-processing approach is usually very robust. This is true especially when it is compared with the watermark inserted in the post-processing approach where the original IP core is first obtained from the synthesis problem and then altered based on the steganography constraints. Such alteration may also be exploited by an attacker to mask or remove the watermark. On the other hand, synthesizing the original problem with the steganography constraints is more likely to result in unpredictable design overhead. The quality of the watermarked IP cannot be guaranteed even for the optimization intensive watermarking techniques [16]. Hence, the steganography constraints in the pre-processing approach must be prudently selected. This problem is not so severe in the post-processing approach. As the steganography constraints are imposed on the already optimized solution to the original problem, the overhead due to the inserted watermark can be better controlled.

To limit the overhead and achieve high robustness simultaneously, a three-phase watermarking approach is proposed by Yuan et al. [5]. In the first phase, an optimized solution is obtained from the original synthesis problem. The optimized design is used as a reference in the second phase where some steganography constraints are selected by considering the overall tradeoff of solution quality, watermark robustness, etc. In the third phase, the watermarked design is re-synthesized to increase the difficulty for an adversary to tamper with the watermark. As suggested by the authors, the watermark should be inserted close to the end of the synthesis to reduce the complexity of the re-synthesis process in the third phase and to increase the predictability of the change in solution quality due to watermarking.

For all the three approaches described above, the strength of the authorship proof depends on the ratio of the steganography problem solution space to that of the original problem. The smaller the ratio, the lower the probability of coincidence (because a solution generated under only the original constraints will be less likely to also satisfy the steganography constraints), and the stronger the proof of the watermark existence.

The three generic watermark embedding procedures can be applied to various optimization problems in the SoC design process. Among them, the pre-processing

approach has the widest appearance. Since its first introduction in [17-20], an extensive number of constraint-based IP watermarking techniques have been proposed at different abstraction levels, which range from the system synthesis level [17, 20-24] to the behavioral synthesis level [25-27], logic synthesis level [27-31] and physical synthesis level [20, 27, 32, 33]. A detailed review of these schemes can be found in [5, 8, 34].

2.2.1.2 Watermark Extraction

A watermarking scheme is incomplete and turns out to be impractical if it is difficult to detect and track. Except the side-channel based watermarking approaches [13-15] where the watermark can be detected from a side-channel signal (e.g., temperature, power, electromagnetic radiation), a majority of IP watermarking schemes require processing the watermarked design so as to extract the watermark.

From the watermark extraction point of view, these watermarking schemes can be classified into two types. The first type is the static watermarking scheme, whereby the presence of watermark is verified indirectly by checking if the watermarked constraints generated by the author signature are satisfied. To verify the existence of the watermark, the circuit under test (CUT) usually needs to be reverse engineered to the level where the watermarked solution is generated. Then existence of the watermark is proved by checking if the steganography constraints due to the authorship signature are satisfied. All the pre-processing based approaches listed in Section 2.2.1.1 belong to this type. There are some concerns about such a detection process. First, reverse engineering a CUT to the level where the watermarked solution was originally generated is often a costly task. Second, the verification process usually needs to expose the grammar used for generating the steganography constraints from the authorship information. This can potentially weaken the security of other designs that are similarly watermarked. The method in [35] suggested the use of a pair of public and private watermarks to facilitate the authorship proof. However, it still needs to be verified if the added constraints of the design are satisfied.

The second type is the dynamic watermarking. This type of schemes is characterized by the watermark stimuli-response pair whereby the watermark bits can be detected from the output response by running the watermarked design with a specific input

sequence. Such an idea appeared first in [17], where it was proposed to embed a covert channel into a design in such a way that only the authors of a design can observe and interpret information obtained through the channel. The authors illustrate the idea with an example of a digital bandpass filter. The steganography constraints which encode a designer signature are added in the filter structure and by observing the outputs for the specific input segments, the embedded message can be identified.

For dynamic watermarking schemes, state transition graphs (STG) of finite state machine (FSM) at behavioral level and test structure such as scan chains at design-for-testability (DfT) level are the two common vehicles. FSM-based watermarking schemes [36-39] have the main limitation that once the watermarked IP is integrated into the chip, the watermarks hidden in the SOC after the chip has been packaged cannot be extracted in the field without dismantling the encapsulation.

The only signal that can be traced after chip packaging is the test signal. Thus, Besides the scheme [12] mentioned in Section 2.2.1.1 which combines a watermark generation circuit with the test circuit of the IP core at the behavior design level, numerous test structure based post-fabrication verifiable schemes [40-42] have been proposed. All these schemes enable field authentication of the authorship by the IP buyer after the chip has been packaged. However, as the watermarked test core of these schemes is independent of the functional logic, it is not a difficult task for an attacker to redesign the test circuit in order to completely remove or partially corrupt the watermark.

To cope with the limitation of the test structure based watermarking techniques, Chang and Cui [43] introduced a synthesis-for-testability (SfT) watermarking scheme. The watermark is embedded as implicit constraints to the scan chain ordering problem as in [41]. Unlike the DfT watermarking methods [12, 40-42], which are performed after logic synthesis, the SfT watermarking scheme inserts the watermark into the scan chain before logic synthesis. This helps to merge the test functions with the core functions, making the attempts to remove or alter the embedded watermark more costly as such attempts are now quite likely to impact the design specification and optimality. Nonetheless, the possible concern of this scheme is that SfT technique is not the mainstream testing technique in industry.

2.2.2 Hardware IP Fingerprinting

IP fingerprinting shares very similar desiderata with watermarking, such as functionality preservation, high credibility of proof, low embedding cost and design overhead, high robustness against attacks, transparency to existing design flows, and ease of verification. These desired attributes have been explained in Section 2.2.1. However, the requirements of low embedding cost and high robustness against attacks are more stringent for fingerprinting. The differences are highlighted by examining these attributes in the context of fingerprinting.

Low embedding cost: A large quantity of distinct high-quality fingerprinted IP instances, instead of the same instance, need to be generated for different buyers. Hence, the incremental embedding cost (mainly the time and effort) for each instance must be kept reasonably low. Caldwell et al. [44, 45] suggest that the run-time for creating every additional fingerprinted instance should be much less than that for solving the synthesis task from scratch. Qu and Potkonjak [46] even advocate that an effective fingerprinting scheme should be able to create $K \gg 1$ fingerprinted IP instances at the expense close to that for finding one single solution.

High robustness against attacks: The fingerprint needs to be robust against all attacks that can remove or mask a watermark. In addition, the fingerprint must also be collusion-secure. In a fingerprinting scheme, different IP buyers receive different fingerprinted instances. If the inserted fingerprints are the only disparities in different instances, it will be relatively easy for a group of malicious buyers to collude and remove the fingerprint (or more commonly, forge an IP instance from which no buyer in the colluding group can be traced). The collusion attack is especially straightforward if the fingerprint is independent of the functional logic. Hence, Qu and Potkonjak [46] postulate that the IP instance of an innocent buyer cannot be created from any combination of distributed fingerprinted IP instances of a robust fingerprinting scheme and at least one of the guilty buyer can be traced from the forged instance created by a collusion attack. As a rule of thumb, the fingerprint should be closely coupled with the functional logic, making any attempt to remove the fingerprint without affecting the IP functionality or rendering the IP instance useless impossible; the distributed fingerprinted instances should be structurally diverse to disguise the differences incurred by different fingerprints.

It is obvious that the intuitive way to create each fingerprinted instance by repeating the entire process of a typical constraint-based watermarking technique is impractical. While the required engineering time and effort for creating a single high-quality watermarked IP instance may be acceptably low, due to the multiplying factor for a large number of IP buyers, the cumulative fingerprinting cost is huge and unrealistic. It should be noted that IP fingerprinting is different from IC metering techniques where each fabricated IC chip has a unique identifier derived from the chip specific process variation [47, 48] or don't care conditions [49, 50]. IP fingerprinting is to create a unique IP instance for each IP buyer, where the fingerprint contains the information of the IP buyer. In fact, only a few methods [44-46, 51-53] have managed to reduce the total cost to a reasonable level, which are to be illustrated below.

Lach et al. [51, 52] proposed the first IP fingerprinting scheme for FPGA designs. The FPGA design can be partitioned into small tiles and each tile has several structurally different but functionally equivalent instances. For example, consider a segment of the design for a Boolean function $Y = (A \vee B) \vee (C \wedge D)$. This Boolean function can be implemented in a tile containing four configurable logic blocks (CLB's) and there are four different implementations that are interchangeable as shown in Fig. 2.4. For all the four implementations, there is one unutilized CLB which can be used to hold a part of the IP buyer's fingerprint (and also the IP owner's watermark). If the same implementation in Fig. 2.4(a) is used for all fingerprinted instances distributed to different IP buyers, the difference in the tile among the fingerprinted instances will be ascribed only to the inserted fingerprint, and simple comparison collusion will reveal the fingerprint bits. In contrast, by randomly using one of the four implementations for the Boolean function, the difference of various IP buyers' fingerprints will be disguised by the variation in functional structures. Attempts to tamper with the differences revealed by comparing fingerprinted IP instances may alter the functionality of the design and render the design useless.

Figure 2.4 Four instances of the same function in a tile with fixed interface [52].

Due to the employed tiling and partitioning technique, the required effort for generating each fingerprinted instance is greatly reduced. Using the same example depicted in Fig. 2.4, where the FPGA design is partitioned into four tiles and each tile can be implemented with four different instances, $4^4 = 256$ instances with different functional logic structures can be obtained. Assume that the required effort to place and route an entire design is E , then the effort required for the implementation of each tile instance is around $E/4$. The effort to implement each tile instance, amortized over its use in $4^3 = 64$ different design instances is $E/(4 \times 64)$. Hence the effective effort to generate each tiled design is only $E/64$.

As the scheme employs only the unused CLB's in an FPGA design, which always exist, the area overhead is low. However, creating the tiled design and generating functionally equivalent but structurally different instances for each tile may introduce non-negligible timing overhead for the resultant design. The vulnerability of this scheme is that the fingerprint is independent of the functional logic after all. The fingerprint can be removed by reverse engineering the fingerprinted IP instance to an abstraction level before the fingerprint was inserted.

As opposed to the previous scheme which is only applicable to designs with specific regular and highly granular structure, the scheme proposed by Caldwell et al. [44, 45] can be applied to almost all synthesis problems. The basic idea is to obtain an initial seed design by performing the synthesis task from scratch. For each IP buyer, a new fingerprinted instance can be generated with the buyer's fingerprint and the seed design, where only incremental iterative optimization is required. As the effort for creating the seed design is leveraged for generating each fingerprinted design, the scheme manages to reduce the cost of fingerprinting tremendously. Take the standard-cell placement problem for example. The target of this problem is to place each cell of a gate-level netlist onto a legal site with no overlap between two cells and minimized interconnection wire lengths. To create a fingerprinted solution, an initial placement solution S_0 is obtained from scratch. Then, a subset of signal nets N' in the design is selected according to the fingerprint. The weight of each net in N' is set to 10 with the remaining nets set to 1. Based on the solution S_0 and the new net weights, the fingerprinted placement solution can be generated by incrementally replacing the design. The pseudo-code of the fingerprinting process is depicted in Fig. 2.5. It should be noted that the seed solution for generating the fingerprinted solution S_i may not necessarily be S_0 . Instead, the solution S_{i-1} , i.e., the fingerprinted solution for the $(i-1)$ -th user, can be used. As S_{i-1} is a locally optimized solution, re-weighting selected signal nets will break the local optimality and facilitate the generation of a new fingerprinted solution S_i that is far away from S_{i-1} . Hence, the new seed design will help to generate fingerprinted instances that are more resilient against collusion attacks. However, as mentioned by the authors, the new solution will inherit the constraints from the previous fingerprinted solution and affect its quality.

- 1: Obtain an initial placement solution S_0 in LEF/DEF format.
- 2: For $i = 1$ to n (n is the number of IP buyers)
- 3: Select a subset $N' \subset N$ of the signal nets in the design according to the i -th user's fingerprint.
- 4: Reset the weights of all signal nets to 1.
- 5: Set the weight of each net in N' to 10.
- 6: Incrementally replace the design, based on the seed solution S_0 and new net weights, to obtain the fingerprinted placement solution S_i .

Figure 2.5 Pseudo-code of the iterative fingerprinting approach on standard-cell placement.

It is worth noting that the iterative fingerprinting approach can also be applied to optimization problems that cannot be solved by iterative improvement. For example, given a Satisfiability problem (SAT), new solutions cannot be generated from a seed solution by applying iterative improvement techniques. However, a fingerprinted solution can be generated by solving a new SAT problem with smaller size which is built by preserving the assignments of variables selected according to the fingerprint and removing them (along with their complements) from the initial SAT.

In the iterative fingerprinting approach, the incremental optimization effort required to generate each fingerprinted IP instance is still non-trivial. As a large number of fingerprinting instances are usually required, the total amount of effort may still be huge. Qu and Potkonjak [46] proposed a scheme with almost zero incremental effort to generate various fingerprinted solutions after a seed solution is found. The key idea is to introduce a set of independently relaxable constraints before solving the original design problem. From the obtained solution for the modified problem (namely the seed solution), a number of distinct solutions for the original design problem can be derived by independently relaxing each constraint. The authors proved this idea on the NP-complete graph coloring problem. Additional constraints can be introduced by duplicating a selected set of vertices, modifying small cliques or adding edges between unconnected vertices. Take as example the technique of duplicating a selected set of vertices on an original graph shown in Fig. 2.6(a). Vertex A is duplicated by adding vertex A' and connecting A' with all the neighbors of A . An edge is also added between vertices A and A' , resulting in a new graph shown in Fig. 2.6(b). After solving the graph coloring problem for the new graph, a coloring solution can be obtained, with vertices A and A' colored differently. From this solution, two coloring solutions for the original graph can be easily derived with vertex A taking one of the two different colors. Similarly, if k vertices are duplicated and one solution is found for the new graph, a total of 2^k different solutions can be derived for the original graph.

Figure 2.6 Duplicating vertex A to generate a solution from which two solutions can be derived for the original graph.

The three techniques mentioned above modify the graph to introduce additional constraints before the graph coloring problem is solved, and hence can be classified as the pre-processing approach. The quality of the derived solutions heavily depends on the added constraints. If the original graph is overly constrained, more colors may be used for the modified graph than the necessary number of colors for the original graph. To cope with this problem, the authors also present a post-processing approach. A solution to the graph $G(V, E)$ with k colors is first obtained. Then the vertices of the graph is partitioned into m subsets by their colors with the number of colors in each subset being $C_1, C_2, \dots, C_m$, respectively. The sub-graph formed by the i -th subset of vertices is C_i colorable. As the size of each sub-graph is in general relatively small, all the C_i -colored solutions can be found by exhaustive enumeration, denoted as n_i . In this manner, all the solutions for each sub-graph can be exhaustively found and a total number of $n_1 \times n_2 \dots \times n_m$ solutions can be derived for the graph.

The scheme, although very effective in generating a lot of different solutions with low overhead, lacks a clear mechanism to insert the fingerprint of high credibility. Although each derived solution is guaranteed to be distinct, how distinct the fingerprinted instances remain a question. If many similarities exist among fingerprinted solutions, the solutions will be vulnerable to collusion attacks.

All the three fingerprinting schemes described above focus on how to create a large number of different good quality fingerprinted instances at a low cost and how to make the inserted fingerprint robust against removal, masking and collusion attacks. As in the early-stage watermarking schemes, the ease of fingerprint verification has

been neglected. The fingerprint is verified by checking whether the corresponding constraints are satisfied, which is very similar to that of the static watermarking scheme introduced in Section 2.2.1.2. This static type of fingerprint verification process is cumbersome and costly. The verification complexity escalates after the fingerprinted IP is integrated and packaged into a system.

2.3 ACTIVE CONTROL OF THE IP USAGE

My research in this aspect focuses on the subset of active control methods which target at realizing secure pay-per-use licensing of IP cores on SRAM-based FPGA devices. Among all FPGA types, the SRAM-based FPGA has the widest range of applications. This is due to its denser construction, lower cost and better performance than other types such as the anti-fuse based and flash-based devices. In 2012, the SRAM-based FPGA accounted for 76.1% of the global sales revenue and its market dominance is expected to continue for years to come [54]. On the other hand, the configuration bitstream of the SRAM-based FPGA is also the most vulnerable target of attacks due to the fact that the SRAM memory is volatile and the bitstream need to be stored in an external non-volatile memory (NVM).

Pay-per-use licensing of FPGA based IP cores are advantageous over the upfront IP licensing model currently used in the FPGA IP market in that it is a better fit to the IP market. The blanket IP license fee is usually too expensive for the majority of FPGA-based system developers who target low-to-medium volume applications. In contrast, pay-per-use licensing possesses the following two merits: 1. the system developer with an application of low to medium volume no longer needs to pay the same expensive blanket license fee as a developer with a high-volume application. 2. By proportionally charging the system developer for the use of an IP core according to the sales of the system developed, the risk for the system developer is considerably reduced.

Pay-per-use licensing methods can be divided into two categories based on their reliance on either the conventional crypto primitives or the physical unclonable function (PUF). PUF is an emerging security primitive that exploits the native and unique variations in the physical properties of the devices induced by the manufacturing process. Due to the property of being unclonable and tamper-proof, the

PUF enables effective IP licensing schemes [55-57]. A detailed review of both types of schemes can be found in [58]. This thesis will focus on the former type of IP licensing schemes, i.e., the conventional crypto primitives based type. The merit of this type of licensing schemes is that they usually require minor or even no modifications to currently available commercial FPGA devices. Considering that no SRAM-based FPGA has yet been equipped with the PUF primitive and no FPGA Vendor, including Xilinx and Altera, has announced such a plan, they are more pragmatic in current FPGA IP market. Before the illustration of these licensing schemes, some background of FPGA IP market and commonly employed FPGA features are explained below.

2.3.1 The FPGA IP Market

2.3.1.1 Principals in the Market

The IP core vendor (CV) and the system developer (SD) are the seller and buyer of the market respectively. Besides these two principals who are directly involved in each IP transaction, another entity who participates indirectly in this process is the FPGA vendor (FV). The FV is the party who designs and sells FPGA chips. The bulk of FPGA fabrics are left uncommitted for the use by the SD. Nonetheless, it is also in the interest of the FV to include in some of their product families hard-wired functional primitives desired by most SDs to boost the chip sales. In general, the device architecture is different for the FPGAs purchased from different FV's, and even for the FPGAs from different device families of the same FV. The CV usually targets his design solutions to specific device families of the chosen FV's for optimized circuit performance and sells them as IP cores. When a transaction between the CV and the SD is committed, the IP core is usually customized according to the implementation requirements of the SD before the SD uses it in his system design. Using externally purchased IP cores and those designed in-house, the SD will then build a complete application, implement it on FPGA chips purchased from the FV, and sell the chips as producer goods or end products.

With the continuous scaling of the process nodes used for producing the FPGA devices, e.g., Xilinx 20 nm Ultra-Scale devices [1], building and maintaining the corresponding state-of-the-art fabrication facility requires a humongous running

expense. The capital investment cost drives most FPGA vendors, e.g., Xilinx and Altera, to adopt a fabless manufacturing model and outsource the chip fabrication to externally contracted fabs, called the *hardware manufacturer* (HM) here. In this business model, the FV pays the HM an upfront cost for the creation of the mask and for the required number of chips to be produced. However, a new problem emerges: although it is expensive and complicated to build a mask for the FPGA chip, the cost of producing extra chips is marginal for the HM after the mask has been created; the HM may produce excess chips and sell them in the grey market at a much lower price than the chips sold by the FV. This will greatly undermine the market revenue and deteriorate the brand name of the FV.

The activities of the four parties mentioned above and the interactions among them are depicted in Fig. 2.7.

Figure 2.7 Interactions and interests of the four main parties in the FPGA IP market.

Besides the four parties described above, there exists an additional party in the market, known as the *malicious attacker* (MA), who steals the valuable IP cores or the design information. An MA is not necessarily a stand-alone party. It can also be acted by some SDs. The possible attacks that may be carried out by the MA will be discussed in details in Section 2.3.1.2.

2.3.1.2 Common Attacks to FPGA based IPs

With the SRAM memory being volatile, designs for the SRAM-based FPGA are vulnerable to attacks. The bitstream is easy to be poached by eavesdropping on the bus connecting the FPGA chip and the external NVM. If in plaintext, the bitstream can be readily duplicated and implemented on uncommitted FPGA chips bought off-the-shelf. Such an attack is called *cloning*. Cloning of unprotected bitstream is straightforward and incurs almost zero cost.

In addition to directly cloning the design, an attacker may also *reverse engineer* [59, 60] the poached bitstream to extract the proprietary design information, and work out a competitive design at a much lower cost. The effort required for reverse engineering a design is not as trivial as that for cloning. Nonetheless, tempted by the market value of the design information, which is significant and inestimable sometimes, the attacker will have enough incentive to conceive such an attack.

The bitstream may also be *tampered* by the MA. For example, an attacker may tamper the bitstream in the set-top box, which controls the channels that the viewer can see, to bypass or even remove the security features to enjoy the services for free [61]. By tampering the bitstream, an attacker may also implant his own malicious logic, widely known as the hardware Trojan (HT) [62], into the design. The goal can be to access data stored in the FPGA, obtain more knowledge of the overall system, or even to hijack the system [63]. Due to the disastrous consequences of malicious tampering, rigorous integrity check is necessary for the third-party IP cores, especially when they are used in sectors like military, finance, energy and politics.

In addition, physical attacks are also commonly used by the MA. This kind of attack is capable of nabbing secret information by analyzing hardware characteristics of the FPGA that implements the target system. Physical attacks can be invasive or non-invasive. Invasive attacks are exemplified by the fault attacks [64, 65], which invade the device, tamper it for a desired fault and then analyze the difference between the correct and faulty outputs to extract sensitive information like the secret key. Powerful invasive attacks may involve sophisticated tools such as mechanical probes, focused ion beam (FIB) and scanning electron microscope (SEM) for studying the inner circuit structure of the chip. On the other hand, typical forms of non-invasive attacks are side-channel attacks, which infer the secret information from the side-

channel information generated while the data is being processed. The side-channel information commonly exploited are computation time [66], consumed power [67] and emitted electromagnetic radiation [68].

2.3.1.3 Desiderata of FPGA IP Licensing Scheme

According to the analysis of security threats to electronic transaction systems performed by IBM [69], a transaction security system is suggested to be designed to the security level of MEDIAN-HIGH (where a successful attack requires expensive equipment and special skills) instead of HIGH (where all known attacks are unsuccessful). Indeed, a system protected at security level HIGH is usually too expensive to be receptive in the market. There should be a good balance between the high security level and the system cost. Similar considerations apply to the FPGA IP licensing scheme. A licensing scheme that makes the IP cores immune to any attacks is not a judicious target. A practical licensing scheme should be the one that makes a successful attack too expensive to be compensated by the benefits obtained from the stolen IP cores; at the same time, the implementation cost for the scheme should be market viable. Hence, it is not necessary for a practical IP licensing scheme to be immune to the expensive physical attacks that use sophisticated tools to study the inner structure of the chip to learn the design information. These attacks exploit the weakness of the silicon technology; when they are possible, it is very difficult to secure the implemented design against them [70]. Nonetheless, it is usually too expensive to perform such an attack. Besides requiring advanced tools, it also needs special skills and tacit knowledge with a large amount of time and effort due to the size and complexity of the valuable IP cores and the FPGA chip today. Based on these considerations, we derive the following quality attributes that are essential for a good IP licensing scheme:

Broad Protection: The IP cores should be well protected against all the perceivable low to medium cost attacks such as cloning, reverse engineering, malicious tampering and common physical and side channel attacks.

Low Implementation Cost: The cost of enforcing the licensing scheme should be relatively low. The scheme should not complicate the already-complex manufacturing process. The use of new ancillary hardware, if any, should be well justified.

Transparency: The scheme should not pose any change to current CAD tools. Neither does it impose any difficulty in reconfiguring or updating the designs in the chip, nor does it impact the chip reliability.

Cryptographically Secure: The secret information should be well protected. The cryptographic algorithms used in the scheme, if any, must be widely accepted as secure.

2.3.2 Commonly Employed FPGA Features in IP Licensing Schemes

A brief explanation of the commonly employed device features is to be provided in this section. As the notations used for the same operation in different schemes may differ, the notations used will also be provided along with their exposition.

As discussed in Section 2.3.1.3, low implementation cost is one of the most important desiderata for an FPGA IP licensing scheme. An intuitive approach is to exploit existing features in FPGA devices as much as possible and to reduce the demand for additional hardware primitives. The existing features that are commonly used by the licensing schemes are listed and explained as follows:

Device identifier: The device identifier uniquely identifies each FPGA device. It can be a printed serial number or a bit string that is stored on chip. For example, on some Xilinx chips, a 57-bit device DNA is hardwired and can be read from the DNA port [71]. The device identifier for an FPGA device i is denoted as $\#ID_i$.

On-chip NVM and decryption engine: In SRAM-based FPGA devices that support bitstream decryption, the secure NVM used for secret key storage is usually designed to be only accessible by the hardwired symmetric key decryption engine due to security concerns. The decryption engine is also not allowed to use an alternative key and can only be accessed by the configuration controller. A commonly used standard for the decryption engine is Advanced Encryption Standard (AES) [72]. An IP core in plaintext is simply denoted as IP_j and its identity as $\#IP_j$. When it is encrypted with a secret key K_{ENC} , the encrypted IP core is denoted as $E(K_{ENC} : IP_j)$.

Keyed-hash message authentication code (HMAC): As mentioned in Section 2.3.1.2, the bitstream may also be maliciously tampered, which may cause disastrous consequences. Although cyclic redundancy check (CRC) codes have been used in all

bitstream formats to prevent faulty bitstream from being configured, they are not adequate to detect intentional malicious changes [73]. Some contemporary FPGA types have offered additional integrity check mechanisms that are more secure than the CRC code. For example, on Xilinx Vertex 6 and 7 devices the bitstream encryption/decryption feature is used in tandem with the SHA-256 HMAC [74] based authentication feature¹ [75]. The decrypted bitstream will only be configured on chip after its successful verification with the HMAC digest. The HMAC digest of IP_j under the key K_{MAC} will be denoted by $HMAC(K_{MAC} : IP_j)$. If no key is used, the generated hash digest is simply denoted as $h(IP_j)$.

Self-reconfiguration: In recent FPGA devices, the configuration controller can be accessed from the fabric through an internal configuration access port (ICAP). This enables the FPGA logic to control the reconfiguration of part of itself. As will be seen in the discussion of some licensing schemes, the feature of self-reconfiguration may be used to remove the burden of using auxiliary modules to direct the IP decryption and installation. Upon completion of their missions, these modules will be erased from the fabric to release the reconfigurable logic it occupied

2.3.3 Conventional Crypto Primitives based Licensing Schemes

Out of numerous existing proposals [76-81], three classic representatives are to be illustrated in this section. The first one is a licensing scheme for IP cores at a high abstraction level, while the remaining two are for IP cores at the bitstream level, which is the last stage of the FPGA design and implementation flow. For the latter two schemes, one uses asymmetric crypto to establish the secret IP encryption and decryption key and the other performs this task using the combination of an external TTP and symmetric crypto.

In most cases, the higher level the protected IP cores reside, the more parties and design tools are involved, and the more complicated the task of protecting the IP cores. As a result, it is considered that the most suitable target for core-level FPGA IP protection is the bitstream [79]. If the IP cores are communicated at this level, it is possible to set the security boundary at the FPGA chip, i.e., the IP cores can be kept in

¹ The encrypted and authenticated bitstream is generated by encrypting the plaintext bitstream, an HMAC key and the HMAC digest of the bitstream with the secret device key.

encrypted form after it leaves the site of the CV and before it is configured on the FPGA chip.

2.3.3.1 A Licensing Scheme for IP cores at a high abstraction level

This is the first pay per use based IP licensing proposal [78]. Besides the FV, CV and SD described in Section 2.3.1.1, a *trusted third party* (TTP) is also involved in the digital rights management protocol. The TTP is an organization that all parties in the FPGA IP market trust for managing the secret information related to the IP transactions. In this scheme, the FV is assumed to design and manufacture the FPGA chips, i.e., taking the roles of both the FV and the HM. The protocol requires that each FPGA contains a permanent secret device key, a symmetric cipher for both bitstream encryption and decryption, and a unique device identity that is available on request through the programming interface.

The three phases of the licensing protocol are depicted in Fig. 2.8. In the first phase, i.e., the *enrollment of FPGA devices*, *security tokens* are created by encrypting *user keys* with each FPGA's secret device key. The pairs of user key and security token are stored in the TTP's database indexed by the chip identifier, which will later be used for the creation of device specific encrypted bitstreams. In the second phase, the SD obtains the needed IP cores from the CVs. The IP cores are in encrypted form so as to prevent misuse or tampering by the SD, but they can be decrypted by the FV's CAD tool to allow processing. This can be realized with the public key cryptographic algorithms like RSA and ECC. After the SD has completed the system design, the CAD tool will use a *design key* to encrypt the bitstream of the system. The bitstream is different from the conventional one in that it contains the copyright information of all the licensed IP cores. It can be decrypted by the programming tool in the next phase, which has access to the secret information stored in the server of the TTP. Again, the bitstream encryption in the CAD tool and its decryption in the programming tool can be realized with the public key cryptography. In the last phase, the programming tool takes as input the encrypted bitstream from the CAD tool, the IDs of the FPGAs to be programmed and the billing information of the customer, decrypts the bitstream with the stored secret information, and then charges the customer and creates the device specific encrypted bitstream. To perform the billing and encryption process, the programming tool is connected with the server provided

by the TTP over the internet. Then the TTP's computer bills the customers and looks up in its database the chip ID to find the corresponding user key and token pairs. Next, the TTP sends one pair of the user key and security token to the programming tool, which will then encrypt the bitstream with the secret user key. Appended with the security token, the encrypted bitstream that can only be configured on a particular FPGA is created.

(a) Enrollment of FPGA devices.

(b). IP usage by the SD

(c). Creation of device specific encrypted bitstreams

Figure 2.8 FPGA IP licensing for high-level IP cores [78].

As security critical information is transmitted between the programming tool and the TTP's server, the communication between them must be secure and is suggested by the authors to be protected by standard internet security protocols. The chip configuration using the device specific encrypted bitstream is kept relatively simple. When the FPGA loads the bitstream, it first recovers the user key from the security token with its secret device key. With the user key, the bitstream is then decrypted before it is used for chip configuration. There are two main concerns for the proposed scheme. One is the requirement of hardwired circuitry for both the encryption and decryption, but only the on-chip decryption engine is available on the commercial devices. The other is that both the CAD tool and the programming tool contain secret

information. These secret information need to be carefully protected. Otherwise, a successful crack to any of these tools will cause the leakage of the IP content. Recognizing this risk, the author suggests some variants of the scheme in [82], where the tools run on the TTP's server. With no secret information contained in the tools at the site of the SD, these alternatives provide better security. But the price to be paid is that the TTP needs to be equipped with much more computational power and higher internet connection bandwidth.

2.3.3.2 A Licensing Scheme with Asymmetric Crypto based Key Derivation

Compared with the setup of the scheme in [78], a key derivation function (KDF) module based on public-key cryptography is added, which helps to securely transport and install the needed key for bitstream decryption. One novelty of the scheme is that it avoids the burden of the public-key functionality, which is usually resource-consuming, by moving the KDF module to a temporary configuration bitstream. After establishing the bitstream decryption key, the resources occupied by the KDF module can be released for use by the IP cores or other system blocks. Out of the four stages of licensing protocol described in [79], the first two are for generating the device specific bitstreams. These two stages are depicted in Fig. 2.9.

The KDF bitstream, which contains the private key SK_{FPGA}^i of the key pair, is the vital part of the proposed scheme and is encrypted with the secret device key K_{FPGA}^i . The asymmetric key pair for a specific chip i , $(SK_{FPGA}^i, PK_{FPGA}^i)$, and for a specific CV, (SK_{CV}, PK_{CV}) , will be used to establish a shared secret IP key (for IP encryption and decryption) between chip i and the CV, as shown in Equations (2.1) and (2.2).

$$K_{CV} = KDF(SK_{CV}, PK_{FPGA}^i, \#ID_i) \quad (2.1)$$

$$K_{CV} = KDF(PK_{CV}, SK_{FPGA}^i, \#ID_i) \quad (2.2)$$

Figure 2.9 FPGA IP licensing with asymmetric crypto based key derivation [79].

As the FV is the party who decides the secret device key K_{FPGA}^i and asymmetric key pair $(SK_{FPGA}^i, PK_{FPGA}^i)$, it is easy for the FV to gain access to the IP content. Hence, the FV is assumed to be a trustworthy and unbiased TTP in this scheme. To implement the scheme, the required modifications are: If secure non-volatile memory is used to store the K_{CV} of each IP core, such memory is not available in current FPGAs and need to be added; current FPGAs do not allow using the hardwired decryption engine to decrypt the IP core with K_{CV} and need to be modified to allow such an operation.

2.3.3.3 A Licensing Scheme with TTP and Symmetric Crypto based Key Derivation

Instead of assuming that the FV is fully trusted by other parties in the market, the scheme in [80] separates the role of the FV and the TTP. With an external TTP appointed in the licensing scheme, the asymmetric cryptography based KDF, which is used for secure key transportation and installation, becomes dispensable and is replaced by the symmetric cryptography.

After the blank FPGA chips are produced by the FV, they are sent to the site of the TTP for embedding the secret device key K_{FPGA}^i . The TTP also associates each FPGA

chip with a core installation module (CIM) before sending the chips back to the FV. The CIM bitstream serves a similar purpose as the KDF bitstream described above, i.e., to establish the IP decryption key on FPGA. The difference between them lies in: instead of the public-key crypto, the CIM of chip i contains one symmetric cipher and one dedicated secret metering key K_M^i . The metering key will be used for recovering the IP decryption key. The licensing protocol is depicted in Fig. 2.10.

(a) FV enrolls devices with TTP

(b) IP core enrollment

(c) IP core licensing

Figure 2.10 FPGA IP licensing with symmetric crypto and TTP [80].

The IP decryption process in this scheme uses the partial reconfiguration feature of the FPGA. The CIM bitstream is first decrypted with the secret device key K_{FPGA}^i by the on-chip decryption engine and configured in the user logic. The design in the CIM contains a symmetric cipher for decryption, an internal configuration access port (ICAP) interface and two key registers, as shown in Fig. 2.11. A key register holds the

metering key K_M^i . The other key register is empty initially and will be used to store the IP decryption key. The ICAP port will be used to access the configuration controller for configuring the decrypted IP cores. When the encrypted IP key $E(K_M^i : K_{IP1})$ for the first IP core is loaded, it will be decrypted by the metering key and saved in the empty key register of the CIM. Next, the first encrypted IP core $E(K_{IP1} : IP_1)$ will be loaded onto the CIM, decrypted with K_{IP1} , and transferred to the configuration controller through the ICAP port for configuration on pre-defined reconfigurable logic area. The above configuration process will continue for all the remaining licensed IP cores.

Figure 2.11 The symmetric cryptography based core installation module.

This scheme does not pose any modification requirements to currently available commercial FPGA devices. Nonetheless, to implement the scheme, the logistic fee for the FV to transfer all his FPGA devices to and fro the TTP's secure site may be very expensive. Besides, the lack of a direct vendor and client relation in the FPGA IP market makes it difficult to find an independent entity that is mutually trustable by all parties involved to assume the role of TTP.

The features and drawbacks of the three schemes discussed above are summarized in Table 2.1.

Table 2.1

Features and drawbacks of existing core-level IP protection schemes.

Scheme	Feature	Drawback
[78]	1. For IP cores at high level 2. Use an external TTP 3. Store secret keys in design and programming tools	A successful crack to any of the tools will cause a leakage of the IP content
[79]	1. For IP cores in bitstream 2. Use the FV as the TTP 3. Use asymmetric crypto to establish IP decryption key	1. Modify current devices to add secure registers and allow the registers to have access to hard-wired decryption engine 2. FV gains easy access to IP content
[80]	1. For IP cores in bitstream 2. Use an external TTP 3. Use symmetric crypto to establish IP decryption key	Requires the FV to transfer all devices to and fro the TTP to install the unique device key

2.4 HARDWARE TROJAN DETECTION

As mentioned in Section 2.3.1.2, hardware Trojan is a very powerful form of attacks that can cause disastrous consequences. Methods of detecting the HT are reviewed in this section.

Detection of the malicious and deliberately stealthy HTs can be extremely hard due to the following reasons. First, at the various design and fabrication steps, there are now many opportunities for an adversary to insert the HT. Inordinately large amount of HT instances of varying forms and sizes can be exploited. Hence, nothing can be assumed a priori about the HT such as its type, size, activation mechanism and effect. Second, the high complexity of SoCs makes it very difficult and costly to detect HT by means by physical inspection and reverse engineering methods. The former has very high false negative detection as HT occupies only a small fraction of the chip area and usually has no impact on the physical dimensions of the chip. The latter is destructive and it is impossible to examine all chips and cost prohibitive to examine a large number of chips. As HT can be selectively implanted into only a portion of the chip population, the absence of HT in some chips cannot guarantee its absence in the rest of the chips. Third, conventional testing methods are designed for the detection of faults and failures due to the design errors or fabrication process of a HT-free circuit and these tests are unlikely to be effective for HT detection. Even when 100% fault coverage for all types of manufacturing faults is possible, there are no guarantees that

the chip is HT-free [62]. In addition, the adversary (particularly the insider attacker) usually inserts the HT in such a way that it can evade detection by conventional testing methods. Fourth, with the decrease in CMOS feature size, environmental and process variations have an increasing impact on the measured parametric signals, which may conceal the effect of the HT and render HT detection methods using simple analysis of the parametric signals ineffective.

Effective defenses against the HT are founded on a thorough understanding of the attack. Before reviewing the HT detection methods in the literature, a detailed taxonomy of the HT is established.

2.4.1 Hardware Trojan Taxonomy

As one of the earliest HT taxonomies, Wolff et al. [83] focused on the HT type which consists of two components, i.e., the triggering logic and the payload activation logic. A simple example of such an HT is shown in Fig. 2.12. The triggering logic monitors a set of q input signals and will activate the payload at a specific input combination (i.e., triggering condition). The triggering condition is usually a very rare condition so that the HT is dormant during chip testing and normal use. For example, the triggering logic can exploit the test enable control line to keep the HT inactive during chip testing. The triggered value may even be delayed by including a k -bit counter, which makes detecting the HT even harder.

Figure 2.12 A simple Trojan with the triggering and payload logic.

Once triggered (namely activated), the HT will deliver a payload to the connected circuit nodes. The payload represents the effect of the HT and can be either

destructive or non-destructive. A destructive payload will affect the chip's normal operation or output, while a non-destructive payload may, for example, enable super-user privilege mode when operating at normal user mode, but has no direct impact on the circuit functionality. Both the triggering and payload logic can be either digital or analog based.

Wang et al. [84] proposed a more complete and detailed HT taxonomy as shown in Fig. 2.13, where the HT is classified according to its physical, activation, and action characteristics.

Figure 2.13 HT taxonomy proposed in [84].

The physical characteristics category describes the various hardware manifestations of the HT, where the HT can be partitioned based on its type, size, distribution and structure. For the *type* category, there are functional and parametric classes. The functional class includes HTs that are realized through addition or deletion of transistors or gates, while the parametric class refers to HTs that are realized through modifications of existing circuit wires and logic. The functional HT is usually represented in the form of a combinational or sequential circuit. Examples for the latter are the thinning of a wire, the weakening of a transistor and the modifications of a physical geometry. These modifications will sabotage the system reliability and

increase the likelihood of a functional failure. The *size* category accounts for the number of components in the circuit that have been added, deleted or compromised, whereas the *distribution* category describes whether the HT components are topologically close in the chip layout or dispersed across the layout. Finally, the *structure* category describes whether the adversary is forced to regenerate the circuit layout, i.e., change the circuit structure, to insert the HT. Any change in the circuit layout can change the delay and power characteristics of the chip, and hence make the HT easier to be detected.

The activation characteristics refer to the way the HT is activated. Based on this criterion, the HT can be divided into *externally activated* and *internally activated* sub-categories. The former type can be activated by an embedded receiver or antenna and/or some specific input signals. The latter type can be further divided into two subclasses, i.e., *always-on* and *condition-based*. The always-on HT is always active and can disrupt the circuit functionality at any time; a typical form of this HT type is the parametric class of HT described above. In contrast, the condition-based subclass refers to the HT that is not active until a specific condition is met; these HTs possess the typical form of a triggering logic and a payload logic shown in Fig. 2.12 and are implemented in the form of a functional HT.

The action characteristics refer to the disruptive behavior that the HT will introduce, based on which the HT can be divided into three classes, i.e., modify-function, modify-specification and transmit-information. The actions of these three classes of HTs are self-explanatory. It is worth to note that the modify-specification HTs, which are realized by changing the circuit's parametric properties, have a typical action of incurring the system failure, whereas the modify-function HTs may essentially cause unlimited number of different changes to the system.

The above two taxonomies assume that the HT is inserted to an originally genuine system, which typically occurs in the chip fabrication phase at the site of an untrusted merchant foundry. In fact, the HT can be inserted in other phases. As proposed by Karri et al. [85], the HT can also be inserted in the system specification phase, design phase, post-fabrication testing phase, as well as assembly phase. Take the design phase for example. The HT might be inserted in any components that aid the design creation, such as the third-party IP cores (e.g., in the FSM [86] or RTL design

[87]) and the standard cell library. An attacker can also maliciously modify the CAD tools and/or scripts so that the generated design will be infected with the HT. In some cases, the modified CAD tools may even mask the effects of the HT by not revealing the HT circuit to the user.

2.4.2 Hardware Trojan Detection Methods

Among all the opportunities to inset the HT, the third-party tools, scripts and IP cores that aid the design activity and the manufacturing processes in the merchant foundry are the most likely to be employed. For the external IP cores and the generated designs from the third-party tool and/or scripts, it is extremely difficult to verify the trustworthiness. This is because there is no golden model with which the IP cores or generated designs can be compared. A two-phase method is proposed in [88, 89] to detect the possible HTs in the soft IP cores. The authors assume the availability of detailed specification requirements of the IP core by the system integrator, which is trustworthy. The requirements are used to verify the integrity of the IP core. The basic idea of the method is that without the redundant circuit and HT, all signals in the IP core are expected to change their values during the verification and the IP core should function perfectly. In the first phase, a high coverage test bench is generated, upon which the signals that stay stable are considered suspicious. In the second phase, the suspicious signals are analyzed. To reduce the number of suspicious signals and simplify the analysis, techniques like redundant circuit removal and a developed suspicious signal equivalence theorem by the authors are used. The remaining suspicious signals are then activated by test patterns generated by a sequential automatic test pattern generation (ATPG) to verify whether the behavior of these signals complies with the specification requirements. To facilitate the activation of the signals, designers may add new circuit structures to increase controllability.

The HT in the fabricated chips may be detected by physical inspection and reverse engineering methods. However, as discussed before, the destructive methods are usually too expensive and cannot be applied to the whole chip population. A more practical choice is the non-destructive method, which can be generally categorized as logic-testing based or side channel analysis (SCA) based.

2.4.2.1 *Logic testing based approaches for HT detection in fabricated chips*

Logic-testing based approaches try to generate stochastic test patterns to activate the HT so that its effect can be detected at the circuit output. A randomization based probabilistic approach is proposed in [90] to detect the possible HT in combinational circuits. The authors showed that a unique probability signature, which is essentially the probability of getting 1 at the output, can be constructed for each circuit with input vectors selected randomly from a specific probability distribution. When the circuit is modified, i.e., inserted with an HT, its probability signature will change. Hence, to detect the HT, the input vectors are applied to the IC under authentication (IUA) and the outputs are compared with those from the design possessed by the system integrator. If the outputs of the IUA and the design differ for any input vector, an HT is present. If a set of input vectors have been applied and the outputs are the same, a confidence interval can be obtained that the IUA has the same probability signature and is HT free.

Wolff et al. [83] proposed to analyze the internal nets of low controllability and observability in the circuit. These nets are the likely targets for the HT triggering logic and payload logic, respectively. To verify the existence of the HT, the authors suggested generating a set of test vectors that can activate the suspected HT triggering logic (i.e., a subset of the low-controllability nets) and propagating the payload (i.e., a subset of the low-observability nets) to the circuit output. A similar method is proposed by Chakraborty et al. [91], where the basic idea is to detect low probability conditions of the internal circuit nodes and then derive an optimal set of vectors that can trigger each of the nodes to its rare logic value multiple times. As the HT is usually attached to the rarely activated nets, the test vectors from the two methods have a higher probability of activating the HT compared to purely random test patterns.

Since logic-testing based approaches verify the existence of the HT by functional validation using logic values, this class of methods has the merit of high immunity to the increasing process variations in the state-of-the-art silicon processing technologies. In particular, these methods are effective in activating small HTs which has relatively simple triggering logic. However, they are feeble in activating and hence detecting relatively large HTs which have complex triggering logic. Besides,

they are just effective for detecting functional HT, not parametric HTs. Fortunately, side-channel analysis based approaches are strong in these two aspects.

2.4.2.2 *Side-channel based approaches for HT detection in fabricated chips*

The inserted HT, regardless of functional or parametric based, will probability have an impact on the side channels of the circuit, such as the transient power, leakage current, path delays, chip temperature or electromagnetic emission. The first side-channel based HT detection approach was proposed in [92], where the power signals are measured under a set of I/O tests. A few ICs are randomly selected from the chip population and destructively tested to ensure that they are genuine (or HT-free). The power signals from these ICs will be used to build a *power fingerprint* for the chip population. As the measurement noise can be minimized with multiple measurements, the difference between power signals of a genuine IC and those of an HT-infected IC lies in the components contributed by the PV and the HT. The power fingerprint is essentially an estimated statistical distribution of the power signal due to PV. For each IUA, its power signal is compared with the power fingerprint. As long as the statistical distribution of the power signal contributed by the HT have components in a subspace that is not spanned by the distribution of the PV, presence of the HT can be noticed. The signal subspaces and components are computed by Karhunen–Loève expansion [93]. The method is effective when the HT is relatively large or the PV is small. Otherwise, the impact of the HT will be masked by that of the PV.

Jin and Makris [48] attributes the limitation of power fingerprint in detecting small-size HT to the vagueness of power fingerprint in representing the complete characteristics of a large circuit. In their method, a series of path delay fingerprints are used instead. The fingerprint generation procedure is similar to that for generating the power fingerprint in [92]. High coverage input patterns are run in the selected sample chips and high-dimension path delay information are collected. According to the path delays, a series of delay fingerprints are generated and mapped to the low-dimension space. The delay fingerprints will then be used for verifying the delay information of each IUA. As the HT is always significant in the path view, the path delay fingerprints have a high sensitivity to detect the HT. However, as the circuit size increases, measuring the delays for all paths will become impractical.

The idea of retrieving the statistical distribution of the measured delay for each circuit path over a number of ICs is also employed in [94] for HT detection. Instead of performing external timing measurements to collect the delay information, a specific architecture using a shadow register is used. The architecture is depicted in Fig. 2.14.

Figure 2.14 Path delay characterization with a negative-skewed shadow register [94].

The main circuit is a register-to-register combinational path that is to be characterized (i.e., for delay measurement). The components outside the dotted box are the testing circuitry inserted to perform the delay characterization. The shadow register takes the same input as the main circuit, but is triggered by a shadow clock *Clk2*, instead of the system clock *Clk1* that triggers the main circuit. The two clocks run at the same frequency, but a negative phase shift (i.e. negative skew) is applied to *Clk2* to trigger the shadow register before the destination register by a precise amount of time. To characterize the path delay, the number of skew steps between the two clocks is increased one at a time until the results in the shadow register and the destination register are unequal. The path delay can then be determined by subtracting the phase shift from the system clock period. To overcome the problem that the chip temperature affects the path delay, an on-die temperature monitor (based on a ring oscillator and a counter) is used. Based on the known temperature-delay relationship, the effective delay can then be calculated from the reported temperature and delay.

Again, this method is impractical for a large circuit with a mass of paths, where a significant area overhead will be incurred.

If there were no PV, HT detection by employing side-channel signals would be straight-forward. However, the reality is that the HT activity is usually negligibly small compared to the entire circuit (or system) activity. The impact of the HT on the measured side-channel signals can be easily masked by that of the PVs, which increases with every generation of nano-scale technologies. The solution to this problem adopted by the above three methods [48, 92, 94] is to build the statistical distribution of the side channel signal due to the PV. To accurately estimate the distribution, it usually requires a number of sample chips which are destructively tested beforehand to verify that they are genuine. Obviously, such a solution is expensive.

To reduce the impact of the PV on side-channel signals, a second solution is to employ localized (or region-based) SCA [95, 96]. The contribution of the HT on side-channel signals may be so small that its effect is submerged into envelop of PV effects. However, the ratio of HT contribution to circuit contribution (denoted as *HT contribution ratio* for short) of a region will be much larger. Hence, HT detection capability can be greatly improved when the side-channel signal is measured locally. For example, regional transient power measurement can be performed via power pads or controlled collapse chip connection (more widely known by its acronym C4) bumps. Alternatively, region-aware test patterns [97-99] can be used to divide the circuit into multiple regions. Each region is selectively activated while the activities in the rest regions are minimized to magnify the HT contribution ratio in the measured circuit transient power. In addition to the localized SCA, signal calibration techniques [96] can also be employed to further attenuate the PV effect in order to increase the distance between HT-free and HT-infected chips.

Gate-level characterization (GLC) approaches provide another solution to cope with the increasing impact of the PV. These approaches construct linear extrapolation between the gate-level properties and non-destructively measured side-channel signals to formulate a system of linear equations (SLE). PV is an integral part of the gate property and is represented as a scaling factor to each nominal gate value in the SLE. The nominal gate value is readily available, e.g., it can be obtained from the

simulation models, and the gate scaling factors are variables to be solved. Depending on the choice of objective function for the SLE, the formulated problem can be stated in different formats, such as linear, convex or nonlinear. The existence of HT will show up by the abnormal scaling factors in the solution. To facilitate the solving of the maximal number of scaling variables from the SLE, Potkonjak et al. [100] proposed to select those that can create an SLE matrix with the highest rank by analyzing the linear equations. Singular value decomposition (SVD) and linear programming maximum likelihood procedure is then used to solve the SLE. Finally, learn-and-test and re-substitution statistical validation techniques are used to estimate the bounds of the calculated scaling factors.

The HT detection approach in [101] solves the SLE matrix with an SVD technique. The employed side channel signals are leakage power and path delay. The authors point out that the leakage power based SCA is effective for HT detection. Although the HT can be inserted so that no timing path from the primary inputs to the FFs and from the FFs to the primary outputs is altered, the leakage power is always altered. Even if the attacker gates the added HT circuitry, the gating requires an additional gate which draws leakage power. Existence of the HT is verified based on whether there is a systematic shift of all the scaling factors, as the contribution of the delay and the leakage power of the HT will be attributed to the gates in the SLE matrix. In particular, the method will conclude that the HT is present if the average scaling factor values of all gates are above the average for other circuits. The assumed attack model is not clearly presented, which should be that the HT is selectively inserted to a portion of the manufactured chips. The method is effective if the HT is relatively large compared to the infected circuit. Otherwise, the increase in the average scaling factor values of all gates due to the HT may not be conspicuous enough to be detectable.

The above linear programming based GLC method is also used in [102, 103]. The leakage current is the measured side channel signal. In these two proposals, a new variable representing the HT circuit is added to each equation of the SLE. Based on the calculated value for the HT variable, the existence or absence of HT can be determined, i.e., there is no HT if the HT variable is close to 0 and HT exists if the HT variable is large. The two proposals also provide effective solutions to two major

challenges for solving the SLE. The two challenges are: 1. two or more scaling variables can be correlated in all equations and thus their actual values cannot be calculated. 2. the SLE for a large circuit, which may contain millions of gates, cannot be solved in a reasonable amount of time. For the first challenge, a thermal conditioning technique based on gate switching, as depicted in Fig. 2.15, is proposed. Since gate-level leakage power depends on gate temperature, input vector control is conducted to selectively increase the temperature of the subset of correlated gates by different amount. The temperature change is due to the heat generated from the gate switching activities. As a result, the nominal leakage values of the correlated gates are different and correlations among the gates are broken. For the second difficulty, a divide-and-conquer paradigm is employed. The large circuit is divided into small sub-circuits by using input vector control. With each circuit segment containing just a small number of gates, accurate GLC can be performed. The segmentation technique based on input vector selection is shown in Fig. 2.16. The possible resultant large segments are then further divided by inserting some test points.

Figure 2.15 Flow of thermal conditioning for breaking gate correlations in GLC [102].

Input: Input Set $PI = \{PI_i | 1 \leq i \leq n_i\}$
 Controlled Gate Set $G(PI_{sub}) = \{G_{sub} | G_{sub} \text{ is controlled by } PI_{sub}\}$.
 Output: Selected segment set Seg .

```

1: while not all gates covered in the circuit do
2:   Select starting  $PI_i$  close to the uncovered gates
3:    $PI_{sub} = \{PI_i\}$ ; //Selected input set
4:   repeat
5:      $E_{cur} = E(PI_{sub})$ ; // Current GLC arruracy
6:      $Seg_{cur} = G(PI_{sub})$ ; //Current segment
7:     Add  $PI_j$  to  $PI_{sub}$ , where  $PI_j \notin PI_{sub}$ ;
8:   until  $E_{cur} < E(PI_{sub})$ 
9:   Add  $Seg_{cur}$  to  $Seg$ ;
10: Return  $Seg$ ;
  
```

Figure 2.16 Circuit segmentation technique to enable scalable GLC [103].

Alkabani and Koushanfar [104] proposed to verify the existence of the HT based on the consistency of scaling variables among different runs of the GLC process. The side-channel signal employed in this method is also the leakage current. Instead of linear programming, a quadratic program is formulated. Scaling variables are iteratively reweighted with Gaussian kernel function. The iterative loop, as shown in Fig. 2.17, ends when the sum of the squared distance between the most recent estimate and the initial estimate is within a threshold. Then based on the highest change in the gates' scaling factors, the existence or absence of HT can be concluded.

```

1: Solve the SLE for initial values of scaling factors ( $\phi^0$ )
2: Calibrate the scaling factors
3: Consistency = 0;  $i = 1$ ;
4: do
5:   use the kernel to reweigh the scaling factors;
6:   adjust the measurements accordingly;
7:   re-estimate the scaling factors ( $\phi^i$ ) ;
8:    $i++$ ;
9:   while ( $\Delta^2 \geq \Delta_{th}^2$ ); //  $\Delta$  is the sum of squared distances between  $\phi^i$  and  $\phi^0$  of all gates
10: Identify the anomalous gates by comparing  $\phi^i$  and  $\phi^0$  ;

```

Figure 2.17 The iterative HT detection algorithm in [104].

Wei and Potkonjak [105, 106] proposed another consistency based approach. Based on the controlled input vectors, the IUA is divided into multiple segments and the GLC is then performed for each segment. Overlapping gates across segments will have multiple estimated scaling factors. For each of the overlapping gates, its scaling factor values will be consistent if all the overlapped segments containing this gate are HT-free and inconsistent if the HT is present in one or more segments. In this method, proper segmentation of the circuit is very crucial. If the segment is large and the HT poses little impact on the scaling factors of overlapping gates, the existence of HT will not be noticed.

There is no silver bullet in the field of HT detection. With so many possibilities for the type, size, activation mechanism and effect of the HT, which are usually unknown before testing, a more judicious way to identify the possible HT is to use HT detection methods in a hybrid manner. As mentioned above, the logic testing based HT

detection approaches and SCA based approaches are complementary. The former is effective in triggering ultra-small functional HT with simple triggering logic, while the latter is good in detecting a complex HT which contributes relatively large side channel signals. The combined use of the logic-testing based and SCA based approaches will help to provide a high overall detection coverage of HTs of varying types and sizes [107, 108].

Besides, the HT detection methods can be incorporated with design methods that can facilitate HT detection. For example, a dummy FF insertion procedure is proposed in [109] to increase the transition probability of rarely triggered nets in the circuit. The method improves the HT detection sensitivity of transient power based side-channel analysis by increasing the number of transitions in a functional HT and the sensitivity of a logic-testing based approach by increasing the opportunity of fully activating the HT and transferring the erroneous effect to the circuit output. Such a design method can be classified as *design for hardware trust*, which is gaining increasing attention. It fulfills multiple objectives of preventing the insertion of HT, facilitating easier HT detection and providing effective IC authentication [62].

2.5 CHAPTER SUMMARY

In this chapter, a comprehensive literature review is provided for IP fingerprinting, per-device licensing of FPGA based IP cores using conventional crypto primitives and HT detection for IP cores and fabricated chips. The survey in this chapter aims to identify the limitations and shortcomings in existing approaches to the three important hardware security problems and path the way to the development of better solutions. The fingerprint verification processes of existing IP fingerprinting schemes are cumbersome and costly. To address the even more challenging problem of fingerprint verification after the fingerprinted IP core has been integrated into a packaged system, a new fingerprinting scheme for sequential circuit IPs will be presented in Chapter 3, where the embedded fingerprint can be conveniently detected off-chip by injecting a specific input sequence. The survey of existing FPGA IP licensing schemes shows that some modifications to commercial FPGA chips are required for their implementation. This problem is overcome in a more pragmatic IP licensing scheme

to be presented in Chapter 4, which employs only existing features of recent FPGA devices and eliminates the need for a TTP. The scheme enables immediate application of the pay-per-device licensing model of FPGA IP cores in the market. Finally, to reduce the computational complexity of GLC for HT detection, an agile HT detection method, which does not require accurate characterization of each gate property, will be presented in Chapter 5.

Chapter 3

A Blind Dynamic Fingerprinting Technique for Sequential Circuit Intellectual Property Protection

3.1 INTRODUCTION

A dynamic fingerprinting technique for the protection of sequential circuit IPs is to be presented in this chapter. As opposed to the static fingerprinting schemes described in Section 2.2.2, this dynamic scheme enables the embedded IP user's fingerprint (and IP owner's watermark) to be conveniently detected off-chip by injecting a specific input sequence. A typical way to insert both the IP owner's watermark and buyer's fingerprint is by concatenation. This naive way doubles the signature length and increases the steganography constraints. In this scheme, a single fused signature that carries both the watermark and fingerprint information is generated through a message exchange in a blind signature protocol. The fused signature can be deemed as the fingerprint to be inserted in this scheme and is referred to as the fingerprint in the remainder of this chapter. Each IP buyer will receive a unique fingerprinted instance where the fingerprint contains his information. The IP buyer will then use this IP core as a module of his system. Once the design of the system is completed, the blueprint of the system (e.g., in the form of GDS-II) will be sent to the foundry for fabrication. By running the IC that contains the fingerprinted IP with a specific input sequence bounded to a buyer, the fused signature can be extracted from the output response. The IP provider can easily identify the buyer without divulging any sensitive information about the fingerprint yet the buyer's endorsement of the detected fingerprint is indisputable.

The fingerprint is inserted by constraining the state encoding of a test machine to be embedded into and synthesized with a sequential circuit IP. The property of the test machine is to be explained in Section 3.2. In a nutshell, the test machine can be used

as an alternative to the scan chain for improving the controllability and observability of the sequential circuit IP. As the test machine is synthesized with the design, the fingerprint encoded into the state variables is well integrated with the functional logic and has a global influence on the circuit structure, making the fingerprinted instances inherently collusion-resistant. To reduce the effort of generating a large number of fingerprinted instances, variable chaining heuristic and dependency-directed partitioning are proposed to break a large sequential circuit into multiple smaller interconnected segments for test machine embedding. The optimized testable segmented circuits are used as seed design for state re-encoding according to the fingerprint. The overheads of different fingerprinted instances have been greatly reduced by the fusion of watermark and fingerprint, the state variable dependency minimization and the pre-optimization of partitioned seed designs. The method is applicable to sequential circuit IPs targeted for both ASIC and FPGA implementations. As there is no prerequisite of specialized scan flip-flops (FFs) or specific FPGA configuration fabrics, the scheme is transparent and applicable to different technology families of FPGAs.

The rest of this paper is organized as follows. Section 3.2 introduces the property of test machine and the fundamentals of test machine embedding. The fingerprint generation, embedding and detection of the proposed fingerprinting method are presented in Section 3.3. Section 3.4 analyzes the security of the proposed scheme with respect to its credibility, robustness and feasibility. Experimental results are presented in Section 3.5. Finally, this chapter is concluded in Section 3.6.

3.2 PRELIMINARIES ON TEST MACHINE INSERTION

The fingerprinting approach is founded on the test machine embedding problem of synchronous sequential circuits [110-113]. This section presents the construction of an n -FF test machine and explains how its test property can be embedded into an n -FF sequential function to synthesize a testable design.

3.2.1 Test Machine Fundamentals

An FSM is a quintuple $M = (S, I, O, \Delta, \Lambda)$, where S is a finite set of N states $\{S_1, \dots, S_N\}$, I is a finite set of p primary inputs $\{y_1, \dots, y_p\}$, O is a finite set of q

primary outputs $\{z_1, \dots, z_q\}$, $\Delta: S \times I \rightarrow S$ is the state transition function and $\Lambda: S \times I \rightarrow O$ is the output function [114]. M can be represented by a state transition graph (STG), which is a directed graph whose vertices and edges correspond to the states and state transitions of M respectively; each edge is labeled with the input and output associated with the transition.

An FSM can be tested by checking if its transitions $t_{ij} = (S_i, S_j; y/z)$, $S_i, S_j \in S$, $y \in I$, $z \in O$, $S_j = \Delta(S_i, y)$ and $z = \Lambda(S_i, y)$, are implemented as specified by its STG. This simple conformance test requires a sequence of inputs $y \in I$ to bring the physical FSM from any state to S_i and a sequence of inputs to verify that the FSM reaches the designated state S_j after the stimulating transition t_{ij} .

A *synchronizing sequence* (SS) is defined as an input sequence which, when applied to any initial state of M , will drive M to a single specific state [115]. Likewise, a *distinguishing sequence* (DS) can also be defined as a sequence of inputs such that for any state of M , the sequence of outputs generated in executing that sequence uniquely identifies the state. The output response that uniquely identifies the state is called the *distinguishing response* (DR) of the state.

An n -FF test machine is a special FSM, denoted by $M' = (S', I', O', \Delta', \Lambda')$, with $N = 2^n$ states, one input y' and one output z' . Any states $S_i \in S', i \in [0, N-1]$, of it can be set by applying an associated n -bit SS. At the same time, each state can be distinguished at the output by applying a DS of any n bits. The test machine can be constructed as follows:

$$\begin{aligned} \Delta'(S_i, y' = a) &= \begin{cases} S_{2i} & \text{if } 2i < N \\ S_{2i-N} & \text{otherwise} \end{cases} \\ \Delta'(S_i, y' = \bar{a}) &= \begin{cases} S_{2i+1} & \text{if } 2i+1 < N \\ S_{2i+1-N} & \text{otherwise} \end{cases} \end{aligned} \quad (3.1)$$

$$\Lambda'(S_i, y') = \begin{cases} b & \text{if } i < N/2 \\ \bar{b} & \text{otherwise} \end{cases} \quad (3.2)$$

where $a, b \in \{0,1\}$ are the arbitrary constants assigned to the input and output of M' .

Fig. 3.1 shows the STG for a 3-FF test machine. Let $SS(S_i)$ be the binary representation of SS of state S_i . By convention, the least significant bit (LSB) of $SS(S_i)$ is input first. One important property of the test machine is that the output response is different from each initial state S_i regardless of the input sequence. Hence, any 3-bit sequence can be used as the DS. Let $DR(S_i)$ denotes the binary representation of DR of state S_i such that the LSB of $DR(S_i)$ is output first. The binary representations of SS and DR for the 3-FF machine in Fig. 3.1 are shown in Table 3.1. If $a = b = 0$, then $SS(S_i)$ and $DR(S_i)$ will be equal to the bit reversal of the binary index i of S_i .

Figure 3.1 STG of a 3-FF test machine.

Table 3.1

Synchronizing sequences and distinguishing responses of 3-FF test machine.

SS	To state	DR							
		S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
aaa	S_0	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}bb$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$\bar{a}aa$	S_1	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}bb$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$a\bar{a}a$	S_2	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}bb$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$\bar{a}\bar{a}a$	S_3	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}bb$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$aa\bar{a}$	S_4	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}bb$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$\bar{a}a\bar{a}$	S_5	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}bb$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$aa\bar{\bar{a}}$	S_6	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}bb$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$
$\bar{a}\bar{a}\bar{a}$	S_7	bbb	$\bar{b}bb$	$b\bar{b}b$	$\bar{b}\bar{b}b$	$bb\bar{b}$	$\bar{b}bb$	$b\bar{b}\bar{b}$	$\bar{b}\bar{b}\bar{b}$

3.2.2 Test machine Insertion

For an arbitrary FSM, due to the limited controllability and observability, it is not always possible to obtain the SS and DS for all the states. These limitations are

usually overcome by replacing the FFs of FSM by specialized scan FFs and chaining them to externally added serial scan input and serial scan output pins. However, such scan insertion procedure is not a useful platform for fingerprinting a sequential design as it is difficult to keep the embedded signature stealthy and robust due to the easily traceable and removable scan FFs and scan IOs. Without introducing specialized scan FFs and additional IO pins, an alternative approach is to insert the test function into the gate-level design of M before logic synthesis by test machine insertion [113].

An n -FF test machine $M' = (S', I', O', \Delta', \Lambda')$, can be embedded into an arbitrary n -FF FSM $M = (S, I, O, \Delta, \Lambda)$ by isomorphic mapping of the states and IO symbols of M to those of M' . The input y' and output z' of M' can be shared with an input $y \in I$ and an output $z \in O$ of M , respectively by using multiplexers. To share the same set of FFs for multiplexing the next state functions, $\Delta'(S', y')$ and $\Delta(S, y)$, and the output functions, $\Lambda'(S', y')$ and $\Lambda(S, y)$, of M' and M , respectively, the state variables, $x'_1, x'_2, \dots, x'_n$, of M' must be mapped to the state variables, $x_1, x_2, \dots, x_n$, of M . The mapping can be performed in polynomial time (proportional to the number of FFs in the target machine) for completely specified function without adding a new transition to M [112]. If M has unspecified transitions for input y at some state in S , M' may not be compatible to M . To obtain an isomorphic mapping, unspecified transitions may have to be added to the STG of M . A new test-enable (TE) input can then be introduced into M to select between the normal function and the test function.

Let $\pi: X' \rightarrow X$ be the mapping of the set of variables from $X' = \{x'_i\}$ to $X = \{x_i\}$ such that $x_i = \pi(x'_j), \forall i, j \in [0, n-1]$ and $S' = S(\pi(X'))$. Then the next state and output equations of M will be modified to:

$$X = \Delta'(S(\pi(X')), y) \cdot TE + \Delta(S, y) \cdot \overline{TE} \quad (3.3)$$

$$O = \Lambda'(S(\pi(X')), y) \cdot TE + \Lambda(S, y) \cdot \overline{TE} \quad (3.4)$$

where Δ and Δ' are the next state equations of X and X' of M and M' , respectively.

The complexity of the merged next state and output decoders can be reduced by keeping the state variable dependency low in state variable mapping, with the help of a dependency graph (DG). A DG is a directed graph with each vertex representing an

FF and each arc representing the signal flow from an FF's output to the same or a different FF's input by abstracting away the combinational logic between them. DG of the object machine M can be obtained by tracing the netlist, but DG of the test machine M' cannot be obtained before state assignment.

A two-block partition [114] on the state set S of M divides S into two disjoint subsets, i.e., $B_1 \cup B_2 = S$ and $B_1 \cap B_2 = \emptyset$; Each subset of states, $B_1, B_2 \subset S$, is called a block. Let $s_i \equiv s_j$ denote that states s_i and s_j are in the same block of partition. A partition for M is said to be closed if $\Delta(s_i, y) \equiv \Delta(s_j, y)$ for all $s_i \equiv s_j$ and $y \in I$, where $s_i, s_j \in S$.

To obtain a state assignment for M' , a two-block partition is induced by each state variable such that it is assigned the same value for all states in the same block of the partition. As there are n two-block partitions and each block of a partition can be assigned either a '0' or '1' value for its state variable, there are 2^n different state encodings. The dependency of state variables is minimized if the n two-block partitions of M' are closed [114].

Figure 3.2 Minimum dependency graph of 3-FF test machine.

Fig. 3.2 shows the DG with the least state-variable dependency for the 3-FF test machine with the following state assignment:

$$\text{FF1': } \{(S_0, S_2, S_4, S_6), (S_1, S_3, S_5, S_7)\},$$

$$\text{FF2': } \{(S_0, S_1, S_4, S_5), (S_2, S_3, S_6, S_7)\},$$

$$\text{FF3': } \{(S_0, S_1, S_2, S_3), (S_4, S_5, S_6, S_7)\}.$$

As an example, consider the insertion of the 3-FF test machine into the ISCAS'89 benchmark circuit S27 shown in Fig. 3.3, with its DG extracted in Fig. 3.4. If (FF1', FF2', FF3') of the test machine is mapped to (FF3, FF1, FF2) or (FF3, FF2, FF1) of S27, there is no increase in dependencies among the state variables.

Figure 3.3 Gate-level schematic of S27.

Figure 3.4 State variable dependency graph of S27.

Let $a = b = 0$ and $S'(x'_1 x'_2 x'_3)$ of M' be encoded as: $S_0(101)$, $S_1(100)$, $S_2(111)$, $S_3(110)$, $S_4(001)$, $S_5(000)$, $S_6(011)$ and $S_7(010)$. From Fig. 3.1, the next state and output equations of M' are given by: $x_1^{'+} = \bar{y}'$, $x_2^{'+} = \bar{x}_1'$, $x_3^{'+} = \bar{x}_2'$ and $z' = \bar{x}_3'$, where $x_i^{'+}$ denotes the next state value of x_i' .

From Fig. 3.3, the next state and output equations of M are given by: $x_1^{+} = \overline{\bar{G}_0 + f_1}$, $x_2^{+} = \bar{f}_1$, $x_3^{+} = \overline{\bar{G}_2 + f_2}$ and $OUT = f_1$, where $f_1 = x_1 + \overline{\bar{f}_3 \cdot f_4}$, $f_2 = \overline{\bar{G}_1 + x_3}$, $f_3 = G_3 + \bar{G}_0 \cdot x_2$ and $f_4 = \bar{G}_0 \cdot x_2 + f_2$. M has four input pins. If G_0 is arbitrarily selected to share with the input y' of M' and (x'_1, x'_2, x'_3) are mapped to (x_3, x_2, x_1) of M , then the next state and output equations after test insertion are given by:

$$x_3^{+} = \bar{G}_0 \cdot TE + \overline{\bar{G}_2 + f_2} \cdot \overline{TE} \quad (3.5)$$

$$x_2^{+} = \bar{x}_3 \cdot TE + \bar{f}_1 \cdot \overline{TE} \quad (3.6)$$

$$x_1^{+} = \bar{x}_2 \cdot TE + \overline{\bar{G}_0 + f_1} \cdot \overline{TE} \quad (3.7)$$

$$OUT = \bar{x}_1 \cdot TE + f_1 \cdot \overline{TE} \quad (3.8)$$

The circuit S27 after test insertion is shown in Fig.3.5.

Figure 3.5 Schematic of S27 after test machine insertion.

Table 3.2 shows the synthesis results of S27 after it has been embedded with each of the 2^3 different encodings of M' and re-synthesized by Synopsys Design Compiler (DC) using TSMC 0.18 μm technology. Different state encodings of the test machine results in different instances of varying implementation cost. The largest instance is 4.7% larger than the smallest one, the slowest instance is 11.5% slower than the fastest one, and the most power hungry instance consumes 5.1% more power than the least power hungry one.

Table 3.2

Synthesis results for S27 after test machine insertion.

Design	Test machine encoding	Area (μm^2)	Delay (ns)	Power (mW)
1	S_0 (000), S_1 (001), S_2 (010), S_3 (011) S_4 (100), S_5 (101), S_6 (110), S_7 (111)	292.7	0.81	0.267
2	S_0 (100), S_1 (101), S_2 (110), S_3 (111) S_4 (000), S_5 (001), S_6 (010), S_7 (011)	296.1	0.87	0.256
3	S_0 (010), S_1 (011), S_2 (000), S_3 (001) S_4 (110), S_5 (111), S_6 (100), S_7 (101)	286.1	0.81	0.254
4	S_0 (110), S_1 (111), S_2 (100), S_3 (101) S_4 (010), S_5 (011), S_6 (000), S_7 (001)	289.4	0.82	0.264
5	S_0 (001), S_1 (000), S_2 (011), S_3 (010) S_4 (101), S_5 (100), S_6 (111), S_7 (110)	292.7	0.82	0.266
6	S_0 (101), S_1 (100), S_2 (111), S_3 (110) S_4 (001), S_5 (000), S_6 (011), S_7 (010)	282.7	0.81	0.255
7	S_0 (011), S_1 (010), S_2 (001), S_3 (000) S_4 (111), S_5 (110), S_6 (101), S_7 (100)	289.4	0.78	0.261
8	S_0 (111), S_1 (110), S_2 (101), S_3 (100) S_4 (011), S_5 (010), S_6 (001), S_7 (000)	292.7	0.79	0.265

3.3 THE PROPOSED FINGERPRINTING SCHEME

One unique challenge of IP fingerprinting over IP watermarking is the design effort required to derive a large number of distinct high-quality solutions of the same functionality for different buyers. Another challenge is the ease of recovering the signature of the embedded IP core off-chip without compromising its security against erasure and collusion attacks. Last but not least is the difficulty to lower the area and timing overheads of embedding both the authorship proof and IP buyer's identification in the same design instance. The latter can be addressed by merging the watermark and fingerprint into one single signature but an associated problem is on keeping this signature unobtrusive without losing the non-repudiation of the authorship proof and origin of misappropriation. This section presents a fingerprinting method based on the test machine insertion problem. For ease of exposition, the original sequential circuit is referred to as *object design* M and the sequential circuit after the test machine insertion as *testable design* M_T . The testable design with the best synthesis result in terms of area or timing is chosen as the *seed design* M_S for fingerprinting.

3.3.1 Fingerprint Generation

Instead of routinely generating the watermark and fingerprint as two independent signatures, the proposed scheme uses a unitary signature to provide an undeniable proof of IP ownership and for traceability of illegal IP distribution. This signature is generated through a blind signature protocol [116] to preserve the anonymity of the IP authorship information endorsed by the buyer. The blind signature protocol requires a digital signature mechanism and a commuting function. For simplicity, Chaum's blind signature protocol based on RSA is considered. Let (n_A, e_A) and d_A be the certified RSA public and private keys of Buyer A , where $n_A = p_A \cdot q_A$ is the product of two large random primes. The fingerprint F to be embedded into the IP instance for Buyer A can be generated through the following message exchange between the IP provider and Buyer A .

1. *Watermark generation*: The IP provider selects a message to convey its ownership information. The ASCII encoded message is first converted into a binary string and

then processed by a hash function such as SHA-2 [117] to generate an integer W , where $0 \leq W < n_A$.

2. *Blinding phase*: When Buyer A makes a purchase request, the IP provider randomly selects a secret integer k_A satisfying $0 \leq k_A < n_A$ and $\gcd(n_A, k_A) = 1$ to compute $W_A = W \cdot (k_A)^{e_A} \bmod n_A$. This concealed watermark W_A is then sent to Buyer A .
3. *Signing phase*: To endorse the purchase, Buyer A signs W_A with his secret key d_A and returns his blinded signature $F_A = (W_A)^{d_A} \bmod n_A$ to the IP provider.
4. *Un-blinding phase*: By computing $F = F_A \cdot k_A^{-1} \bmod n_A$, the IP provider obtains the signature of Buyer A on his watermark W , which is the fingerprint to be inserted into the IP instance sold to Buyer A .

The IP provider can verify if the fingerprint F is genuine by decrypting it with Buyer A 's public key. Since F is the digital signature of Buyer A on the IP provider's watermark W , it provides an undeniable proof for tracing the redistribution of the copies of IP bought by him. Meantime, as W is concealed in W_A , Buyer A has no knowledge of the IP provider's watermark W and his own signature on W . The blindness of F increases the deterrent effect of uncertainty.

3.3.2 Fingerprint Insertion

By applying automatic test pattern generation on the seed design, a set of combinational test vectors can be obtained. Each test vector is then converted to a test sequence which includes an SS to excite a testable design to a specific state S_s at test mode (i.e., $TE = '1'$) and a parallel input stimulus to produce the output of a transition from S_s to some destination state S_d at capture mode (i.e., $TE = '0'$). Irrespective of the input stimulus, the destination state S_d from any starting state S_s can always be identified by its DR based on the test machine property. As demonstrated in Table 3.1 for $n = 3$, the DR of any destination state can be observed from the primary output z by applying an arbitrary input sequence of length n through the primary input y at test mode.

To insert the fingerprint $F = \{f_i\}_{i=0}^{m-1}$ into an object design, a test vector V for the seed design M_S is randomly selected. Let S_s and S_d be the initial and destination states of M_S for the test vector V . To randomly modulate m out of n ($m \ll n$) binary bits of $DR(S_d)$ by F , a keyed one-way pseudorandom number generator (PNG) [118] is used to generate an order set $L = \{l_i\}_{i=0}^{m-1}$, of m unique integers between 0 and $n-1$ such that $l_i \in [0, n-1]$, $\forall i = 0, \dots, m-1$ and $l_i \neq l_j, \forall i \neq j$. l_i corresponds to the location of f_i , where f_0 is the LSB of F . The keyed one-way PNG can be realized by SHA-2 or any other cryptographically secure hash function, as long as it is computationally infeasible to find a collision of the same group of numbers without the knowledge of the secret key.

According to L , a fingerprint-compatible $DR^* = \{r_j\}_{j=0}^{n-1}$ is generated by setting $r_j = f_i$ if $j = l_i$ for all $i = 0, \dots, m-1$ and $r_j = '-'$ otherwise, where $'-'$ denotes a don't-care value.

A binary ($\{0, 1\}$) sequence A is said to be compatible to a ternary ($\{0, 1, -\}$) sequence B of the same length, denoted as $A \subset B$, if all but the don't care values of A and B in the same bit positions are matched, i.e., $a_i = b_i, \forall a_i \in A, b_i \in B$ and $b_i \neq '-'$. If $DR(S_d)$ is compatible with DR^* , M_S will be used as the fingerprinted instance M^* . Otherwise, a subset $S^* = \{S_u \in S \mid DR(S_u) \subset DR^*\}$ is extracted from the state set S of M_S such that the DRs of all its members are compatible with DR^* . The fingerprinted instance can then be generated by modifying the state encoding of M_S by $\phi: x_i \rightarrow \dot{x}_i, \forall i = 1, \dots, n$, where $\dot{x}_i = x_i$ or $\bar{x}_i$, such that the encoded value of one of its states $S_d^* \in S^*$ is equal to the encoded value of S_d , i.e., $S_d^* = \phi(S_d)$. This re-encoding of state variables can be performed through the embedded test machine of M_S . If there are more than one state encodings that satisfy this requirement, the testable design instance with the least overhead is selected as the fingerprinted design M^* . A fingerprint verification code V^* can then be derived from V by replacing its $SS(S_s)$ by $SS(S_s^*)$, where $S_s^* = \phi(S_s)$. The fingerprint insertion process is depicted in Fig. 3.6.

```

1: fingerprint( $M_s, F, L, V$ )
2: {
3:    $M_s$ : Seed design of  $n$  FFs;
4:    $F$ : Fingerprint to be inserted, binary vector of length  $m$ ;
5:    $L$ : List of  $m$  location indices;
6:    $V$ : Randomly selected combinational test vector;
7:    $r$ : Ternary vector of length  $n$ 
8:   for ( $j = 0; j < n; j++$ ) {
9:      $r[j] = '-'$ ;
10:  for ( $i = 0; i < m; i++$ )
11:    if ( $j == L[i]$ ) {
12:       $r[j] = F[i]$ ;
13:      break;
14:    }
15:  } // end for loop to obtain  $r$ 
16:  if ( $DR(S_d) \subset r$ ) return ( $M_s, V$ ); // if DR is compatible with  $r$ 
17:   $CS = \emptyset$ ; // create an empty list of DR-compatible states
18:  for ( $i = 0; i < 2^n; i++$ ) // search for DR-compatible states
19:    if ( $DR(S_i) \subset r$ )  $CS += S_i$ ;
19:   $min\_overhead = MAX\_VALUE$ ; // initialize minimum overhead
20:  for each ( $2^n$  state encoding  $\phi$  of  $M$ ) {
21:    if ( $\phi(S_d) \in CS$ ) {
22:       $fingerprint\_M = resynthesize(M_s, \phi)$ ;
23:       $overhead = overhead(fingerprint\_M, criteria)$ 
24:      if ( $overhead < min\_overhead$ ) {
25:         $min\_overhead = overhead$ ;
26:         $opt\_M = fingerprint\_M$ ;
27:         $opt\_V = modified\_SS(V, \phi(S_s))$ ; // modify SS of  $V$ 
28:      }
29:    }
30:  } // end search for optimal fingerprinted instance
31:  return ( $opt\_M, opt\_V$ );
32: }
```

Figure 3.6 Fingerprint insertion algorithm.

As an example, consider the insertion of a one-bit signature $f_0 = '1'$ into the 3-FF testable design of S27. If design area is the prime concern, the smallest Design 6 in Table 3.2 is selected as the seed design M_s . Let $V = "110;0100;---"$ be a randomly selected test vector. The first 3-bit vector is $SS(S_3)$, the 4-bit vector is applied to the primary inputs " $G_0G_1G_2G_3$ " at capture mode to bring $S_s = S_3$ to $S_d = S_4$ and the 3-bit don't care vector is the DS to obtain $DR(S_4)$. For $a = b = 0$, $DR(S_4) = "001"$ from Table 3.1. If $L = \{1\}$, then $DR^* = "-1-"$. Since $DR(S_4)$ is not compatible with DR^* , a search for all states S_i with $DR(S_i) \subset DR^*$ is performed. From Table 3.1, the compatible states are $S^* = \{S_2, S_3, S_6, S_7\}$. From Table 3.2, it can be observed that the encoded values of S_2 in Design 7, S_3 in Design 3, S_6 in Design 8 and S_7 in Design 4 are mapped to the encoded value of $S_4 = "001"$ of M_s . Among them, Design 3 has the

least area and is selected as the fingerprinted design M^* , with $\phi: x_i \rightarrow \bar{x}_i$, $S_s^* = \phi(S_3) = S_4$ and $S_d^* = \phi(S_4) = S_3$. From Table 3.1, $SS(S_s^*) = "001"$ and $DR(S_d^*) = "110"$. Thus, $V = "001;0100;- -"$. By applying V^* to M^* , $f_0 = '1'$ can be detected from the second bit (i.e., $l_0 = 1$) of DR^* .

3.3.3 Partitioning for Efficient Fingerprinting

It is difficult to obtain an optimized seed design for fingerprinting by resynthesis when the number of FFs n is large. The resynthesis can be made more efficient by partitioning the object design into smaller interconnected segments. If each segment contains only c FFs, it becomes affordable to synthesize all the 2^c testable design instances to obtain an optimized seed design for each segment if $c \ll n$. The total number of resynthesis processes is reduced to $\lfloor n/c \rfloor \cdot 2^c + |n|_c \cdot 2^{|n|_c}$, where $\lfloor \cdot \rfloor$ is the floor function and $|a|_b$ denotes the remainder of a divided by b , and only a very small part of the design is re-synthesized in each process.

As the DG of the test machine has a chain-like structure as shown in Fig. 3.2, multiple c -FF test machines can be easily cascaded to form a linear chain of state variables. To minimize the dependencies added in the DGs of the partitioned design, the longest chain removal technique [112] is used to chain the state variables of the object design before it is partitioned into smaller interconnected segments. The result is an ordered list of FFs that can be used to direct the partitioning of an n -FF object design M . The following procedure outlines the *state variable chaining heuristic*.

- 1) Extract the DG of M by omitting the self-dependencies of the FFs.
- 2) Compute strongly connected components (SCCs)² of the DG and replace each SCC in the DG with a single vertex to form a directed acyclic graph (DAG).
- 3) Obtain the longest chain of the DAG.
- 4) Obtain the longest chain in each SCC.
- 5) Replace the vertices representing the SCCs in the chain obtained in Step (3) with the chains obtained in Step (4) to form the *longest chain* of the DG.

² A SCC of a directed graph $G(V, E)$ is a maximal set of vertices $C \subseteq V$ such that for every pair of vertices $\{u, v\} \in C$, there is a directed path from u to v and a directed path from v to u .

- 6) Remove the vertices in the current longest chain and their associated edges from the DG. Repeat Steps (1) to (5) for the new DG. The newly extracted longest chain obtained in Step (5) is then prefixed to the existing longest chain.
- 7) Repeat Step (6) until there is no vertex left in the DG.

To find the longest chain for a directed graph containing loops is an *NP*-complete problem [119]. Therefore, loops are removed in Step (2) to simplify the DG to DAG so that dynamic programming [120] with linear computational complexity can be used to extract the longest chain from the DAG. Finding the longest chain in a SCC is also an *NP*-complete problem. This problem in Step (4) is simplified by determining the longest chain from the spanning tree of SCC. The DG is reduced by pruning its longest chain in Step (6). The process repeats until no vertex is left in the DG. A linear chain of state variables for the object design is obtained with dependencies added merely for the connections from the tail of a newly extracted longest chain to the head of an existing chain between two iterations.

The algorithm for the partitioning of the object design into *c*-FF segments is shown in Fig. 3.7. The output list of each segment after the partitioning is recorded. The output list of segment *i* contains the outputs of FFs or gates in segment *i* that are either primary outputs or inputs to FFs or gates in other segments.

```

1: dependency_directed_partitioning( $M, FF, c$ )
2: {
3:      $M$ : Sequential design to be partitioned;
4:      $FF$ : List of FFs of  $M$  ordered according to state variable chain;
5:      $Gate$ : List of gates of  $M$ ;
6:      $IO$ : List of primary inputs and primary outputs of  $M$ ;
7:      $c$ : Predefined maximum number of FFs in each partition;

8:      $Gate[i].used = 0$ ;  $FF[i].used = 0$ ; // Initialize all gates and FFs of  $M$ 
9:      $j = 0$ ;
10:    while ( $FF$  is not empty) {
11:         $Partition[j] = \text{create\_partition}()$ ; //create a new partition
12:        if (number of unused FFs in  $FF > c$ )
13:             $Partition[j].FF +=$  last  $c$  unused FFs of  $FF$ ;
14:        else
15:             $Partition[j].FF +=$  all unused FFs of  $FF$ ;
16:        mark\_used\_FFs(); // mark those FFs added to  $Partition$  as used;
17:        for each ( $Partition[j].element$ ) { //  $element$  can be  $Gate$  or  $FF$ 
18:             $Predecessor = \text{get\_predecessor}(Partition[j].element)$ ;
19:            if ( $Predecessor == FF[k]$  and  $FF[k] \notin Partition[j].FF$ )
20:                 $Partition[j].input += FF[k].output$ ;
21:            else if ( $Predecessor == Gate[k]$ )
22:                if ( $Gate[k].used == 0$ ) { //  $Gate[k]$  is unused
23:                     $Partition[j].gate += Gate[k]$ ;
24:                     $Gate[k].used = 1$ ;
25:                } else //  $Gate[k]$  is used
26:                    if ( $Gate[k] \notin Partition[j].Gate$ )
27:                         $Partition[j].input += Gate[k].output$ ;
28:                else  $Partition[j].input += IO[k]$ ; //  $Predecessor$  is  $IO[k]$ 
29:            } //end for loop
30:             $j++$ ;
31:        } //end while loop
32:    return  $Partition$ ;
32: }
```

Figure 3.7 Dependency-directed partitioning algorithm.

The object design M is partitioned into t smaller interconnected segments, $M_0, M_1, \dots, M_{t-1}$, each with n_i FFs, where $i \in [0, t-1]$ and M_0 denotes the segment that is nearest to the output z . Then an n_i -FF test machine is inserted into each M_i independently as described in Section 3.2 with 2^{n_i} possible state encodings. These 2^{n_i} different testable designs for M_i are re-synthesized and the most optimized testable design is selected as the seed design M_{Si} for M_i .

To disperse the fingerprint $F = \{f_j\}_{j=0}^{m-1}$ with $L = \{l_i\}_{i=0}^{m-1}$ location indices into the partitioned seed design, the fingerprint-compatible DR^* is divided into t ternary strings, $DR_i^* = \{r_{i,j}\}_{j=0}^{n_i-1}$ for $i = 0, \dots, t-1$ and $r_{i,j} \in \{0, 1, '-'\}$. DR_i^* can be

generated by setting $r_{i,j} = f_k$ if $\sum_{b=0}^{i-1} n_b + j = l_k$, $\forall k = 0, \dots, m-1$ and $r_{i,j} = '1'$ otherwise. If all bits of DR^* are don't cares, no fingerprint bit needs to be inserted into M_{Si} and the fingerprinted instance $M_i^* \equiv M_{Si}$. Otherwise, the fingerprinting algorithm in Fig. 3.6 is used to insert the fingerprint bits in DR^* to produce the fingerprinted instance M_i^* . All fingerprinted instances M_i^* are then cascaded back together and re-synthesized to produce the final fingerprinted design M^* .

As an example, consider the ISCAS benchmark S344. Its DG after removing the self-dependencies is shown in Fig. 3.8, where the FF dependencies are represented by solid directed edges. Using the state variable chaining heuristic, the existing longest FF chain $FF1 \rightarrow FF2 \rightarrow FF3 \rightarrow FF15 \rightarrow FF4 \rightarrow FF5 \rightarrow FF6 \rightarrow FF7 \rightarrow FF8 \rightarrow FF9 \rightarrow FF10 \rightarrow FF11$ is first extracted. After removing the FFs in the longest chain from the DG, $FF14$, $FF13$ and $FF12$ are individually extracted as the longest chains in the subsequent iterations. By appending the existing chain to the newly extracted longest chains, the final chain of FFs is given by $FF12 \rightarrow FF13 \rightarrow FF14 \rightarrow FF1 \rightarrow FF2 \rightarrow FF3 \rightarrow FF15 \rightarrow FF4 \rightarrow FF5 \rightarrow FF6 \rightarrow FF7 \rightarrow FF8 \rightarrow FF9 \rightarrow FF10 \rightarrow FF11$. As shown in Fig. 3.8, the selected existing dependencies are indicated with red solid directed edges, while new dependencies that are added to form this FF chain are indicated by red dashed directed edges.

Figure 3.8 Dependency graph of S344 excluding self-dependencies.

Assume the maximum number of FFs per segment is 3, the algorithm in Fig. 3.7 returns the following partitions:

$$\begin{aligned}
 M_0 & \text{ with } \{x_3(FF11), x_2(FF10), x_1(FF9)\}, \\
 M_1 & \text{ with } \{x_3(FF8), x_2(FF7), x_1(FF6)\}, \\
 M_2 & \text{ with } \{x_3(FF5), x_2(FF4), x_1(FF15)\}, \\
 M_3 & \text{ with } \{x_3(FF3), x_2(FF2), x_1(FF1)\}, \\
 M_4 & \text{ with } \{x_3(FF14), x_2(FF13), x_1(FF12)\}.
 \end{aligned}$$

Table 3.3 shows the synthesized areas for each partitioned testable design M_{Ti} . Designs 1 to 8 correspond to the 8 different state encodings of the embedded 3-FF test machine in the second column of Table 3.2. The embedding is performed by mapping (x'_1, x'_2, x'_3) of the test machine to (x_1, x_2, x_3) of each segment M_i . The best designs in terms of area, i.e., Design 6 for M_{S0} , Design 3 for M_{S1} , Design 5 for M_{S2} , Design 1 for M_{S3} and M_{S4} , are selected as the seed designs for fingerprinting.

Table 3.3

Synthesis results (area in μm^2) for test machine embedded components of S344.

Design	M_{T0}	M_{T1}	M_{T2}	M_{T3}	M_{T4}
1	439.1	575.5	459.0	319.3	282.7
2	415.8	595.4	485.7	349.3	316.0
3	439.1	568.8	452.4	319.3	292.7
4	415.8	602.1	479.0	349.3	312.7
5	439.1	575.5	445.7	319.3	282.7
6	409.1	595.4	472.3	342.6	316.0
7	439.1	568.8	452.4	319.3	292.7
8	415.8	602.1	479.0	349.3	312.7

Assume a fingerprint $F = "110001"$ and its location indices $L = \{5, 11, 9, 2, 4, 0\}$. Then $DR^* = "---1-0---10-0-1"$ is the fingerprint compatible DR. It can be partitioned into $DR_0^* = "0-1"$, $DR_1^* = "10-"$, $DR_2^* = "- - -"$, $DR_3^* = "1-0"$ and $DR_4^* = "- - -"$. Since $DR_2^* = DR_4^* = "- - -"$, M_{S2} and M_{S4} can be used as fingerprinted instances M_2^* and M_4^* , respectively.

Let $V = "011110001011001; 001010111; - - - - -"$ be the test vector randomly selected for interconnected seed designs $M_{S4} \rightarrow M_{S3} \rightarrow M_{S2} \rightarrow M_{S1} \rightarrow M_{S0}$ with $S_s = "S_6-S_3-S_4-S_6-S_4"$, $S_d = "S_7-S_2-S_7-S_7-S_4"$ and $DR(S_d) =$

"111010111111001". The underlined bits of $DR(S_d)$ are incompatible with those of DR^* . Since $DR_0 = "001" \subset "0-1"$, M_{S0} can be used as the fingerprinted instance M_0^* . Hence, all together, the seed design M_{S0} , M_{S2} and M_{S4} can be directly employed as the fingerprinted instance for segment 0, 2 and 4. As $DR_1 = "111" \not\subset "10-"$ and $DR_3 = "010" \not\subset "1-0"$, the seed designs for segments 1 and 3 cannot be used as the fingerprinted instances and need to be updated.

First, consider the fingerprinting of segment 1, where DR_1 upon the application of the test vector need to be compatible with "10-". From Table 3.1, for $a = b = 0$, $DR(S_1) = "100"$ and $DR(S_5) = "101"$ satisfy this requirement. The new destination state S_d^* must also be encoded with the same value of $S_d = S_7 = "101"$ of M_{S1} . From Table 3.2, the encoded value of S_5 in Design 1 and S_1 in Design 2 satisfies this requirement. In Table 3.3, Design 1 has smaller area than Design 2 of M_{T1} , and is selected as M_1^* ; then $\phi: (x_1, x_2, x_3) \rightarrow (x_1, \bar{x}_2, x_3)$, and $S_s^* = \phi(S_6) = S_4$ and $S_d^* = \phi(S_7) = S_5$. Finally, the SS and DR of segment 1 are updated. From Table 3.1, $SS(S_4) = "001"$ and $DR(S_5) = "101"$.

Similarly, for segment 3, $DR_3 = "010" \not\subset "1-0"$. From Table 3.1, $DR(S_1) = "100"$ and $DR(S_3) = "110"$ are compatible with "1-0". The encoded value of S_1 in Design 7 and S_3 in Design 5 is the same as the encoded value of $S_d = S_2 = "010"$ of M_{S3} . Design 5 and Design 7 of M_{T3} have the same cost and either of them can be selected as the fingerprinted instance M_3^* . If Design 7 is selected, then $\phi: (x_1, x_2, x_3) \rightarrow (x_1, \bar{x}_2, \bar{x}_3)$, and $S_s^* = \phi(S_3) = S_0$ and $S_d^* = \phi(S_2) = S_1$. From Table 3.1, $SS(S_0) = "000"$ and $DR(S_1) = "100"$.

Fingerprinted instances are chained in the form of $M_{S4} \rightarrow M_3^* \rightarrow M_{S2} \rightarrow M_1^* \rightarrow M_{S0}$ and re-synthesized to produce the fingerprinted design M^* . The resultant V and DR is also updated, where $V^* = "011000001001001; 001010111; -----"$ and $DR^* = "111100111101001"$. The underlined bits are the modified ones due to the update of M_1^* and M_3^* . The area, delay and power consumption of the seed and final fingerprinted instances are ($1616.6 \mu\text{m}^2$, 1.19 ns, 1.13 mW) and ($1623.3 \mu\text{m}^2$, 1.18 ns, 1.18 mW), respectively.

3.3.4 Fingerprint Detection and Authorship Verification

The IP provider keeps a safe repository of records. Each record consists of the fingerprint F , the secret integer k used in the blinding phase, the buyer's public key (n, e) , the fingerprint location vector L and the test vectors V^* for each fingerprinted instance. To detect the authorship of a fingerprinted design, the IP provider can apply the test vector V^* to obtain a DR, from which the fingerprint F can be recovered at the bit locations specified by L . The buyer's public key (n, e) corresponding to F can then be retrieved from the database. The IP provider's watermark can be detected by $W = F^e \bmod n$. The IP provider can prove his ownership of the IP by demonstrating that his legible ownership message can be hashed to W . Since the IP provider's watermark W can be successfully recovered by decrypting the fingerprint F with the specific IP buyer's public key, it proves that the fingerprinted design is sold to that buyer. Thus the buyer of a fingerprinted design can be tracked and held responsible if the fingerprinted design was found to be illegally copied and redistributed.

Decrypting F to detect W needs only to be done once but to detect F from a fingerprinted instance of unknown buyer may require the application of all the different test vectors V^* . To aid the detection of F , the IP provider can store the responses DR_a to a common test vector V_a for all fingerprinted instances into the corresponding entries of the database. With high probability, DR_a for the same test vector V_a will be different for different instances. By applying V_a to the instance, the number of test vectors V^* required to detect F will be reduced dramatically to only those few matching records of DR_a .

3.3.5 Augmented Fingerprint

To increase the attackers' difficulty to successfully erase the fingerprint F , error correction code (ECC) can be added to it. More fingerprint bits will have to be modified in order to prevent F from being correctly detected by the IP provider. Owing to the randomized dispersion by L , the ECC bits are automatically interleaved with the fingerprint bits in DR. If necessary, F can be compressed by cryptographic hash function to adapt to the embedding capacity of the design. The IP provider needs only to demonstrate additionally that F can be hashed to the embedded bit stream recovered from the sold design.

3.4 SECURITY ANALYSIS OF THE PROPOSED SCHEME

No hardware IP protection scheme is complete without an analysis of its *credibility*, *robustness* and *feasibility*. Credibility refers to the strength of ownership proof, robustness refers to the resistance against known and perceivable attacks, and feasibility refers to the effort in finding the number of quality solutions required for the protection scheme. These attributes are analyzed below.

3.4.1 Fingerprint Credibility

A common measure to quantify and compare the credibility or strength of a fingerprinted solution is the probability of coincidence (P_c), which are the odds that a non-fingerprinted design carries the fingerprint by coincidence. A lower P_c implies a stronger proof of ownership and original recipient. In the proposed scheme, since each bit of DR of a design is equally probable to be '0' or '1', the probability that m randomly selected bits in DR match a specific binary string of fingerprint is given by:

$$P_c = \frac{1}{2^m} \quad (3.9)$$

With a 64-bit fingerprint, the P_c of a fingerprinted instance is as low as 5.4×10^{-20} . This probability reduces to the order of 10^{-40} for a 128-bit fingerprint.

3.4.2 Fingerprint Robustness

The following attack scenarios are analyzed with Alice as the IP provider using the proposed fingerprinting scheme to protect her IP core and Bob as a malefactor attempting to steal her IP.

3.4.2.1 Collusion attack

This attack requires a coalition of multiple copies of fingerprinted instances in possession by one or more malicious users in order to create a forged copy without the fingerprint for redistribution. Bob may collude with other buyers to compare multiple protected instances to find out their structural dissimilarities, with the hope that these dissimilar parts of the circuits can be modified at an acceptable cost and effort to remove the fingerprint without losing the usefulness and correct functionality

of the IP. To succeed in such an attack, the fingerprint must be altered to an extent that none of the users in the coalition can be identified by Alice.

While this attack is a biggest threat to data hiding algorithms on multimedia content in which the hidden signature (watermark or fingerprint) does not depend on the host data, several features in the proposed fingerprinted instance make such attack ineffective. First, the fingerprint bits randomly modulate the values of state variables that affect the next state and output decoding logic in both the normal and test functions, making them inextricable from the functional components of the IP. This makes it very challenging for Bob to figure out the functionality of any structural mismatch in order to successfully remove even a single fingerprint bit. Secondly, the same segment M_i^* of different fingerprinted instances can be identical (i.e., both equal to M_{Si}) or different irrespective of the value and number of fingerprint bits embedded in them. Lastly and most importantly, the entire chain of testable machines is re-synthesized as a whole to obtain M^* . The domino effect of the hard optimization problem involved in the resynthesis process whitens the differences between the fingerprint-modulated and un-modulated sub-circuits, causing the entropy of identifying and associating the matching or mismatching sub-circuits between any two instances to increase with the number of fingerprinted instances being compared.

3.4.2.2 *Combinational logic resynthesis*

Bob may attempt to remove the fingerprint by performing a combinational logic resynthesis through various approaches [121]. Although this attack can successfully modify the circuit structure by changing the combinational logic implementations without affecting the functionality, the input and output behaviors of the sequential elements are preserved. Thus, Alice can still recover the fingerprint F from the DR of the fingerprinted design modified by Bob. One characteristic of the test machine is that it can be decomposed into multiple smaller test machines and independently embedded into a correspondingly partitioned large machine. Although the original circuit topologies of these interconnected machines may not be maintained upon global optimization of the final testable design, the full controllability and observability of all FFs are not affected. As an example, consider the STGs of one-FF and two-FF test machines shown in Fig. 3.9. If they are interconnected such that the

output of the first test machine is the input of the second test machine, then the $2^1 \times 2^2 = 8$ combined states can be mapped to the 8 states for the 3-FF test machine of Fig. 3.1. Let the state of the final test machine be represented as $(S_{1,i}, S_{2,j})$ when the first test machine is in state $S_{1,i}$ and the second test machine is in state $S_{2,j}$. If the bijection maps $(S_{1,i}, S_{2,j})$ to S_k of the 3-FF test machine, then $SS((S_{1,i}, S_{2,j}))$ and $DR((S_{1,i}, S_{2,j}))$ of the combined machine can still be determined from $SS(S_k)$ and $DR(S_k)$, respectively in Table 3.1 for the 3-FF test machine. The proposed method utilizes this property to efficiently distribute the fingerprint bits and propagate the embedded fingerprint F to all subsequent stages of the design flow. The fact that $SS(S_k)$ and $DR(S_k)$ remain intact after the resynthesis of the final testable design implies that F will survive all forms of logic synthesis and optimization performed on the fingerprinted instance.

Figure 3.9 STGs of (a) 1-FF and (b) 2-FF test machines.

3.4.2.3 Circuit retiming

Circuit retiming relocates the FFs of sequential circuit to improve its performance while preserving the functionality. Bob may retime the fingerprinted instance. As retiming may change the STG of the circuit, it seems possible for Bob to partially remove the fingerprint inserted by Alice if the altered parts of the STG host the fingerprint bits.

According to [122], retiming merges two states that are one-step equivalent or splits one existing state into two new states that are one-step equivalent. Two states are said to be one-step equivalent if they have the same output and the same destination state for any input. Fortunately, each state of the test machine is distinct and the fingerprinted instance contains no one-step equivalent state. Hence, none of the states in the fingerprinted instance can be merged by circuit retiming. Meantime, as the test

machine is fully specified, it is also not possible to split one state of the fingerprinted instance into two one-step equivalent states without adding extra sequential elements. Even if Bob manages to create one-step equivalent states at the cost of additional circuitry without losing the usefulness of Alice's IP, the states and their associated transitions involved in the fingerprint extraction are retained. Thus Alice can still recover the correct fingerprint from the DR of the retimed fingerprinted instance redistributed by Bob.

3.4.2.4 State re-encoding

There are two possible ways by which Bob can recode the state variables of Alice's protected design. The first way is to extract the STG directly from the fingerprinted design to carry out the state re-encoding. It has been proven that extracting the STG from a large sequential circuit after logic synthesis is computationally intractable [39]. Thus this approach is infeasible in practice. Alternatively, Bob may perform a partitioning on the fingerprinted instance to obtain many smaller segments of the design. With no knowledge of the design or architecture of Alice's IP, Bob may only be able to extract a limited number of STGs for some segments to recode their state variables. Since the buyer has no knowledge of the embedded fingerprint F and input vector V^* for its recovery, Bob can only select the segments by wild guess. Due to the existence of loops (SCCs) in the dependency graph, the solution for the chaining of state variables with minimum additional dependencies is not unique. Various new SCCs have been formed after the embedding of test machines, and the final re-synthesis of the entire fingerprinted design further masks the partitioning and chaining of state variables. Hence, it is highly unlikely that Bob could precisely relate the fingerprint bits to the corresponding state variables. It is reasonable to assume that Bob is incapable of extracting the STG for every partition, recode and re-synthesize the STG to forge a completely new design as this task incurs no less effort than the complete redesign of the IP functionality with a high, and almost certain risk of violating the circuit functionality or performance. Even if he succeeds in forging a new design, he will not be able to generate the proper test vectors for the counterfeit design without knowing the encodings used by the test machines, which will devalue his design.

3.4.2.5 State reduction

IP protection schemes, such as [39], that leverage on redundancy to mark the states to display some specific property for authorship proof are susceptible to this kind of attack. This vulnerability does not exist in the proposed fingerprinting scheme because the embedded test machine is a fully specified machine with distinct states. The final fingerprinted design contains no redundant state as, if any, it would have been reduced upon the re-synthesis performed after fingerprint insertion. In addition, state reduction is usually performed on explicit STG [123], which implies a very high cost of attack on a fingerprinted design instance released at the gate level or lower level of design abstraction.

3.4.3 Fingerprint Feasibility

Generally, the feasibility of a fingerprinting scheme is evaluated by three main criteria. First, the solution space required for the generation of a large number of unique instances. Second, the additional cost and effort required to produce a fingerprinted instance. Preferably, the fingerprinting method is compatible with existing design tools and no new investment of tool is needed to embed or detect the fingerprint. The additional effort required should also be substantially lower than designing the IP from scratch. Finally, the impact on quality degradation of any fingerprinted instance should be minimized and kept within a tolerable limit.

The first criterion can be evaluated by the number of possible solutions to fingerprint an IP. The number of combinations to select m state variables from a design of n FFs to host the m fingerprint bits is given by:

$${}^nC_m = \frac{n!}{m!(n-m)!} \quad (3.10)$$

As each state variable can be assigned either a value of '0' or '1', the number of different fingerprinted instances that can be generated by the proposed scheme is:

$$2^m \cdot {}^nC_m = \frac{2^m \cdot n!}{m!(n-m)!} \quad (3.11)$$

For $n = 100$ and $m = 32$, the solution space has exceeded 6×10^{35} .

For the second criterion, the fingerprint embedding process of the proposed scheme is completely transparent and can be used with existing design tools and standard cell libraries for ASIC implementation. As no specialized scan FF is required, the scheme is also applicable to fingerprint IPs for different configuration technologies of FPGAs, including the antifuse or flash-memory based FPGAs for which [52] may not be applicable. The additional design effort to generate each fingerprinted design instance is marginal after the seed design is obtained. Only those segments that are not compatible with its corresponding fingerprinted sequence DR^* at the fingerprint locations specified by L need to be remapped and a final re-synthesis is needed to globally optimize the entire fingerprinted design after the segments have been re-stitched.

Lastly, the proposed fingerprinting scheme incurs insignificant overhead on the design quality. This is attributed to the efficient dependency-directed partitioning algorithm and the utilization of seed designs, which are optimized for each partitioned testable machine. The low fingerprinting overhead will be demonstrated by experiments on different benchmark circuits in the next section.

3.5 EXPERIMENT RESULTS

Sequential circuits from ISCAS'89 benchmark suite are used to analyze the quality of the fingerprinted instances of the proposed scheme. Depending on the circuit size, the first 16, 32, 64, 128 or 256 bits of the keyed SHA-2 hash value of the fingerprint F were inserted. The location integers L were generated with a SHA-2 based PNG. The design partitioning, test machine embedding as well as the fingerprint insertion algorithms were coded in C++. Combinational test generation on the seed design was carried out with DfT tool DFTadvisor and ATPG tool Fastscan from Mentor Graphics. As part of the fingerprint insertion algorithm, the selected combinational test pattern is converted into a sequential test pattern of the form {SS; test vector; DR}. The synthesis was performed using Synopsis DC with TSMC 0.18 μm technology library, with C-Shell scripts written to automate the synthesis processes for the encoding instances of each segment as well as the whole design.

The synthesis results are shown in Table 3.4. " $\#FF$ " denotes the number of FFs in each benchmark circuit. " A_{opt} " and " A_m ", " D_s " and " D_m ", and " P_s " and " P_m " denote the

area in μm^2 , delay in ns and power in mW of the optimized and marked designs and " D_{opt} " and " D_m " the delay of the non-fingerprinted optimized seed design and fingerprinted design, respectively. ΔA , ΔD and ΔP denote the respective percentage increases. The area and delay were simulated by Synopsys DC, and the power by Synopsys PrimeTime PX with back annotation. The area, delay and power overheads due to fingerprint insertion are on average 1.6%, 1.2% and 3.4% respectively, and below 2.5%, 5.0% and 5.0% respectively for most circuits.

Table 3.4

Fingerprinting overheads with state re-encoding for minimum area. $\#FF$ is the number of FFs. A_s and A_m , D_s and D_m , P_s and P_m are areas in μm^2 , delays in ns, and powers in mW of the non-fingerprinting optimized seed design and fingerprinted design, respectively. ΔA , ΔD , and ΔP are their respective percentage increments.

Circuit	$\#FF$	A_s	D_s	P_s	m	A_m	D_m	P_m	ΔA (%)	ΔD (%)	ΔP (%)
S382	21	2212	1.35	1.26	16	2229	1.42	1.33	0.77	5.19	5.32
S838	32	3323	3.60	1.82	16	3453	3.31	1.71	3.91	-8.06	-6.04
S1423	74	8452	5.39	5.10	16	8472	5.38	5.42	0.24	-0.19	6.28
					32	8482	5.03	5.34	0.35	-6.68	4.67
S5378	179	17347	1.68	12.0	32	17414	1.65	12.5	0.39	-1.79	4.17
					64	17540	1.78	12.6	1.11	5.95	5.00
S9234	211	24143	2.67	14.3	32	24462	2.22	15.0	1.32	-16.85	4.90
					64	24702	2.8	14.2	2.32	4.87	-0.70
S15850	534	54151	4.04	34.5	64	54862	3.96	38.8	1.31	-1.98	12.46
					128	55473	4.04	35.3	2.44	0.00	2.32
S13207	638	56496	3.18	35.5	64	57068	3.76	36.2	1.01	18.24	1.97
					128	57633	3.16	36.1	2.01	-0.63	1.69
S35854	1426	159274	3.44	95.5	128	161234	3.57	98.1	1.23	3.78	2.72
					256	163206	3.60	103.8	2.47	4.65	8.69
S38417	1636	166097	3.40	67.7	128	167677	3.78	67.7	0.95	11.18	0.00
					256	168625	3.67	68.8	1.52	7.94	1.62
S35932	1728	150506	1.49	112	128	153370	1.51	110.5	1.90	1.34	-1.34
					256	155752	1.52	120.9	3.49	2.01	7.95

Let $R_F = m/n$ represents the fingerprinting ratio, which is the number of fingerprint bits to the number of bits in DR. The area overhead increases with R_F for all designs. This is expected as the originally optimized encoding instance will have to be re-coded in more segments as R_F increases. However, the overhead of fingerprinting one design with a smaller R_F may not necessarily be lower than that of another design with a larger R_F . This could be due to the design and topology dependent merging of functional and test logic of different encoding instances. For some circuits, the

differences in area, delay and power among different encoding instances are very small. The fingerprinting overheads of these designs may only increase mildly with R_F as opposed to those that have a large quality gap between a suboptimal and the most optimized encoding instances.

As area is used as the selection criterion for the seed design, the delay and power overheads have wider spreads than the area overhead among different designs. For instance, some fingerprinted designs are more than 18% slower than their seed designs while some other could be about 17% faster. This inconsistency is because a smaller instance may not necessarily be faster or consume less power. Hence, the fingerprinted segment that uses a suboptimal encoding instance may have shorter delay or consumes less power than the optimal encoding instance. This explains why the delays or power consumptions of some fingerprinted designs in Table 3.4 are smaller than their seed designs.

Table 3.5

Fingerprinting overheads with state re-encoding for minimum area-delay product. The definitions and units of all notations are the same as those in Table 3.4.

Circuit	#FF	A_s	D_s	P_s	m	A_m	D_m	P_m	ΔA (%)	ΔD (%)	ΔP (%)
S382	21	2202	1.42	1.27	16	2265	1.49	1.37	2.86	4.93	8.21
S838	32	3400	3.32	1.93	16	3463	3.42	1.89	1.85	3.01	-1.92
S1423	74	8436	5.23	5.01	16	8416	4.80	5.10	-0.24	-8.22	1.86
					32	8422	4.84	5.46	-0.17	-7.46	9.05
S5378	179	17064	1.68	12.4	32	17114	1.69	12.6	0.29	0.60	1.61
					64	17264	1.64	12.8	1.17	-2.38	3.23
S9234	211	24160	2.51	15.2	32	24409	2.32	15.8	1.03	-7.57	3.95
					64	24868	2.33	15.8	2.93	-7.17	3.95
S15850	534	54217	4.17	35.9	64	54736	4.06	37.3	0.96	-2.64	3.90
					128	55071	4.07	38.6	1.58	-2.40	7.52
S13207	638	57074	3.62	36.9	64	57869	3.41	38.5	1.39	-5.80	4.34
					128	57909	3.67	38.0	1.46	1.38	2.98
S35854	1426	160306	3.61	97.0	128	161686	3.70	98.0	0.86	2.49	1.03
					256	163523	3.60	97.0	2.01	-0.28	0.00
S38417	1636	166184	3.79	71.6	128	168017	3.46	72.7	0.80	-8.71	1.54
					256	169430	3.93	72.0	1.65	3.69	0.56
S35932	1728	152559	1.56	120	128	155473	1.58	120	1.91	1.28	-0.58
					256	157339	1.58	125	3.13	1.28	3.99

The choice of state encoding influences the complexity of the logic functions of the FSM and hence the area, timing and power dissipation of the synthesis results. Thus, the policy for selecting the optimal state encoding can be adapted to the optimization target. Table 3.5 shows the synthesis results using area-delay product as the selection criterion if the area and delay of the fingerprinted design are equally important. The definitions and units of all notations are the same as those in Table 3.4. This selection policy reduces the delays of fingerprinted instances from those of Table 3.4 by 0.17% on average but their areas are also correspondingly increased by 0.12% on average.

Table 3.6

Area and timing overheads of fingerprinted designs implemented on FPGA. A_s and A_m , D_s and D_m are the areas in number of slices and delays in ns of the seed design and fingerprinted design, respectively.

Circuit	A_s	D_s	m	A_m	D_m	ΔA (%)	ΔD (%)
S382	43	2.58	16	41	2.34	-4.65	-9.44
S838	57	2.61	16	60	2.79	5.26	7.18
S1423	174	4.99	16	176	5.60	1.15	12.19
			32	166	5.51	-4.60	10.85
S5378	223	3.54	32	232	3.79	4.04	7.01
			64	230	3.42	3.14	-3.25
S9234	422	4.16	32	421	5.24	-0.24	25.83
			64	422	4.60	-0.00	10.48
S15850	784	6.05	64	799	5.43	1.91	-10.31
			128	853	5.39	8.80	-10.89
S13207	709	6.04	64	698	5.68	1.27	-5.98
			128	718	5.83	1.91	-3.48
S35854	2660	6.43	128	2669	6.48	0.34	0.82
			256	2674	6.36	0.53	-1.09
S38417	2565	6.94	128	2730	6.97	6.43	0.42
			256	2720	6.96	6.04	0.27
S35932	1729	3.58	128	1749	3.57	1.16	-0.31
			256	1775	3.38	2.66	-5.67

In order to evaluate the area and timing overheads of the fingerprinted circuits on FPGA, designs in Table 3.4 are implemented on Xilinx XC6VCX240T device. The results are shown in Table 3.6. The number of slice registers used by each design is the same as $\#FF$ in Table 3.4. A_s and A_m , and D_s and D_m are the areas in number of slice LUTs and delays in ns of the seed design and fingerprinted design, respectively. For most designs, similar trend of increasing overhead with R_F is observed. On

average, the area and delay overheads are 1.8% and 1.4% respectively. Unlike the fingerprinted solutions of [52], the area and delay of fingerprinted solutions in this proposed scheme can be improved as manifested by the negative ΔA and ΔD values in Table 3.6. This is possible because the resynthesis after state re-encoding may perturb the optimization heuristic to the FPGA fabric mapping, placement and routing problems to escape the local minima. The fingerprinted design may be easier to be placed and routed than the seed design, leading to the use of less logic slices or shorter interconnections for nearly half of the fingerprinted solutions. In addition, the proposed technique can also be applied to IPs targeting antifuse and flash memory based FPGA technologies, which are not feasible for [52]. Due to the dissimilar approaches to logic minimization, the optimal encodings for FPGA may be different from ASIC. The impact of encoding and decoding logic on power consumption is usually higher in FPGA than in ASIC, particularly for large FSMs. Among the 2^n different minimal-bit encodings, Gray encoding and those with reduced hamming distance of the most probable state transitions are more likely to lead to lower resource utilization and power consumption.

Reported experimental results of fingerprinting methods [45, 46, 51, 52] based on only one or a few fingerprint instances for each design are inadequate to demonstrate their capability to generate a large number of reasonably high-quality fingerprinted designs. To evaluate this capability, the same experimental settings for obtaining the data in Table 3.4 were used to synthesize 100 different fingerprinted instances with randomly generated fingerprints and location integers for each design. The minimum, maximum and average areas and delays of the 100 fingerprinted designs for each circuit are listed in Table 3.7. A_m and D_m denote area in μm^2 and delay in ns, respectively. $\%A_{err}$ and $\%D_{err}$ are obtained by calculating the margins of error A_{err} and D_{err} for the area and delay for 95% confidence interval and normalized them by the average area and delay, respectively. The minute margin of error percentages, $\%A_{err}$ and $\%D_{err}$, indicate that different fingerprinted instances of the same IP produced by the proposed technique have consistent quality. By comparing the mean area and mean delay in Table 3.7 with the seed design area A_s and delay D_s in Table 3.4, a more informative group average fingerprinting overheads, ΔA and ΔD , of 1.4% and 2.3% respectively are obtained.

Table 3.7

Synthesis results for 100 different fingerprinted instances of each circuit. A_m and D_m denote area in μm^2 and delay in ns , respectively. $\%A_{err}$ and $\%D_{err}$ are obtained by calculating the margins of error A_{err} and D_{err} for the area and delay for 95% confidence interval and normalizing them by the average area and delay, respectively.

Circuit	m	A_m			D_m			$\%A_{err}$	$\%D_{err}$
		min	Ave.	max	min	Ave.	max		
S382	16	2179	2240	2305	1.32	1.44	1.56	0.26	0.68
S838	16	3320	3388	3473	3.02	3.37	3.78	0.18	0.85
S1423	16	8369	8472	8589	4.71	5.25	5.82	0.12	1.05
	32	8363	8473	8575	4.71	5.28	5.85	0.10	0.98
s5378	32	17349	17358	17550	1.61	1.76	2.06	0.08	0.84
	64	17361	17395	17653	1.62	1.78	1.98	0.11	0.76
s9234	32	24166	24422	24662	2.22	2.51	2.92	0.09	1.01
	64	24266	24579	24915	2.23	2.49	2.87	0.12	0.95
s15850	64	54540	54851	55271	3.42	3.95	4.21	0.05	0.75
	128	54999	55327	55674	3.48	3.99	4.45	0.05	0.79
s13207	64	56788	57094	57397	3.00	3.46	3.90	0.04	0.95
	128	57227	57529	57936	3.16	3.52	4.01	0.05	0.87
s35854	128	160605	161356	161976	3.12	3.57	4.20	0.03	0.96
	256	162122	162880	163729	3.12	3.64	4.11	0.04	1.01
s38417	128	166819	167573	168299	3.41	3.67	4.14	0.03	0.82
	256	168199	168859	169430	3.41	3.75	4.32	0.03	0.91
s35932	128	152905	153419	154109	1.43	1.50	1.64	0.03	0.56
	256	154980	155678	156743	1.43	1.53	1.64	0.04	0.61

There has been no FSM-based fingerprinting technique reported so far. Among the FSM watermarking schemes, the method in [124] is also based on state encoding. Unlike the proposed method, it does not incorporate and blend the FSM logic with testability and thus is unable to realize field authentication of the inserted authorship information. Besides, it is based on graph isomorphism problem and requires the sequential circuit to be represented with the state transition graph (STG). It should be noted that the STG representation is usually not available for large circuits and hence its performance can only be evaluated on several small benchmark circuits with up to 14 FFs and 218 states [124]. In addition, the performance of [124] varies greatly with the used encoding schemes. For example, in the case of the gray encoding and one-hot encoding, the area and delay overheads are (119%, -5%) and (-16%, 57%), respectively. In contrast, the average area and delay overheads of the proposed technique are only (1.4%, 2.3%), which is negligibly small and more acceptable.

Table 3.8

Comparison of synthesis results with FSM watermarking [43]. A_s and A_m , and D_s and D_m are the areas in μm^2 and delays in ns of the seed design and fingerprinted design, respectively.

Circuit	#FF	m	A_s	D_s	A_m		D_m	
					Prop.	[43]	Prop.	[43]
S3330	132	32	29696	45	30024	33504	49.8	46.4
		64			30088	33496	54.2	47
S3384	183	32	42704	69.6	42920	47736	85	83.2
		64			43440	46160	88	83.6
S9234	228	32	57824	67.6	58752	59528	74.2	65.6
		64			58944	59520	73.8	65.2
S6669	239	32	69720	142.8	70728	75136	126.2	134.6
		64			72608	75312	127	134.2
S15850	597	64	129184	132.6	130568	130928	157.2	123
		128			132280	130880	170.4	123.6
S13207	669	64	118632	128.4	120904	126024	143.8	140.2
		128			123920	123456	154.6	131
S38584	1452	64	338816	469.8	343424	345872	484.2	400.6
		128			346608	342520	490.6	401.6
S38417	1636	64	389320	387	391768	372288	404.4	354
		128			393776	373696	423.4	356.4
S35932	1728	64	330968	354.8	333712	366528	361.2	348.2
		128			335328	366920	363	347
B14	245	32	137512	143.6	137896	139664	171.8	142.4
		64			138296	139176	175.4	142.4
B15_1	449	64	220610	132.4	222208	219152	139.4	147.6
		128			223112	218208	148	148
B21	490	64	287080	154.2	288184	298464	196.6	155.2
		128			288880	298720	207.6	155.2
B22	735	64	428008	215.4	429448	429832	274.4	210.2
		128			430760	430168	272.4	209.6
B17	1415	64	710784	374.4	710960	690296	382.4	404.6
		128			712160	689840	403.4	403

On the other hand, the other method in comparison [43] is the only technique that deals explicitly with the problem of field authentication (or off-chip detection) of embedded sequential circuit IP ownership. To facilitate the comparison of the proposed technique with [43], the same synthesis tool SIS [125], technology library *msu.genlib*, optimization script *script.algebraic* and ISCAS benchmark circuits in *blif* format as [43] are used to produce the fingerprinted circuit results in Table 3.8. A_s and A_m , and D_s and D_m are the areas in μm^2 and delays in ns of the seed design and fingerprinted (or watermarked) design, respectively. In most cases, fingerprinted

designs from the proposed scheme have smaller area than watermarked designs of [43], with an average area reduction of about 2% even though reduction of hardware overhead has been well acknowledged as a more challenging problem in fingerprinting than in watermarking. Fingerprinting by state variable encoding of the embedded partitioned test machine has not only resulted in better area optimization but also provided a stronger integration of test and functional logic than the scan-chain based synthesis-for-testability technique. The delay comparison is inconclusive however. Due to the cost function used by the algebraic script from SIS, its meta-heuristic tends to create a larger delay discrepancy between the originally optimized cost surface and the alternate cost surfaces than Synopsys design compiler.

The areas and delays of fingerprinted circuits from Table 3.5 are also compared with those of [39] for the same signature lengths of 64 and 128 bits. The percentage area and delay overheads for each circuit are shown in Table 3.9. Overall, the proposed scheme incurs smaller area and delay overheads. Moreover, being essentially a watermarking scheme, [43] and [39] are incapable of generating many high-quality topologically different designs (for different marks) of the same IP.

Table 3.9

Comparison of area and delay overhead with [39]. A negative overhead means an improvement in the corresponding design attribute.

The probability of successful removal of fingerprint bits by state re-encoding is also analyzed. Based on the resilience analysis of Section 3.4.2, Bob is able to recode only a small number r ($r \ll n$) of state variables from a few randomly created partitions. The probability that the r state variables he re-coded had already been embedded with exactly j fingerprint bits is given by $\binom{m}{j} \cdot \binom{n-m}{r-j} / \binom{n}{r}$, and the probability of modifying exactly i bit values by re-encoding a j -bit binary code is given by $\binom{j}{i} (\frac{1}{2})^i (\frac{1}{2})^{j-i} = \binom{j}{i} (\frac{1}{2})^j$. Thus, the probability of successfully removing k ($k \leq m$) or more fingerprint bits by randomly re-encoding r out of n state variables is given by:

$$P_r(E \geq k) = \sum_{j=k}^m \left(\frac{\binom{m}{j} \cdot \binom{n-m}{r-j}}{\binom{n}{r}} \sum_{i=k}^j \frac{\binom{j}{i}}{2^j} \right) \quad (3.12)$$

Table 3.10 compares the robustness of the proposed method with [43] by the probability of successfully erasing a quarter of the mark, assuming conservatively that the attacker is able to recode $r = m$ state variables for the proposed method and randomly derange $r = m$ FFs for [43]. Fig. 3.10 provides a more intuitive comparison of the probability for the two methods. It can be seen that the proposed method is generally more robust against removal attacks than [43]. As we assume $r = m$, P_r is relatively high when R_F is large. When R_F is around 0.1, P_r reduces drastically to the order of 10^{-8} . The probability of removal is expected to be reduced across-the-board if $r < m$. This observation suggests that it is unlikely for state re-encoding attack to succeed in altering a large fraction of the fingerprint bits even with a moderate R_F . With ECC, it is highly improbable to remove even a single bit of fingerprint for a reasonable-size design. Given a minimum fingerprint length $m_{\min}$ for an allowable probability of coincidence and an empirically determined $R_{F\max}$ for an allowable probability of erasure, the smallest IPs should contain no less than $n_{\max}$ FFs, where

$$n_{\max} = \frac{m_{\min}}{R_{F\max}} \quad (3.13)$$

Table 3.10
Comparison of mark robustness with [43].

Circuit	#FF	m	$P_r(E \geq m/4)$	
			proposed	[43]
S3330	132	32	2.6E-02	3.1E-02
S3384	183	32	3.5E-03	3.1E-02
S9234	228	32	8.1E-04	3.3E-02
S6669	239	32	5.8E-04	3.9E-02
S15850	597	64	5.2E-08	5.6E-08
S13207	669	64	1.0E-08	1.1E-08
S38584	1452	64	9.8E-14	1.0E-13
S38417	1636	64	1.6E-14	1.7E-14
S35932	1728	64	6.8E-15	7.0E-15
B14	245	32	4.9E-04	3.3E-02
B15_1	449	64	2.7E-06	3.4E-03
B21	490	64	8.1E-07	3.6E-03
B22	735	64	2.6E-09	2.8E-09
B17	1415	64	1.5E-13	1.5E-13

Fig. 3.10 Comparison of mask robustness with [43] in terms of $P_r(E \geq m/4)$.

3.6 CHAPTER SUMMARY

In this chapter, a new and robust fingerprinting technique on sequential circuits that offers a convenient way to identify the legal IP owner and users is presented. A large number of high quality fingerprinted instances can be created from an originally optimized seed design of the IP with incremental design effort. The overheads

incurred by different instances have been effectively bounded to a reasonably low level by the proposed method due to its state variable chaining heuristic, dependency based partitioning algorithm and the use of a single blind signature to double as IP ownership and buyer identity marks. The security analysis shows that the fingerprint credibility increases exponentially with fingerprint length and is immune to collusion attack, circuit re-synthesis, retiming and state reduction attacks. The risk of fingerprint removal by state re-encoding attack is significantly reduced with the augmentation of ECC or by keeping the ratio of fingerprint length to state variable number low. From 100 fingerprinted instances for each of the ten ISCAS benchmark circuits tested, the average area and delay overheads were found to be 1.4% and 2.3% respectively with negligible margins of error for 95% confident interval.

Despite of the strong authorship proof and convenience of tracing the original buyer from an illegal copy of the IP, the fingerprinting scheme still has the limitation of just passively confirming the occurrence of IP infringement. In the next chapter, a preventive scheme will be presented for IP cores on SRAM based FPGAs. Each distributed IP instance is encrypted with a specific secret key, and only after the IP instance is decrypted with the key can the functionality be started.

Chapter 4

Pragmatic and Secure Per-Device Licensing Schemes for FPGA IP Cores

4.1 INTRODUCTION

The main merit of realizing pay-per-use licensing scheme for FPGA IPs using conventional cryptographic primitives is that the usage control is achieved with minor or even no modification made to commercial FPGA chips. However, none of existing proposals can be considered pragmatic enough for adoption in the current FPGA IP market. Even though the scheme [80] is declared to be the first of its kind that can be implementable solely based on commercially available FPGA devices, the expensive logistic required for the FV to transfer all his FPGA devices to and fro the TTP's secure site and the lack of an independent entity that is fully trusted by all the parties in the IP market makes the scheme hard to be receptive by the market. This chapter will present a more pragmatic IP licensing scheme for the FPGA IPs. The proposed scheme requires only existing features of recent FPGA devices, and has eliminated the need for an external TTP. The hardware cost has also been minimized substantially by employing only symmetric crypto. Two additional variants of the scheme are also suggested. They use asymmetric crypto to provide better protection to the licensed IP cores at the cost of a temporary increase in hardware resource consumption.

The remainder of this chapter is organized as follows: Section 4.2 presents the prerequisite and judicious assumptions of the scheme. Section 4.3 describes the scheme in details, which includes the explanation of its pivotal module - the core installation module (CIM), the illustration of the protocol execution, and the demonstration of the implementation steps of a multi-IP core system. The security, two possible enhancement variants, practicality, and implementation overhead of the

proposed scheme are discussed in Section 4.4. Finally, the chapter is summarized in Section 4.5.

4.2 PREREQUISITE AND ASSUMPTIONS

The scheme to be presented only employ the existing features of commercial chips described in Section 2.3.2, i.e., the device identifier, on-chip NVM and decryption engine, SHA-256 based HMAC, and self-reconfiguration. It should be noted that the configuration readback feature supported by most FPGA families, which allows the configured bitstream to be read for debugging purpose, must be disabled. Some readback attacks [126] can deactivate the security bits used for disabling the readback feature. Hardened readback disabling circuitry [75] is used in Xilinx Vertex 6 and 7 devices to ensure that external readback is disabled when an encrypted bitstream is loaded.

The scheme does not require an external TTP. Instead, this role is indirectly played by the FV. There are several advantages in this arrangement. Firstly, there have already been direct interactions between the FV and both the SD and CV. Having the FV as the middleman simplifies the communications among different parties. Secondly, there are strong incentives for all stakeholders involved to have the FV facilitate and manage the communications between them. By offering IP repository services to the CVs, the FV can now provide a more complete portfolio of IP cores to value add to his devices, making them more competitive and marketable. SDs find it convenient to look for a wide range of cores that meet their system requirements without having to deal with many different parties and have greater assurance of the tool support to integrate different cores on the FV's devices. CVs will also benefit immensely from the broad customer base of the FV. Thirdly, the FV is committed to his device and service quality and cannot afford to stain his market reputation and credibility by conducting malicious acts such as colluding with the SD to steal the IP cores.

Even though the FV is trusted to act honestly in discharging this duty, the scheme prevents the FV from being accessible to the secret information such as the IP core content and the encryption key. This is different from [79, 81, 127] where the FV has easy access to those secrets.

The scheme assumes that the communication between the CV and FV is authenticated and confidential (by using a standard internet security protocol like transport layer security (TLS)). This is easy to be achieved considering the mutually beneficial strategic partnership between the CV and FV. As the SD is the party which may conduct maliciously, the communication between the CV and SD is assumed to be made through an unprotected public channel.

4.3 THE PROPOSED SCHEME

4.3.1 Core Installation Module (CIM)

A CIM is used exclusively for the secure installation of the encrypted authenticated bitstream onto the FPGA chip. To ensure the design confidentiality and integrity, the protected IP core is delivered in the form of an encrypted authenticated bitstream from the CV to the SD, where the IP bitstream, along with an HMAC key and a HMAC digest of the IP bitstream, is encrypted with a secret IP key K_{IP} , as described in Section 2.3.2. The CIM, delivered together with the protected IP core, is used to generate K_{IP} to decrypt the protected IP core, authenticate its integrity and direct its configuration.

The CIM is created by the CV and has a typical block diagram shown in Fig. 4.1. Basically, the CIM comprises a symmetric decryption engine, a hash function, a comparator, an ICAP interface, a device identifier (e.g., device DNA) interface, three key registers and the associated control unit. One of the three key registers is pre-loaded with the CV's secret key K_{CV} , which is to be used for deriving the secret IP key K_{IP} for IP decryption; the other two registers are empty initially. The secret key K_{CV} is hidden and obfuscated in the routing information of the CIM's bitstream, which is generally believed to be the hardest to reverse accurately [128].

Figure 4.1 Block diagram of the core installation module (CIM).

4.3.2 Protocol Execution

As depicted in Fig. 4.2, the licensing protocol is executed in three phases. These phases are illustrated in details as follows.

4.3.2.1 Device Enrollment

After the HM produces and delivers the FPGA chips to the FV according to the contract, the FV divides the chips into small batches and stores a secret device key K_{FV}^i in the on-chip NVM of the devices, where i is the batch index. This key will be used to encrypt and decrypt the CIM's of the IP cores that are licensed for use on its corresponding batch of FPGA devices. After the device enrollment, each chip k has a record of $\{\#ID_k, i, K_{FV}^i\}$ in the FV's database.

4.3.2.2 IP Core Enrollment

CVs who want to sell their IP cores in the FV's IP store also need to enroll them into the FV's database. The CV needs to provide the description of the core functionality and resource consumption together with a compiled IP model that can be used only for functional simulation but not synthesis. Besides, the CV also sends the associated CIM of the IP core to the FV. To enhance the secrecy of K_{CV} and attack resistance of the cryptographic components (i.e., the symmetric decryption engine and the hash algorithm) in the CIM, the CV can refresh the CIM in the FV's database after certain time with an updated version using a different K_{CV} and the state-of-the-art cryptographic components.

For an enrolled IP core IP_j , the FV issues a core identity $\#IP_j$ to the CV. Hence, each IP core is enrolled into the FV's database as $\{\#IP_j, CIM_j, \text{core description}\}$. The CV also keeps a record of the IP core as $\{\#IP_j, CIM_j \text{ with } K_{CV_j}, \text{IP content } IP_j \text{ and its related information}\}$.

Figure 4.2 The proposed pay-per-use IP licensing protocol.

4.3.2.3 IP Core Licensing

When an SD decides to adopt an external IP core for his system, he searches in the FV's IP store, checks the descriptions of the qualified IP cores and their functional simulation results, and selects the one that best suits his requirements. It should be noted that the CIM is securely stored by the FV and inaccessible to the SD. The SD initiates a licensing request for the chosen IP core to the FV. The request includes the identification information of the SD (denoted by $\#SD$), the ID of the selected IP core, the device IDs of the chips purchased from the FV, and the implementation requirements. Upon receiving the licensing request, the FV checks each received device ID against his database of enrolled devices. If no matching ID is found, the device is not legally produced and the licensing request for the device will be rejected. The FV may be alerted to take action to trace the source of illegal chips. If all the devices' IDs are found in his database, the FV will select a random HMAC key, use the key to generate the digest for the CIM, and then encrypt the CIM, HMAC key and

digest with the on-chip secret key K_{FV}^i . The encrypted authenticated CIM is denoted as $E(K_{FV}^i : CIM_j)$ ³. Finally, the FV transmits the IP request to the CV of the selected IP core, along with the device IDs and the encrypted CIM, $E(K_{FV}^i : CIM_j)$. Depending on the number of batches to which the devices belong, there can be several instances of encrypted CIM.

The following steps are to be carried out by the CV. The requested IP_j is first tailored to meet the implementation requirements of the SD. To deter the SD from IP abuse and thwart collusion attacks, a fingerprint [45, 52, 53] that contains the SD's identity and the CV's ownership information can be inserted into the IP core to identify the IP licensee from each licensed IP instance. Fingerprinting techniques such as the one that the author proposed in [53] can insert a blind fingerprint that carries both the SD's signature as well as the CV's watermark at gate-level with little design effort and overhead. Interested readers can refer to [45, 52, 53] for more details about the fingerprinting of FPGA IPs. A *secret IP key* K_{IP_j} is then selected to encrypt the IP core in the encrypted authenticated form, i.e., $C_j = Enc_{K_{IP_j}}(IP_j)$, for this transaction, and a different K_{IP_j} is used for each SD. Using a different K_{IP_j} in each transaction enhances the security of C_j . After all, the IP_j for each SD has to be independently encrypted and is unique due to the different implementation requirement. Meanwhile, a *secret device token* $K_{CV_j}^k = h(K_{CV_j}, \#ID_k)$ for each device k is also generated by the CV based on the device ID and his secret key K_{CV_j} . The device token is then used to encrypt the secret IP key to create a *device-specific license token* $T_{CV_j}^k = E(K_{CV_j}^k : K_{IP_j})$ in tamper-proof encrypted and authenticated form. After invoicing the SD, the CV delivers the encrypted IP core C_j along with the license token $T_{CV_j}^k$ and the encrypted CIM, $E(K_{FV}^i : CIM_j)$ for each device k of batch i ordered or purchased from the FV.

³ In fact, all the encrypted data in our scheme are in the encrypted authenticated form. For brevity, they will be referred to as encrypted data.

4.3.3 Implementation of Multi-IP Core System

After completing the system design that contains in-house IP modules and externally licensed IP cores, the SD stores and queues all the design modules up in the external NVM as shown in Fig. 4.3. The queue can be divided into two parts, where the first part comprises external IP cores obtained through licensing and the second part comprises the internally designed cores, free third-party IP cores and other non-security sensitive functional blocks. The first part consists of several segments, each corresponding to an IP core. Each segment contains one encrypted CIM, one license token and the encrypted core. Upon powered up, the chip is first loaded with the encrypted CIM_1 of the first IP core. The CIM_1 is deciphered by the on-chip decryption engine and authenticated against its HMAC code before it is implemented onto the FPGA. Once implemented, CIM_1 takes control of the configuration of IP_1 . Firstly, the hash function of CIM_1 uses the two inputs, K_{CV_1} and $\#ID_k$, to generate the secret device token $K_{CV_1}^k$. This device token is stored into one of the two empty key registers of CIM_1 . Then the license token $T_{CV_1}^k$ is fed to the decryption engine of CIM_1 and is decrypted using $K_{CV_1}^k$ to recover the secret IP key K_{IP_1} . Upon successful authentication against the HMAC digest contained in the decrypted $T_{CV_1}^k$, the IP key K_{IP_1} is stored in the other empty key register. Next, the encrypted IP core C_1 is decrypted with K_{IP_1} . The decrypted IP_1 is authenticated against the HMAC code contained in the decrypted C_1 . If the authentication is passed, the bitstream will be delivered to the configuration controller through the ICAP port for implementation. Failure in any of the above authentications will either render the clearing of the configuration and loading of a fallback bitstream or the disabling of the configuration interface to block any access to the device until a power-on reset is applied.

Figure 4.3 Queue of design modules in external NVM of device k in batch i .

These steps are depicted in Fig. 4.4. The hash component and the comparator for the authentication is combined with the symmetric key decryption and represented by the component named *Decrypt & Authenticate*. After the successful configuration of IP_1 , the bitstream of the second segment is loaded. A new CIM replaces the CIM of the first IP core to control the configuration of the second core. The above steps repeat until all the licensed IP cores have been implemented. Finally, the fabrics occupied by the last CIM are released for use with the available uncommitted logic to implement the rest of the functional blocks.

(a) Steps for implementing the encrypted CIM_1 (b) Steps for establishing the secret IP key K_{IP_1} (c). Steps for configuring IP_1 with K_{IP_1} Figure 4.4 Configuration steps for protected IP_1 on device k in batch i .

4.4 EVALUATION AND DISCUSSION

4.4.1 Security Analyses

Fig. 4.5 provides an overview of the communications among FV, CV and SD for authorizing the use of IP_j on device k . The IP core is protected by keeping all its sensitive data (i.e., CIM, license token and IP core) secure in encrypted authenticated form at all time in transit from one party to another and when they are stored in the external NVM of the chips. Authenticity of these data is further verified with the HMAC code before they are configured onto the device to stop the maliciously tampered IP core from being used. Only standard cryptographic primitives (i.e., AES, SHA-256 and TLS security protocol) are used in the scheme, so that the protected data in the scheme is immune to common attacks such as chosen and known plaintext attacks, replay, parallel session and type-flaw attacks.

In [24], similar standard and provably secure symmetric cryptographic components are also used for the key establishment. Instead of directly passing the IP decryption key to the metering authority like [24], the presented scheme passes the CIM that contains the secret information for deriving the key. This setup can better protect the secret key from being stolen.

The following analyses show that the licensing protocol will safeguard the IP core of the CV against infringement and misappropriation by the SD, external MA and FV.

Figure 4.5 Communication of IP core pertinent data among FV, CV and SD.

4.4.1.1 Evasion of license fee by SD

In order to configure an encrypted IP_j onto a device k , the CIM_j associated with IP_j and a device specific license token $T_{CV_j}^k$ are required. To evade the licensing fee by implementing IP_j on other devices, an SD needs to know the secret IP key K_{IP_j} , which can only be recovered from the license token $T_{CV_j}^k$ using the secret device token $K_{CV_j}^k$. There are two possible attacks from the SD. One attack is to extract the secret device key K_{FV}^i from the on-chip NVM and use it to obtain the plaintext CIM_j . By reverse engineering CIM_j , the CV's secret key for the IP core K_{CV_j} may be extracted, from which the decryption key can be derived. The other attack is to directly steal the secret device token $K_{CV_j}^k$ or the IP key K_{IP_j} during the configuration process through the fault injection attack, side-channel attack or other more expensive physical attacks.

4.4.1.2 Attacks by MA

It is relatively easy for an external MA to obtain the encrypted data sent from the CV to an SD by eavesdropping on the communication channel between them if the channel is not secure. However, as the cryptographic components adopted in the scheme are widely accepted as computationally secure, extracting the plaintext IP core from the encrypted data without knowing the keys is prohibitively expensive, if not impossible. From Fig. 4.5, it is observed that the only way for the MA to succeed in stealing the IP core is to steal first the plaintext CIM and then an encrypted IP instance. The MA needs to reverse engineer the CIM to recover the embedded K_{CV_j} before he could derive the secret key for the IP instance. As the communication channel between the FV and the CV is secure and authenticated, it prevents the plaintext CIM from being stolen by the MA through attacks such as eavesdropping on the channel, replay attacks, parallel session attacks and type-flaw attacks. Even though the MA may obtain the CIM by some sophisticated attacks and extract K_{CV_j} by reverse engineering the CIM, he has not yet succeeded in stealing the IP core. This is because K_{CV_j} is changed after certain period by the CV. The MA could only succeed in his attack provided that he also manages to obtain an encrypted IP instance

that has its encryption key derived from the same K_{CV_j} for the CIM in his possession. The probability that he could intercept a matching IP instance transmitted through a different channel between another pair of communicators (CV and SD) at a different and unknown time is very slim, and the jeopardy is limited to only those matching IP instances of the stolen CIM.

4.4.1.3 Breaching of trust by FV

In [79, 127], the FV can derive the secret key straightforwardly for the IP decryption, which increases the potential liability of the FV for frauds. To minimize the liability of the FV, the protocol does not assume that the FV is fully trusted. The secret information pertaining to the IP core is refrained from being accessible by the FV, except the plaintext CIM. Some additional measures to fortify the CIM to increase the barrier for the FV to steal the IP core will be presented in Section 4.4.2. Collusion attack between the FV and the SDs is also thwarted by fingerprinting the IP core. The stakes are high as the fingerprint of the misappropriated IP instance will expose the SD who participated in the collusion.

To satisfy the desiderata described in Section 2.3.1.3, only the plaintext CIM is made known to the FV and standard secure internet protocol is adopted for the communications between the FV and the CV. These make it harder to exploit the FV as an oracle to mount a successful parallel session attack. The above analyses show that the cost of a successful attack to steal an IP core will be prohibitively expensive if the FV acts honestly. Even when this assumption is not met, i.e., the FV conducts maliciously, the scheme is still more secure than the protocols of [79, 127].

4.4.2 Possible Security Enhancements and Variants

FV can use fault tolerant and side-channel resistant on-chip cryptographic components to protect the on-chip secret device key K_{FV}^i , which make it harder for an MA to obtain the plaintext CIM. In the scheme, each K_{FV}^i is used for only a small batch of devices. The leakage of a K_{FV}^i affects only the IP cores that are implemented on a particular batch of devices. By carefully controlling the size of a batch, the benefit of extracting the key can be reduced to an extent that the expense of

the attack can hardly be justified. In fact, the benefit can be further reduced by blacking out the batches of devices from the list of legal devices once the security of their device keys are suspected to be compromised. Only after these devices are reprogrammed by the FV with the new device keys can they be used as trusted implementation platforms and participating in the licensing protocol again.

To make the extraction task harder for the MA, K_{CV_j} is proposed to be hidden in the routing information of CIM's bitstream as in [128]. Alternatively, white-box block ciphers [129, 130], which can protect the embedded secret key through strong obfuscation, can be used as the decryption engine of CIM to raise the barrier of successfully extracting the key.

Besides, although the scheme is based on commercially available hardware primitives, emerging new primitives can also be used once they become commercially available. For example, if the PUF primitive is available in commodity chips, it can be used in place of the current hardwired device ID to generate the secret device key K_{FV}^i . With the tamper-resistant PUF generating the secret key only on demand, the invasive and time consuming micro-probing attack can also be thwarted.

Moreover, the scheme can be strengthened with the following two variants where public-key crypto is employed.

4.4.2.1 Adding an ancillary establishment module

An ancillary *establishment module* (EM) can be added to the scheme. The EM consists of an Elliptic Curve Diffie-Hellman (ECDH) core, an AES core and a unique private key SK_{EM}^k for device k . The EM is encrypted by the secret device key K_{FV}^i . The public-private key pair $\{PK_{EM}^k, SK_{EM}^k\}$ of the EM for each device is stored in the FV's database. The public key PK_{EM}^k for the device k is sent with the IP request from the FV to the CV in Step 4 of Fig. 4.2 (c). Using ECDH with PK_{EM}^k and his private key SK_{CV} , the CV will generate a shared key to encrypt his CIM by AES. The public key PK_{CV} of the CV is transmitted to the SD along with the encrypted CIM in Step 9 of Fig. 4.2 (c). The implementation steps are similar to the current scheme, except that the decryption key for the CIM is firstly generated by the ECDH core of

the EM using SK_{EM}^k and PK_{CV} . The additional cost of the strengthened scheme is only the ECDH core as the same AES core can be shared by the EM and the CIM. The advantages are that all the important data related to the IP core, including the CIM, are in encrypted form when they leave the site of the CV. The secret device key K_{FV}^i is used merely to protect the EM of the FV instead of various CIMs of different CVs, thus effectively avoiding the chosen and known plaintext attacks. Although ECDH core is employed as in [79, 127], it does not require the FV to act as the fully trusted TTP and avoids any party except the CV from knowing the secret information related to the IP core. More importantly, there is no need to transport the devices to the third party site for key installation.

4.4.2.2 Use asymmetric crypto for IP key establishment

Although the alternative described in Section 4.4.2.1 employs public-key crypto, it is still based on symmetric crypto for recovering the IP decryption key. The security of the key is ensured by letting the CV creating the corresponding CIM. In fact, if the temporary implementation overhead of an asymmetric crypto based function is tolerable, the IP decryption key can be recovered using asymmetric crypto. Such an alternative will help to simplify the required communications and provide a higher level of confidentiality of the IP content to the FV. Besides, the CIM in this case can be provided by the FV, eliminating the need for the CV to create and manage the CIM.

The structure of the CIM is depicted in Fig. 4.6. It can be stored in the external NVM of each FPGA chip and contains the private key SK^i of a unique public-key pair $\{PK^i, SK^i\}$ for batch i of chips. Each CV will also devise a key pair $\{PK_{CV_j}, SK_{CV_j}\}$ for his IP core IP_j . The confidentiality of the IP decryption key is realized by letting the SD generates a purchase token $P_j = Enc_{PK_{CV_j}}(\eta_j)$ for each request, which is a random nonce η_j encrypted with the CV's public key. The nonce η_j will be used for deriving the device specific IP decryption key $K_{IP_j}^k$ by the CV and the specific device k as shown in Equations (4.1) and (4.2), respectively. This way, although the FV has the knowledge of $\{PK^i, SK^i\}$, he is not aware of η_j and hence cannot devise $K_{IP_j}^i$.

$$K_{IPj}^k = KDF(\#ID_k, SK_{CVj}, PK^i, \eta_j) \quad (4.1)$$

$$K_{IPj}^k = KDF(\#ID_k, SK^i, PK_{CVj}, \eta_j) \quad (4.2)$$

Figure 4.6 Structure of the CIM using asymmetric-crypto for key establishment.

The implementation steps for this variant are also similar to those described in Section 4.3.3, except that the encrypted CIM for each IP core is replaced by just one encrypted CIM provided by the FV and that η_j needs to be stored in the NVM.

4.4.3 Practicality of the Scheme

The presented scheme can be immediately applied in the current FPGA IP market. Apart from basing on only the commercially available primitives and features, the scheme does not change current FPGA design and implementation flow. Most FPGA chips that support on-chip decryption are equipped with more than one secret key memory. For example, even the FPGA devices from the low-cost Xilinx Spartan-6 family support the hardwired decipher to read the secret key from a battery-backed SRAM-based key storage besides the on-chip NVM. Hence, the way by which the SD used to protect his own internally designed functional blocks will not be constrained by the on-chip NVM key storage taken up by the FV's secret device key K_{FV}^i . He can protect his own designs by encrypting them using a secret key and store the secret key in the alternative key storage. During the implementation of the final system, the SD's own IP cores will be loaded onto the configuration controller after the externally purchased IP cores have been implemented. The configuration controller will instruct the hardwired decipher to decrypt these cores before configuring them to the previously planned locations of the chip.

Modular design methodology and partial re-configuration are required in the scheme. Functional blocks are implemented onto the allocated resources of the chip one by one at the bitstream level. This makes it impossible to combine all the functional blocks of the system before a system-level synthesis is run for optimal performance. In fact, both the modular design method and partial re-configuration may incur some additional design effort and resource consumption. Nonetheless, these features have already been supported by the design tools from the mainstream FVs. With the continual advancement of these tools, the additional design effort and performance overhead of the overall system is expected to be reduced substantially. Each IP core has actually been maximally optimized by the CVs. Precise physical synthesis information, which includes the resource allocation and interface logic, can be incorporated into the floorplanning to better tailor the IP cores to the system requirements. By using standardized bus interface like AXI and specific tools like Xilinx PlanAhead to floorplan the functional blocks in the system, the impact on the performance of the final system due to the lack of global system optimization will become marginal.

4.4.4 Implementation Considerations

The scheme does not require any modification to currently available FPGA devices and no new party is introduced. The FV needs only to extend his existing upfront IP core licensing services to clients who opt for pay-per-device licensing. The proposed scheme effectively inhibits the use of gray market chips and counterfeit chips for the installation of legally licensed cores, which gives the FV the added incentives to act as the principal of the CV and SD besides the validity of his service obligation established in Section 4.2. The scheme puts less burden on the FV than the existing schemes [79, 127]. It prevents the FV from issuing and updating the CIM, and acting as a completely trusted TTP that may lead to greater tort liability. Unlike [15, 23], the protocol enables the CV who has the greater interest and responsibility to protect his IP cores to design and enhance the CIM based on his security requirements. At the same time, the communications among the involved parties in the scheme are simplified tremendously comparing to the protocol in [80] which requires an external TTP to act as the metering authority. An immediate benefit of the setup is that the

high logistic cost of transporting the chips to and fro between the FV and the external TTP sites for programming the secret device key is avoided.

As for the implementation overhead, the CIM employed in the scheme occupies the reconfigurable resources only temporarily. These resources will be released for other functional blocks after the configuration of protected IP cores. Nonetheless, it is still necessary to keep the resource consumed by the CIM low to preserve as much uncommitted user logic for the configuration of IP cores and to keep the operations performed by the CIM simple and fast so as to minimize the overall system configuration time. Unlike the CIM in [79, 127], which needs to establish an ECDH core of 2706 slices, the most resource hungry component of the CIM in the proposed scheme is the symmetric key decryption engine and the hash function based authentication component if the security-enhanced EM option in Section 4.4.2 is not considered. This is similar to the CIM of [80] whose main component is an authenticated symmetric decryption engine.

As an example, the resource consumption of the CIM on Xilinx Vertex-6 devices is evaluated. Except for the symmetric decryption core and the HMAC core, the CIM contains only three key registers, the device DNA and ICAP ports, and the associated control unit. The device DNA and ICAP ports are simply the device-specific primitives provided by the FPGA and consume no more than 100 equivalent slices together with the three key registers and the simple control unit of CIM. Both the decryption engine and the hash function can be constructed using commercially available IP cores, such as the AES core of 331 slices running at 5236 Mbps [131] and the SHA-256 core of 312 slices running at 2071 Mbps [132]. All in all, a succinct CIM consumes just less than 800 slices. The smallest Vertex-6 device XV6VLX75T has 11,640 slices, the largest device XV6VLX760 has 118,560 slices and the average size of all device types is 54,828 slices. The CIM utilizes 6.4%, 0.6% and 1.4% of the available resources in the smallest, largest and average size devices, respectively.

The implementation overheads reported for existing FPGA based IP protection schemes [55, 56, 70, 78-81, 127, 133, 134] does not include the overheads required for countermeasures against common types of physical attacks. As the secret key of an unprotected cryptographic design can be deduced by low-cost physical attacks such as power analysis and fault injection, the cost of designing a CIM with

reasonable level of protection against these attacks need to be accounted. Countermeasures [135-139] against the power analysis attacks can be broadly divided into two main types: algorithmic based and generic hardware based. The former category masks the security-critical processes by computational manipulation while the latter masks the side-channel signals by using noise generators, non-deterministic processors or side-channel resistant logic styles. The overheads incurred by different methods in both categories are different and vary significantly. The overheads of some masked AES implementations against power analysis attacks are listed in Table 4.1, where the area overheads range from 20% to 300%. For the generic hardware countermeasures [139], a short-circuit based noise generator consumes only 48 LUTs on Xilinx Vertex II pro and the block memory content scrambling method uses 8 BRAMs, 1706 LUTs and 1169 FFs. Comparing with the plain AES implementation, which uses 8 BRAMs, 1182 LUTs and 397 FFs, the overheads of generic hardware countermeasures may vary widely from less than 4% to more than 140%. In fact, the CV can select a countermeasure or combine any of the above mentioned methods to trade the overhead for security. For example, a combination of noise generator, clock randomizer and block memory content scrambler [139] strengthens the CIM's protection level by increasing the area and timing overheads to around 200% and 60%, respectively. The masked s-box implementation in [136] can be used together with the block memory content scrambling technique at the expense of 200% area overhead and negligible timing overhead. It is estimated that the CIM can be well protected against common power analysis attacks by a countermeasure of 200% or less area overhead and an acceptably low timing overhead, as exemplified by the relatively strong masked SHA-256 based HMAC [140] with 100% area overhead and 13.4% timing penalty.

Table 4.1

Overheads of Boolean and multiplicative masking [135], masked S-box [136], masked S-box optimized by larger LUT in recent FPGA [137] and dual-rail precharge logic [138] methods of masked AES implementation.

Performance	[135]	[136]	[137]	[138]
Area overhead	20%	60.1%	200%	300%
Timing overhead	30%	4%	50%	33.3%

An overview of countermeasures against fault attacks can be found in [141], where two major types of techniques, namely fault detection using space redundancies and using duplication are illustrated. According to [141], the actual costs of many countermeasures are close to duplication if fair comparisons are performed. Although some recent proposals claim to have area overhead lower than 100%, e.g., the method in [142] incurs less than 50% area and 2% timing overheads, it is conservatively assumed that a countermeasure like the duplication scheme in [143] for the AES core with about 100% area overhead and no throughput penalty is used.

The countermeasure against the power analysis attack is applied after the duplication method on the cryptographic components in the CIM. The resulting cryptographic components are expected to consume 6 times the resources of the unprotected design. This means that the protected AES and SHA-256 cores will consume 1986 and 1872 slices, respectively. The entire CIM will consume around 4000 slices, which is still less than 8% of the resources of an average size Vertex 6 device. The resources consumed by the four versions of CIM are shown in Table 4.2.

Table 4.2

Resource consumptions (in # equivalent slices) on Xilinx Vertex-6 devices of the unprotected, side-channel attack (SCA) resistant, fault attack (FA) resistant, and final physical attack (PA) resistant versions of CIM.

CIM	Decryption	HMAC	Others	Total	Utilization rate		
					smallest	average	largest
Unprotected	331	312	100	743	6.4%	1.4%	0.6%
SCA resistant	993	936	100	2029	17.4%	3.7%	1.7%
FA resistant	662	624	100	1386	11.9%	2.5%	1.2%
PA resistant	1986	1872	100	3958	34.0%	7.2%	3.3%

AES and SHA-256 are used in the discussion because they are the NIST-approved standards widely adopted by FVs like Xilinx and Altera in the hardwired on-chip decryption and authentication engine. It should be noted that the CV can substitute the AES and SHA-256 based HMAC core with any state-of-the-art cryptographic algorithms, such as Keccak, the winner of the NIST secure hash algorithm (SHA-3) competition in 2012. A good overview of its implementations, including those with the protection against SCA, is provided in [144]. Keccak does not have the security hole of SHA-256, which leads to a simplified HMAC digest generation without the need for a nested approach. It also provides the mode of authenticated encryption

(based on duplex construction), making it possible to replace the AES and HMAC cores by a single Keccak core to compact the CIM.

4.5 CHAPTER SUMMARY

In this chapter, a pragmatic per-device licensing scheme, which can be directly applied to existing devices without requiring any modifications, is presented. The scheme empowers the FPGA vendor to act as a broker for the system developers and core vendors. It facilitates the secure transaction of protected IP cores and effectively prevents the IP core content from being cloned, reverse engineered or tampered with by even sophisticated attackers. By fingerprinting the IP cores, misappropriation and infringement of their contractual usage by the system developers can be tracked. The scheme incurs low implementation overhead, even after adding the countermeasures against common physical attacks. This is ascribed to the simple symmetric cryptography and hash function used in the core installation module and its temporary occupancy of user logic. Two variants of the scheme which employ public-key crypto are also illustrated. They are able to provide even better security to the IP core at the cost of higher temporary FPGA resource consumption.

Without requiring an external TTP, the scheme can be applied immediately in the FPGA IP market to replace the current upfront licensing model with a more competitive per-device licensing model. Not only will the per-device cost be considerably reduced for the system developer, but the core vendor can also control the number of FPGA chips that is implemented with the IP core licensed to the system developer. An extra benefit of binding the distributed IP core to the chip identity is that the chip is identified before the licensing request is processed, which effectively denies the use of overproduced FPGA chips and counterfeit chips.

Chapter 5

Hardware Trojan Detection with Linear Regression based Gate-level Characterization

5.1 INTRODUCTION

Classic hardware Trojan (HT) detection methods have been reviewed in Section 2.4. Among them, gate-level characterization (GLC) based approaches are attractive for the following reasons. Firstly, as a side-channel analysis (SCA) technique, GLC based approaches outperform many other types of SCA techniques in their high HT detection sensitivity despite the process variation (PV); Secondly, most HT detection methods require the creation of a golden model, against which an IC under authentication (IUA) will be compared. The most reliable means to establish a golden model is through destructive reverse engineering, which is very expensive and other means to obtain a golden model is unreliable and often impractical. GLC based approaches have completely do away with such a requirement.

Most of the existing GLC-based HT detection methods [100, 102-105, 145] try to realize accurate characterization of every gate before verifying the existence of HT. With each gate accurately characterized, the existence of HT will show up obviously by the abnormal scaling factors. Besides, if the HT exists, the HT location, size, type, etc. may also be inferred from the GLC results to facilitate HT diagnosis [103, 105]. However, complete and accurate GLC of every circuit gate is computational intensive and not always feasible. For example, to identify and break the correlation of each gate, thermal conditioning technique is used [102, 103]. The effort required is non-trivial. To ensure the accuracy of complete GLC, it requires expensive computations, series of statistical methods and a large number of measurements.

Post-silicon test time is expensive. Although complete and accurate GLC provides a high HT detection sensitivity, it is too slow for thousands of IUAs. In this chapter, a new GLC based approach for detecting functional HTs in the form of additional gates and/or FFs will be presented. Based on post-silicon leakage current measurements, a fast GLC process will be proposed. Despite the presence of gate correlations, scaling factors of the circuit gates that are not in correlation can be efficiently and accurately estimated by solving the normal equation in linear regression analysis. Based on the disparity in the variation of the bias term introduced by linear regression and a subset of accurately estimated scaling factors, HT-infected chips can be easily distinguished from the genuine chip. The detection sensitivity is much higher than [] that simply checks the disparity in the averaged sum of all scaling factors. As the HT gates consume leakage power at all time, the method presented in this chapter does not require the HT to be activated in order to detect it, making it possible to detect different types of HT, including those that are hard to be randomly triggered.

The remainder of this chapter is organized as follows: Section 5.2 presents the preliminaries for formulating the HT detection problem. Section 5.3 describes the proposed HT detection method. Experimental evaluations of the proposed scheme are presented in Section 5.4. Finally, the chapter is summarized in Section 5.5.

5.2 PRELIMINARIES ON PROBLEM FORMULATION

5.2.1 Process Variation

A unique characteristic of deep sub-micron technologies is the intrinsic non-deterministic variations of process parameters. Such process variation (PV) is usually classified into two categories: inter-die and intra-die variations. Inter-die variations account for the variations arising between chips in the same or different wafers and its effect can be considered as being constant over a specific chip. In contrast, intra-die variations affect the devices in the same IC differently. Due to the normally distributed intra-die variations of device dimensions (like gate oxide thickness and effective channel length) and the exponential relation between these dimensions and the leakage current, the distribution of the leakage current variation is approximately lognormal [146]. Leakage current variation of a device can be deemed as an independent skewness that results in $I_{leak}^r = s \times I_{leak}^n$, where I_{leak}^n and I_{leak}^r are the nominal and

real leakage currents for the device, respectively, and s is the scaling factor ascribed to the PV.

The scaling factor is usually assumed to be the same for different states of a device. These states typically correspond to different nominal leakage values, which are readily available from the simulation models or other sources.

5.2.2 Gate-level Characterization of Leakage Current

By representing PV as a scaling factor of the leakage current of each device, the total leakage current of an IC can be obtained by summing up the leaking current contributions from each gate on the chip. For every input vector, the state of each gate can be determined from the design information, e.g., the gate-level netlist. By summing up the scaled nominal leakage values of all gates and from the measured total leakage current for each input vector applied at the input pins, a system of linear equations (SLE) can be formulated. The i -th linear equation of the SLE is given by:

$$I_{leak}^m(\mathbf{v}_i) = \sum_{j=1}^N s_j \times I_{leak,j}^n(\mathbf{v}_i) \quad (5.1)$$

where N is the number of gates in the chip, $I_{leak}^m(\mathbf{v}_i)$ and $I_{leak,j}^n(\mathbf{v}_i)$ are respectively the measured leakage current of the chip and the nominal leakage current of gate j for the i -th input vector $\mathbf{v}_i$, and s_j is the PV scaling factor for gate j .

As the measurements are usually taken at the external pins, they are highly susceptible to measurement errors due to variations in environmental conditions, thermal effects and noise. Taking the measurement errors into account, $I_{leak}^m(\mathbf{v}_i)$ in Equation (5.1) is the sum of the chip leakage current $I_{leak}^c(\mathbf{v}_i)$ and the measurement error $e(\mathbf{v}_i)$. The scaling factors are the variables to be estimated and can be calculated by solving the SLE with an objective function of minimizing the measurement errors.

5.2.3 Linear Regression with Multiple Variables

Let $\mathbf{x}^{(i)} = (x_1^{(i)}, \dots, x_N^{(i)})$ be an input vector of N features and $y^{(i)}$ be the desired output. Assume that $\mathbf{D} = \{\mathbf{D}^{(1)} \dots \mathbf{D}^{(M)}\}$ is a set of M training data, where $\mathbf{D}^{(i)} = \langle \mathbf{x}^{(i)}, y^{(i)} \rangle$ is the i^{th} data. A bias term $x_0^{(i)} = 1$ is added to each input vector to learn the mapping from $\mathbf{x}^{(i)}$ to $y^{(i)}$ for all $i=1, \dots, M$ by the linear regression model in Equation (5.2). The optimal set of parameters $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_N)$ can be obtained with the objective of minimizing the averaged sum of squared errors $J(\boldsymbol{\theta})$ in Equation (5.3).

$$h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \dots + \theta_N x_N^{(i)} = \boldsymbol{\theta}^T \cdot \mathbf{x}^{(i)} \quad (5.2)$$

$$J(\boldsymbol{\theta}) = \frac{1}{2M} \sum_{i=1}^M (y^{(i)} - \boldsymbol{\theta}^T \cdot \mathbf{x}^{(i)})^2 \quad (5.3)$$

The M input vectors $\mathbf{x}^{(i)}$ (with $x_0^{(i)}=1$) form a matrix $\mathbf{X}$ of dimension $M \times (N+1)$, while the M outputs $y^{(i)}$ form a column vector $\mathbf{y}$ of dimension M . Equation (5.3) can be rewritten in terms of $\mathbf{X}$ and $\mathbf{y}$ in Equation (5.4). The optimal set of linear regression parameters $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_N)$ that minimizes $J(\boldsymbol{\theta})$ can be obtained when the partial derivative of $J(\boldsymbol{\theta})$ with respect to each parameter θ_j is 0, as expressed in Equation (5.5).

$$J(\boldsymbol{\theta}) = \frac{1}{2M} (\mathbf{y} - \mathbf{X} \cdot \boldsymbol{\theta})^T \cdot (\mathbf{y} - \mathbf{X} \cdot \boldsymbol{\theta}) \quad (5.4)$$

$$\nabla J(\boldsymbol{\theta}) = -\frac{1}{M} \mathbf{X}^T \cdot (\mathbf{y} - \mathbf{X} \cdot \boldsymbol{\theta}) = \vec{0} \quad (5.5)$$

where $\vec{0}$ is a vector of $(N+1)$ 0's.

By rearranging the terms in Equation (5.5), a normal Equation (5.6) is derived for solving $\boldsymbol{\theta}$.

$$\boldsymbol{\theta} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y} \quad (5.6)$$

Even though $\mathbf{X}^T \cdot \mathbf{X}$ is non-invertible, the normal equation can be efficiently solved by using singular value decomposition to compute the pseudo-inverse of $\mathbf{X}^T \cdot \mathbf{X}$ [147].

5.3 THE PROPOSED HT DETECTION METHOD

The proposed HT detection method is based on the observed similarity between Equations (5.1) and (5.2). The main idea is to map the scaling factors in the SLE $\sum_{j=1}^N s_j \times I_{leak,j}^n(\mathbf{v}_i)$ to the parameters in the linear regression $\theta_1 x_1^{(i)} + \dots + \theta_N x_N^{(i)}$, while treating the gate nominal leakage currents as the features in the vector x and the measured leakage current $I_{leak}^m(\mathbf{v}_i)$ with measurement error $e(\mathbf{v}_i)$ as a predicted value of $h_\theta(\mathbf{v}_i)$ with some prediction error. The remnant θ_0 can then be ascribed to the extraneous anomaly. By minimizing the measurement error, $I_{leak}^c(\mathbf{v}_i)$ of an HT-free circuit can be construed as contributed entirely by the leakage of N circuit gates, and θ_0 is usually very small. Additional leakage contributed by the existence of HT to $I_{leak}^c(\mathbf{v}_i)$ will thus be manifested by the abnormally large value of θ_0 or by the inconsistency in θ_0 values obtained from multiple runs with different test vector sets. The reason for the latter is that an HT is an extraneously introduced sub-circuit and its leakage current will also vary with the input vector like any other circuit gates. To improve the HT detection sensitivity, instead of using the absolute value of θ_0 , the consistency of θ_0 under different test vector sets is used to verify the existence of HT. The consistency is measured by the variance of θ_0 for each fabricated IC, which has a higher tolerance to the PV noise than the calculated absolute value for θ_0 .

In Equation (5.6), the only computationally expensive operation is the matrix inverse, which can be efficiently solved using singular value decomposition techniques. Hence, the vectorized implementation in Equation (5.6) gives a fast approximation of the scaling factors. If the input matrix X is full rank, the calculated θ_j ($j = 1, \dots, N$) will match well with the actual scaling factor for each gate. However, due to the gate correlations, which make the coefficients of some gates collinear in X , the rank of X is usually smaller than N , resulting in only a portion of θ_j 's being a good approximation of the actual scaling factor. The accurately estimated θ_j 's, together with θ_0 , are to be used as an indicator for the existence of HT, denoted by $Ind(\theta)$. By examining the variance of each element in $Ind(\theta)$, an HT-infected circuit will be identified.

The HT detection method consists of two phases, i.e., the HT indicator extraction phase and the HT detection phase. In phase I, elements of the HT indicator $Ind(\theta)$, i.e., the accurately estimated θ_j 's and θ_0 , are to be identified. The mean and variance of

each element in $Ind(\theta)$ will be recorded. This phase can be performed by the design house using the simulation model of the circuit incorporating the PV, where each gate is incorporated with a simulated scaling factor. Due to this setup, the need for an HT-free golden chip is avoided.

For a circuit of N gates, a set of $M = gN$ ($g > 1$) test vectors is randomly generated. The test vectors will produce an SLE of M linear equations in the form of Equation (5.1), from which an $M \times N$ matrix G of the gate nominal leakage current and an M -element vector y of the measured leakage current can be constructed. The scaling factors can be calculated using the normal equation of Equation (5.6) after augmenting the first column of G with an M -element column vector of all ones and forming the input matrix X . The generic forms of the SLE, vector y and matrix X are depicted in Fig. 5.1.

$$\begin{cases} s_1 \cdot I_{leak,1}^n(\mathbf{v}_1) + s_2 \cdot I_{leak,2}^n(\mathbf{v}_1) + \dots + s_N \cdot I_{leak,N}^n(\mathbf{v}_1) = I_{leak}^m(\mathbf{v}_1) \\ s_1 \cdot I_{leak,1}^n(\mathbf{v}_2) + s_2 \cdot I_{leak,2}^n(\mathbf{v}_2) + \dots + s_N \cdot I_{leak,N}^n(\mathbf{v}_2) = I_{leak}^m(\mathbf{v}_2) \\ \dots \\ s_1 \cdot I_{leak,1}^n(\mathbf{v}_M) + s_2 \cdot I_{leak,2}^n(\mathbf{v}_M) + \dots + s_N \cdot I_{leak,N}^n(\mathbf{v}_M) = I_{leak}^m(\mathbf{v}_M) \end{cases}$$

(a)

$$\begin{bmatrix} I_{leak}^m(\mathbf{v}_1) & I_{leak}^m(\mathbf{v}_2) & \dots & I_{leak}^m(\mathbf{v}_M) \end{bmatrix}^T$$

(b)

$$\begin{bmatrix} 1 & I_{leak,1}^n(\mathbf{v}_1) & I_{leak,2}^n(\mathbf{v}_1) & \dots & I_{leak,N}^n(\mathbf{v}_1) \\ 1 & I_{leak,1}^n(\mathbf{v}_2) & I_{leak,2}^n(\mathbf{v}_2) & \dots & I_{leak,N}^n(\mathbf{v}_2) \\ & & \vdots & & \\ 1 & I_{leak,1}^n(\mathbf{v}_M) & I_{leak,2}^n(\mathbf{v}_M) & & I_{leak,N}^n(\mathbf{v}_M) \end{bmatrix}$$

(c)

Figure. 5.1 The generic forms of (a) the SLE with M equations and N variables, (b) the vector y with M elements, and (c) the M -by- $(N+1)$ Matrix X .

The above steps for calculating θ are repeated K times, with a different random test vector set used at each time. The obtained K sets of θ are then post-processed to extract the HT indicators. For each θ_j ($j = 0, \dots, N$), the K values below the 20th percentile and

above the 80th percentile are curtailed to remove the influence of outliers. The remaining 60% of the K values are then used to calculate the mean and variance of θ_j . For simplicity, such 20th percentile modulated mean and variance are referred to the *modulated mean* and *variance* below, denoted as $\mu(\theta_j)$ and $\text{Var}(\theta_j)$, respectively.

Next, $\mu(\theta_j)$ of each calculated parameters θ_j ($j = 1, \dots, N$) is compared with the corresponding gate's simulated scaling factor s_j that is used in the simulation. θ_j is deemed to be accurately estimated if the fractional difference between $\mu(\theta_j)$ and s_j falls within a small error margin ε (ε is set to be 0.01 or 1% in the experiment), as depicted in Equation (5.7). θ_0 and the accurately estimated θ_j 's then form the HT indicator $\text{Ind}(\theta)$. The mean and variance of each parameter in $\text{Ind}(\theta)$ are recorded.

$$\frac{|\mu(\theta_j) - s_j|}{s_j} < \varepsilon \quad (5.7)$$

In phase II, the K sets of test vectors used in phase I will be injected to the IUA and the total chip leakage current is measured for each set of test vectors. Multiple measurements can be performed for each input vector to reduce the random measurement errors. Based on the SLE formed under each test vector set, the matrix X and vector y can be constructed. Then, the normal equation, Equation (5.6), is used to solve for θ . Next, the same modulated mean and variance of each θ after removing the lower and upper 20th percentiles are calculated. The modulated variance for each θ in $\text{Ind}(\theta)$ of the IUA is compared with the variance recorded in phase I. If a large discrepancy exists, the IUA is considered to be HT-infected. Otherwise, the IUA is deemed to be HT-free. The flow of this HT detection method is depicted in Fig. 5.2.

Figure 5.2 Flow chart of the proposed 2-phase HT detection method.

As an illustration, consider a circuit of seven NAND gates in Fig. 5.3. The simulated scaling factors s_j of each gate j (including the HT gate) obtained from the PV model is listed in Table 5.1(a). A total of $K = 50$ runs of tests are performed and each test vector set consists of 20 randomly generated vectors. Hence, for each test vector set, a 20×8 matrix X of the nominal leakage current and a 20-element vector y of the measured chip leakage current are constructed. By using the normal equation of Equation (5.6), θ can be calculated. The modulated mean values and variances of the calculated θ for the seven gates of the genuine and HT-infected circuits are listed in Table 5.1(b).

As Gates 3 and 7 always have the same inputs (i.e., they are correlated), θ_3 and θ_7 , as red bold-printed in Table 5.1(b), are not good estimates of the corresponding red bold-printed scaling factors in Table 5.1(a). However, this does not influence the correct characterization of the other five gates. We will use the estimated scaling factors of these five gates, i.e., $\theta_1, \theta_2, \theta_4, \theta_5, \theta_6$ as well as θ_0 as the HT indicator. That is, $Ind(\theta) = (\theta_0, \theta_1, \theta_2, \theta_4, \theta_5, \theta_6)$. It can be seen that there exists a large gap between the variance of $Ind(\theta)$ of the HT-free and HT-infected circuits, which provides a clear evidence to distinguish an HT-infected circuit from the genuine circuit.

Figure 5.3 The example circuit of 7 NAND gates with a 1-NAND HT.

Table 5.1

(a) The scaling factor for each gate in the simulated circuit.

	Gate 1	Gate 2	Gate 3	Gate 4	Gate 5	Gate 6	Gate 7	HT gate
s_j	0.83	1.27	0.92	0.70	0.72	1.24	1.50	0.99

(b) The modulated mean and variance of calculated θ_j for the HT-free and HT-infected cases

θ_j	θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7
$\mu(\theta_j)$ (HT-free)	3E-4	0.83	1.27	1.21	0.70	0.72	1.24	1.21
$\mu(\theta_j)$ (HT-infected)	-21	2.25	0.79	1.34	0.59	1.23	1.15	1.35
$\text{Var}(\theta_j)$ (HT-Free)	2E-4	9E-7	7E-7	6E-7	9E-7	1E-6	1E-6	6E-7
$\text{Var}(\theta_j)$ (HT-infected)	41.35	0.18	0.09	0.12	0.15	0.22	0.19	0.12

5.4 EXPERIMENTAL EVALUATIONS

The presented method is tested on the circuits obtained from the ISCAS'89 benchmark suite. Both the HT-free and HT-infected circuits are simulated. The nominal leakage values of the elementary gates are estimated by Cadence SoC Encounter with TSMC 0.18 μm CMOS standard cell library. It should be noted that the successful HT detection rate of the presented method is not dependent on the choice of process technology since it is the relative difference among the leakage powers of the library cells instead of their absolute values that matters most [148]. A total PV of 12% is assumed in the simulations as in [149], where 20% of the PV is the inter-die variation, 60% is the spatially correlated intra-die variation and 20% is the random uncorrelated intra-die variation. The measurement error is modeled using the triangular distribution with mean value of 1% as in [102]. Custom C++ codes were written to randomly generate the test vectors, deduce the final gate state for each vector, and construct the matrix X and y . The estimated parameters θ are obtained with Python codes which implements Equation (5.6) and calculates the modulated mean and variance for $Ind(\theta)$.

The HT used is a 2-input, 1-output NAND gate which is well regarded as the most difficult case for HT detection [145]. It is randomly inserted into the benchmark circuits. For each benchmark, five circuit instances with different PVs are generated; each circuit instance is then tested for both the HT-free and HT-infected cases. The simulation results for the 6-gate C17 are shown in Table 5.2. All the six gates can be accurately characterized in the five HT-free circuit instances. Thus, the HT detection indicator can be constructed as $Ind(\theta) = (\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$. For clarity, only the modulated mean and variance of $\theta_0, \theta_1, \theta_2$ and θ_3 (which are randomly selected from the HT indicator $Ind(\theta)$) for both the HT-free and HT-infected cases of the five circuit instances are listed in the table. The trends in θ_4, θ_5 and θ_6 are similar to those in θ_1, θ_2 and θ_3 .

From Table 5.2, the modulated variance of the estimated θ is consistent among the five HT-free circuits. In contrast, the modulated mean of the estimated θ varies apparently for different circuit instances, which is expected because the same circuit gate usually have different scaling factors in different chips due to the PV. Compared with the absolute (mean) value of the calculated scaling factors, it is verified that the

variance in the calculated scaling factor values among different runs is a better indicator for detecting the HT. The difference between the variances of the HT-free and HT-infected circuits is around four orders of magnitude, which is an obvious gap to distinguish the two cases.

Table 5.2

Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C17 associated with different PV. The HT used is a 2-input NAND gate.

Instance	HT	$\mu(\theta_0)(\text{Var}(\theta_0))$	$\mu(\theta_1)(\text{Var}(\theta_1))$	$\mu(\theta_2)(\text{Var}(\theta_2))$	$\mu(\theta_3)(\text{Var}(\theta_3))$
1	Free	8.7E-5 (3.7E-4)	1.098 (1.6E-6)	1.387 (9.9E-7)	1.047 (2.4E-6)
	Infected	108.9 (5.793)	1.199 (0.025)	0.612 (0.022)	1.117 (0.035)
2	Free	-0.001 (3.7E-4)	1.003 (1.54E-6)	1.152 (9.2E-7)	0.766 (2.3E-6)
	Infected	102.3 (5.451)	1.097 (0.023)	0.425 (0.023)	0.822 (0.032)
3	Free	4.0E-4 (2.7E-4)	0.579 (1.3E-6)	0.872 (1.1E-6)	1.795 (1.2E-6)
	Infected	168.1 (8.962)	0.735 (0.038)	-0.327 (0.038)	1.902 (0.053)
4	Free	-5.0E-4 (4.0E-4)	0.575 (1.3E-6)	0.824 (7.3E-7)	1.598 (2.2E-6)
	Infected	164.37 (8.756)	0.728 (0.037)	-0.342 (0.037)	1.698 (0.052)
5	Free	2.3E-4 (3.3E-4)	1.452 (1.2E-6)	0.774 (1.1E-6)	0.751 (1.8E-6)
	Infected	120.6 (6.426)	1.557 (0.027)	-0.083 (0.027)	0.828 (0.038)

The simulation results for the 160-gate C432 and 202-gate C499 are presented in Table 5.3 and Table 5.4, respectively. For both benchmarks, there exist gate correlations and not all gates' scaling factors can be accurately estimated. $Ind(\theta)$ is constructed with θ_0 and those scaling factors that can be accurately estimated. It can be observed that the variances of the HT-free and HT-infected C432 is different by approximately two orders of magnitude, while the difference in variances of the HT-free and HT-infected cases in C499 reduces to only around an order of magnitude.

As a comparison, Nelson et al. [101] uses the averaged scaling factor values of all gates as an indicator to detect the HT. They reported the results for only C17 and C432. Their average values of the HT-infected cases are at most 11.3% and 3.89% larger than those of the HT-free cases for C17 and C432, respectively. The increase in the averaged scaling factor value due to the HT can be easily concealed by the PV. On the contrary, the difference between the HT-free and HT-infected chips based on the variance of scaling factor values, as shown in the simulation results of the proposed method and summarized in Table 5.5, is far more discriminative.

Table 5.3

Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C432 associated with different PV. The HT used is a 2-input NAND gate.

Instance	HT	$\mu(\theta_0)(\text{Var}(\theta_0))$	$\mu(\theta_{87})(\text{Var}(\theta_{87}))$	$\mu(\theta_{95})(\text{Var}(\theta_{95}))$	$\mu(\theta_{104})(\text{Var}(\theta_{104}))$
1	Free	4.4E-4 (2.9E-8)	0.867 (2.6E-5)	1.359 (3.1E-5)	0.826 (2.4E-5)
	Infected	4.4E-4 (1.4E-5)	0.867 (1.0E-3)	1.364 (1.0E-3)	0.826 (1.0E-3)
2	Free	4.5E-4 (3.3E-8)	1.136 (9.0E-5)	1.052 (4.3E-5)	1.098 (2.2E-5)
	Infected	4.5E-4 (1.8E-5)	1.204 (4.9E-3)	1.053 (4.0E-3)	1.098 (2.0E-3)
3	Free	4.6E-4 (2.9E-8)	0.875 (2.0E-5)	0.925 (2.1E-5)	0.914 (2.6E-5)
	Infected	4.5E-4 (1.5E-5)	0.875 (1.0E-3)	0.925 (2.0E-3)	0.914 (2.0E-3)
4	Free	4.3E-4 (3.0E-8)	1.063 (2.1E-5)	1.135 (1.3E-5)	1.120 (2.2E-5)
	Infected	4.3E-4 (1.7E-5)	1.063 (2.0E-3)	1.135 (3.0E-3)	1.119 (2.0E-3)
5	Free	4.1E-4 (2.3E-8)	0.859 (2.2E-5)	0.950 (1.5E-5)	0.905 (2.2E-5)
	Infected	4.1E-4 (1.4E-5)	0.859 (1.0E-3)	0.950 (2.0E-3)	0.904 (2.0E-3)

Table 5.4

Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C499 associated with different PV. The HT used is a 2-input NAND gate.

Instance	HT	$\mu(\theta_0)(\text{Var}(\theta_0))$	$\mu(\theta_7)(\text{Var}(\theta_7))$	$\mu(\theta_{132})(\text{Var}(\theta_{132}))$	$\mu(\theta_{133})(\text{Var}(\theta_{133}))$
1	Free	-0.070 (0.390)	0.864 (8.3E-6)	0.921 (6.6E-6)	0.897 (7.7E-6)
	Infected	-100.8 (7.171)	0.864 (1.1E-4)	0.920 (9.3E-5)	0.897 (8.0E-5)
2	Free	-0.143 (0.560)	0.859 (7.2E-6)	0.807 (8.3E-6)	0.816 (7.6E-6)
	Infected	-73.3 (5.275)	0.859 (7.7E-5)	0.801 (6.8E-5)	0.812 (6.2E-5)
3	Free	0.042 (0.403)	0.721 (6.7E-6)	0.775 (8.7E-6)	0.745 (9.5E-6)
	Infected	-140.6 (9.722)	0.720 (1.5E-4)	0.776 (1.4E-4)	0.740 (1.1E-4)
4	Free	-0.260 (0.488)	0.978 (6.1E-6)	0.963 (7.2E-6)	0.965 (9.8E-6)
	Infected	-74.35 (5.023)	0.977 (8.2E-5)	0.961 (6.8E-5)	0.962 (6.0E-5)
5	Free	-0.051 (0.482)	0.744 (8.3E-6)	0.736 (5.8E-6)	0.758 (9.0E-6)
	Infected	-106.2 (7.630)	0.743 (1.1E-4)	0.736 (1.0E-4)	0.755 (8.7E-5)

Table 5.5

Comparison of the averaged discrepancy between HT free and infected circuits in [101] and the proposed method.

Circuit	Averaged discrepancy	
	[101]	Proposed
C17	11.3%	$10^4 X$
C432	3.89%	$10^2 X$

It is noted that the gap between the variances of the HT-free and HT-infected cases reduces with increasing circuit size, or rather, decreasing Trojan-to-circuit size ratio (TCR). As the TCR gets smaller, the effect of the HT on the gate scaling factors of the benchmark circuits reduces and hence the inconsistency in the calculated scaling factors (i.e., the variance) of the gates becomes smaller. In fact, the difference is so small between HT-infected and HT-free circuits that the distinction becomes inconclusive for the 383-gate C880a benchmark circuit. This is an indication that the TCR has fallen below the HT detection sensitivity of the proposed method. The TCR in terms of gate count for one NAND-HT in C499 is around 0.49%. To confirm that the proposed method can detect the HT as long as TCR in terms of gate counts exceeds 0.5%, two NAND gates are randomly inserted into C880, where the TCR in terms of gate counts are now 0.52%. The experimental results for a subset of $Ind(\theta)$ are shown in Table 5.6. The HT-infected C880a circuit can be distinguished from the HT-free circuit based on the modulated variance values, where there is a gap of around 10~100 times.

Table 5.6

Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C880a associated with different PV. The HT used is two 2-input NAND gates.

Instance	HT	$\mu(\theta_0)(\text{Var}(\theta_0))$	$\mu(\theta_3)(\text{Var}(\theta_3))$	$\mu(\theta_{191})(\text{Var}(\theta_{191}))$	$\mu(\theta_{292})(\text{Var}(\theta_{292}))$
1	Free	0.014 (3.5E-6)	1.337 (1.8E-5)	1.240 (4.0E-6)	1.503 (2.7E-6)
	Infected	0.019 (1.2E-3)	1.337 (1.2E-3)	1.241 (1.6E-3)	1.503 (1.4E-3)
2	Free	0.016 (1.7E-5)	0.909 (1.8E-5)	1.635 (5.4E-6)	1.069 (1.9E-6)
	Infected	0.018 (5.9E-4)	0.909 (5.9E-3)	1.635 (7.4E-4)	1.069 (6.7E-4)
3	Free	0.013 (8.2E-6)	0.589 (1.5E-5)	1.369 (4.8E-6)	1.020 (2.8E-6)
	Infected	0.016 (6.9E-4)	0.589 (6.9E-3)	1.369 (8.6E-4)	1.020 (8.0E-4)
4	Free	0.014 (1.5E-5)	0.891 (1.3E-5)	1.132 (5.4E-6)	0.589 (2.1E-6)
	Infected	0.015 (4.0E-4)	0.891 (4.0E-3)	1.132 (5.0E-4)	0.589 (4.5E-4)
5	Free	0.014 (1.8E-6)	1.386 (2.2E-5)	0.758 (4.6E-6)	1.361 (2.7E-6)
	Infected	0.017 (7.6E-4)	1.386 (7.7E-3)	0.758 (9.6E-4)	1.361 (8.6E-4)

It should be noted that the proposed method is unable to detect the HT gate if it is inserted in such a way that it receives exactly the same inputs as an existing circuit gate of the same type, i.e., it is fully correlated with an existing circuit gate. In this case, the scaling factor of the HT gate will be merged with that of the circuit gate and the effect of the HT will only be shown on the scaling factor of the merged circuit

gate. This case is investigated by inserting a 1-NAND HT that is perfectly correlated with gate 5 (also a NAND gate) in C17. The simulation results are shown in Table 5.7. For the HT-free case, all the six circuit gates can be accurately characterized, and for the HT-infected case, all gates except gate 5 can also be accurately characterized. For clarity, only the modulated mean and variance of θ_0 , θ_1 , θ_2 and θ_5 are listed. The trend in θ_3 , θ_4 , and θ_6 are very similar to that in θ_1 and θ_2 .

Table 5.7

Modulated mean and variance of randomly selected elements in $Ind(\theta)$ from the five instances of C17 associated with different PV for correlated HT case. The HT gate is inserted to receive the same inputs as gate 5 of the circuit.

Instance	HT	$\mu(\theta_0)(\text{Var}(\theta_0))$	$\mu(\theta_1)(\text{Var}(\theta_1))$	$\mu(\theta_2)(\text{Var}(\theta_2))$	$\mu(\theta_5)(\text{Var}(\theta_5))$
1	Free	3.0E-4 (3.0E-4)	0.663 (9.8E-7)	1.019 (9.1E-7)	0.902 (1.0E-6)
	Infected	-8.0E-5 (3.0E-4)	0.663 (1.9E-6)	1.019 (1.4E-6)	2.183 (2.0E-6)
2	Free	3.0E-4 (4.0E-4)	0.619 (2.0E-6)	0.731 (1.1E-6)	1.060 (2.1E-6)
	Infected	3.0E-4 (3.0E-4)	0.611 (1.5E-6)	0.731 (9.3E-7)	1.701 (2.1E-6)
3	Free	-6.0E-4 (3.0E-4)	1.464 (1.2E-6)	1.010 (9.3E-7)	0.710 (2.4E-6)
	Infected	-8.0E-4 (3.0E-4)	1.464 (9.4E-7)	1.012 (8.1E-7)	1.734 (1.6E-6)
4	Free	-6.7E-4 (2.7E-4)	0.669 (1.7E-6)	0.528 (1.1E-6)	0.733 (2.1E-6)
	Infected	2.1E-4 (3.0E-4)	0.669 (1.5E-6)	0.528 (1.2E-6)	1.463 (1.8E-6)
5	Free	-1.8E-4 (4.4E-4)	0.907 (1.1E-6)	0.918 (1.4E-6)	0.930 (2.3E-6)
	Infected	-4.6E-4 (4.6E-4)	0.907 (1.4E-6)	0.918 (1.2E-6)	1.976 (2.7E-6)

From Table 5.7, the difference in the modulated variances between the HT-free and HT-infected cases is very small. It is incapable of distinguishing the two cases. The modulated mean of calculated scaling factor for gate 5, $\mu(\theta_5)$, is consistently larger in the HT-infected case than in the HT-free case. This is because the scaling factor of the HT gate is merged into that of gate 5.

In reality, functional HT is usually much more complex than a single NAND gate. The HT payload fulfills a specific function upon activation, which is usually implemented in the form of a combinational or sequential circuit. It is very difficult to insert a functional HT in such a way that all its gates are perfectly correlated with the existing circuit gates. Perfectly correlated gate is just a harmless duplication of an existing gate which can be easily identified in the design tool flow if it is inserted prior to tape-out. Hence, the proposed method is expected to be able to detect most of the functional HT with a TCR in terms of gate counts of larger than 0.5%.

5.5 CHAPTER SUMMARY

The HT detection method presented in this chapter has efficiently solved the SLE formed by the GLC process. The scaling factors are estimated with linear regression. By using the bias term of the linear regression and a subset of accurately estimated scaling factors, the method is able to detect the presence of a wide range of functional HTs without the need for a golden reference nor the need to trigger the HTs as long as the TCR in terms of gate counts exceeds 0.5%. The approach is an excellent complement to the logic testing approach which is effective in triggering ultra-small HTs. Without having to solve the GLC completely for every IUA at individual gate level, the time and computational cost required by the presented method is much lower than the conventional GLC approaches. A possible extension of this method to further improve the HT detection sensitivity is presented in the future work of Chapter 6.

Chapter 6

Conclusions and Future Work

6.1 CONCLUSIONS

Since its inception, reuse based design methodology has bolstered the development of SoC design with phenomenal success. At the same time, the rising commercial values of hardware IP blocks and the ease of their accesses under the current IP sharing and licensing platform opens out the door to various IP infringement attacks such as IP misuse and IP tampering. The flagrant breach of IP laws has reaches an alarming scale, which lends credence to the view that the protection from legal means such as copyright law and patent protection act are feeble and need to be fortified by technology-enabled IP protection measures. In fact, as early as 1997, the VSI Alliance had identified IP protection as one of the six key enabling technologies for SoC development [10]. Technology-based forensic techniques would help to foster a more secure IP transaction environment than legal enforcement alone. The scope of the trusted framework should ideally provide the IP vendors with the confidence that their IP cores cannot be easily abused and no design information will be divulged or maliciously manipulated and give the IP users the assurance that their received IP cores are authentic and uncompromised. A secure and trusted IP transaction environment will encourage more IP cores from different vendors to be developed and sold in the IP market to drive the prices down under the fair competition framework and boost the development of multifarious IP cores. The abundant choices of IP cores by the system developer will in turn create a virtuous circle to maximize the productivity gain from the reuse based design methodology.

Three methods to enhance the hardware design security have been presented in this thesis. They help to protect and verify the integrity and authenticity of the IP cores

from three facets of application, i.e., identification of the IP owner and IP buyer, active control of IP usage, and integrity verification of IP content.

Unlike existing schemes which require cumbersome and costly fingerprint verification by checking if the additional fingerprinting constraints are satisfied, the fingerprinting method presented in this thesis makes detection of fingerprint convenient and in situ. In fact, this is the first dynamic and blind fingerprinting scheme for which the embedded fingerprint carrying a concealed digital signature of the IP owner endorsed by the IP buyer can be detected directly from the output sequence by injecting a specific input sequence. As the fingerprint is conveyed by a seamless integration of test machines into the IP core, the identity of the IP owner and the IP buyer can be authenticated in the field even after the IP core is integrated and hidden in a packaged chip. Due to the use of a single blind fingerprint, state variable chaining heuristic and dependency based partitioning algorithm, the scheme is capable of generating a large number of high quality fingerprinted instances from an originally optimized seed design of the IP core with incremental design efforts. The inserted fingerprint is also shown to be robust against various perceivable attacks.

The presented per-device licensing scheme for FPGA based IP cores facilitates the secure transaction of the protected IP cores and enables the IP vendor to have a control over the configuration of the IP cores on the devices. It effectively protects the IP core against various common attacks such as IP cloning, reverse engineering and tampering. The licensing protocol blends well the IP fingerprinting techniques to allow the source of IP infringement (e.g., the IP buyer who misuses his IP instance) to be traced. The scheme can be directly applied to existing FPGA devices without requiring any modifications and incurs low implementation overheads even after the addition of countermeasures against common physical attacks. Without requiring an external TTP, the scheme can be applied to replace the current upfront IP licensing model in the IP market by the more attractive pay-per-use IP licensing model for FPGA IPs.

The presented HT detection method is able to efficiently check the integrity of the fabricated chip and verify if an extraneous component has been inserted or not. The method does not need a golden model which is very expensive to obtain. It needs only to accurately characterize the gate-level leakage currents of a subset of circuit gates.

The gate-level characterization problem is solved by the normal equation in linear regression. By using the bias term of the linear regression and the subset of accurately estimated leakage current scaling factors, the method is able to detect the presence of a wide range of HTs without the need to trigger the HTs as long as the HT-to-circuit ratio exceeds a minimum threshold.

In summary, the methods developed in this thesis contribute to fortifying the current IP protection schemes towards establishing a trusted IP transaction platform. IP protection is a research field that requires continuous efforts. There will always be new types of IP infringement emerging, upon which new IP protection techniques would be necessary. The techniques developed in this thesis aim to strike a good balance between the provided security level and the incurred cost. An IP protection technique that makes the IP core immune to any attacks is not a judicious target, as it can impose too high a cost to be receptive for market adoption. Instead, a practical technique should be the one that makes a successful attack too expensive to be compensated by the benefits obtained from the stolen IP cores; at the same time, the implementation cost for the scheme should be market viable.

6.2 RECOMMENDED FUTURE WORK

Based on the research accomplished thus far, the following ideas are recommended for further exploration.

6.2.1 Fragile IP tagging for HT detection

Most of the IP watermarking and fingerprinting techniques discussed in this thesis aim to prove IP misappropriation by identifying the signatures of the IP owner and IP buyer. There is a strong emphasis on the robustness of the embedded tags (watermark or fingerprint) by preventing the tags from being removed without altering the IP functionality. Such techniques can be termed as robust IP tagging. In fact, the inserted tag can also be made extremely sensitive to design changes and be destroyed as soon as the IP core is tampered with, forming the fragile tag. These fragile tags will be very helpful to prevent and detect the insertion of the HT. A small change to the tagged IP core may render the inserted tag unrecoverable, which can be used as an alarm of HT

infection. The resultant fragile IP tagging technique can be an effective design-for-hardware-trust technique introduced in Section 2.4 of Chapter 2.

The fragile IP tag may also be partially recoverable from the HT-infected IP core, with the unrecoverable part of the tag corresponding to the circuit region that the HT resides. This way, localization of the HT may be achieved. One of the obstacles for detecting/diagnosing the HT is the ultra-small Trojan-to-circuit ratio (TCR). The localization of HT will direct the HT diagnosis to the suspected region, increase the TCR, and hence produce an accurate HT diagnosis. For example, the watermarking scheme proposed by Cui and Chang [31] poses some features of fragile watermarking. If the inserted watermark can cover the entire circuit or at least the critical part of the circuit, the IP integrity can be verified based on whether all the watermark bits can be recovered. The unrecoverable bits correspond to the possibly tampered regions that contain the HT.

6.2.2 FPGA IP licensing schemes based on public-PUF

The presented FPGA IP licensing scheme is designed based on the conventional crypto primitives, although PUF primitive can be used to replace the hardwired device ID and on-chip secret key. Most existing PUF-based licensing schemes take advantage of the large number of challenge-response pairs (CRPs) possessed by the (strong) PUF and are very different from those based on conventional crypto primitives.

For the PUF-based IP licensing schemes, a common assumption is that during the IP licensing stage and thereafter, the PUF response R is only available internally to the on-chip decryption and authentication circuit. However, recent research results on PUFs may subvert this assumption. For example, numerous strong PUFs with a public model, denoted as PPUF, have been proposed [150-154]. A typical form of PPUF exploits the time gap of generating the response to a challenge between the direct execution with PUF and the simulation with public model. A radically different communication protocol for secure IP transaction may emerge as any party can now obtain the CRP response using the publicly published simulation model. The PPUF-based secret key exchange protocol [150] and public-key communication protocol

[154] have already been developed. If the licensing scheme of FPGA IP cores can leverage on such PPUF based protocols, there will be no need to have authenticated and secure communication channels and the secure key establishment can be achieved without the high implementation cost of conventional public-key crypto such as ECC and RSA.

6.2.3 HT detection with segmentation techniques

The presented HT detection method is characterized by its fast detection of possible HT insertion into a fabricated chip. However, the method has a limited HT detection sensitivity of TCR above 0.5%. The limited sensitivity can be overcome by combining the method with a circuit segmentation approach. As the TCR for each circuit segment is increased, the HT detection problem can be solved more efficiently by a divide-and-conquer manner. The presented HT detection method can be applied to each small segment independently, and as a result the HT detection sensitivity can be significantly improved.

The circuit segmentation approach proposed in [105] may be adopted. It is based on input vector control, where only the selected input signals are varied while the rest of the input signals are frozen. Consequently, only the gates controlled by the varying inputs will change the states and their nominal leakage values (i.e., coefficients) in the system of linear equations (SLE), whereas all the other gates will have identical coefficients in all the equations. These frozen gates can be represented by a single variable in the SLE. This way, the circuit can be virtually divided into a set of small segments. The SLE of each segment can then be processed and analyzed by the presented HT detection method.

AUTHOR'S PUBLICATION

Book Chapters

- [1]. **L. Zhang**, and C. H. Chang, "Secure Licensing of IP Cores on SRAM-based FPGAs", to appear in "Secure System Design and Trustable Computing", ed. by C. H. Chang and M. Potkonjak, Springer, 2015.
- [2]. C. H. Chang, M. Potkonjak, and **L. Zhang**, "Hardware IP Watermarking", to appear in "Secure System Design and Trustable Computing", ed. by C. H. Chang and M. Potkonjak, Springer, 2015.

Journal Papers

- [1]. **L. Zhang**, and C. H. Chang, "A pragmatic per-device licensing scheme for hardware IP cores on SRAM based FPGAs," *IEEE Trans. Inform. Forensics and Security*, vol. 9, no. 11, pp. 1893-1905, Sept. 2014.
- [2]. C. H. Chang, and **L. Zhang**, "A blind dynamic fingerprinting technique for sequential circuit intellectual property protection," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 1, pp. 76-89, Jan. 2014.

Conference Papers

- [1]. L. Zhang, and C. H. Chang, "Public key protocol for usage-based licensing of FPGA IP cores," in *Proc. IEEE International Symposium on Circuits and Systems*, May 2015, Lisbon, Portugal, accepted.
- [2]. W. Wei, **L. Zhang** and C. H. Chang, "A modular design of elliptic-curve point multiplication for resource constrained devices", in *Proc. International Symposium on Integrated Circuits*, Singapore, December 2014, pp. 596-599.
- [3]. **L. Zhang**, and C. H. Chang, "Hardware Trojan Detection with Linear Regression Based Gate-Level Characterization," in *Proc. IEEE Asia Pacific Conf. Circuits and Systems*, Okinawa, Japan, November 2014, pp. 256-259.
- [4]. **L. Zhang**, and C. H. Chang, "State encoding watermarking for field authentication of sequential circuit intellectual property," in *Proc. IEEE*

- International Symposium on Circuits and Systems*, Seoul, Korea, May 2012, pp. 3013-3016.
- [5]. A. J. Cui, C. H. Chang, and **L. Zhang**, "A hybrid watermarking scheme for sequential functions", in *Proc. IEEE International Symposium on Circuits and Systems*, Rio de Janeiro, Brazil, May 2011, pp. 2333-2336.

BIBLIOGRAPHY

- [1] M. Santarini. (2014). Xilinx Ships Industry's First 20-nm All Programmable Devices. Xilinx Inc. [Online]. Available:
http://www.xilinx.com/support/documentation/xcell_articles/xcell86-xilinx-ships-industry-first-20nm.pdf.
- [2] Market research report (March 2014). Semiconductor (Silicon) IP Market by Form Factor, Design Architecture, Processor Type, Application, Geography and Verification IP - Forecast & Analysis to 2013 - 2020. MarketsandMarkets. [Online]. Available:
<http://www.marketsandmarkets.com/PressReleases/semiconductor-ip.asp>.
- [3] VSI Alliance, "VSI Alliance Architecture Document: Version 1.0," 1997.
- [4] P. Clarke. (April 2012). Spreadtrum, Dialog, MegaChips shine in fabless rankings. *EE Times*. [Online]. Available:
http://www.eetimes.com/document.asp?doc_id=1261538.
- [5] L. Yuan, G. Qu, L. Ghouti, and A. Bouridane, "VLSI design IP protection: Solutions, new challenges, and opportunities," in *Proc. NASA/ESA Conference on Adaptive Hardware and Systems*, Istanbul, Turkey, June 2006, pp. 469-476.
- [6] VSI Alliance, "Intellectual property protection: Schemes, alternatives and discussion," White paper, Version 1.1, August 2001.
- [7] I. J. Cox, M. L. Miller, and J. A. Bloom, *Digital Watermarking*. San Diego, CA: Academic Press, 2001.
- [8] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "A survey on IP watermarking techniques," *Design Automation for Embedded Systems*, vol. 9, no. 3, pp. 211-227, September 2004.
- [9] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "IP watermarking techniques: Survey and comparison," in *Proc. IEEE Int. Workshop on System-on-Chip for Real-Time Applications*, Alberta, Canada, June - July 2003, pp. 60-65.
- [10] VSI Alliance, "Fall Worldwide member meeting: a year of achievement," Santa Clara, CA, USA, October 1997.

-
- [11] Kerckhoffs, "A. La Cryptographie Militaire.," *Journal des sciences militaires*, vol. IX, pp. 5–38, January 1883.
 - [12] Y. C. Fan, "Testing-based watermarking techniques for intellectual-property identification in SOC design," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 3, pp. 467-479, March 2008.
 - [13] T. Kean, D. McLaren, and C. Marsh, "Verifying the authenticity of chip designs with the DesignTag system," in *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*, Anaheim, USA, June 2008, pp. 59-64.
 - [14] D. Ziener and J. Teich, "Power signature watermarking of IP cores for FPGAs," *Journal of Signal Processing Systems for Signal Image and Video Technology*, vol. 51, pp. 123-136, April 2008.
 - [15] G. T. Becker, M. Kasper, A. Moradi, and C. Paar, "Side-channel based watermarks for integrated circuits," in *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*, Anaheim, USA, June 2008, pp. 30-35.
 - [16] J. L. Wong, G. Qu, and M. Potkonjak, "Optimization-intensive watermarking techniques for decision problems," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 1, pp. 119-127, January 2004.
 - [17] I. Hong and M. Potkonjak, "Techniques for intellectual property protection of DSP designs," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Washington, USA, May 1998, pp. 3133-3136.
 - [18] A. B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, W. Huijuan, and G. Wolfe, "Robust IP watermarking methodologies for physical design," in *Proc. Design Automation Conference*, CA, USA, June 1998, pp. 782-787.
 - [19] A. B. Kahng, D. Kirovski, S. Mantik, M. Potkonjak, and J. L. Wong, "Copy detection for intellectual property protection of VLSI designs," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, CA USA, November 1999, pp. 600-604.
 - [20] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. J. Wang, and G. Wolfe, "Constraint-based watermarking techniques for design IP protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1236-1252, October 2001.

-
- [21] G. Qu and M. Potkonjak, "Hiding signatures in graph coloring solutions," in *Proc. Int. Workshop on Information Hiding*, Dresden, Germany, September 1999, pp. 348-367.
 - [22] R. Chapman and T. S. Durrani, "IP protection of DSP algorithms for system on chip implementation," *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 854-861, March 2000.
 - [23] S. Megerian, M. Drinic, and M. Potkonjak, "Watermarking integer linear programming solutions," in *Proc. Design Automation Conference*, New Orleans, LA, USA, June 10-14, 2002, pp. 8-13.
 - [24] G. Wolfe, J. L. Wong, and M. Potkonjak, "Watermarking graph partitioning solutions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 10, pp. 1196-1204, October 2002.
 - [25] D. Kirovski and M. Potkonjak, "Local watermarks: Methodology and application to behavioral synthesis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 9, pp. 1277-1284, Sept 2003.
 - [26] F. Koushanfar, I. K. Hong, and M. Potkonjak, "Behavioral synthesis techniques for intellectual property protection," *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, no. 3, pp. 523-545, July 2005.
 - [27] E. Charbon and I. Torunoglu, "Watermarking techniques for electronic circuit design," in *Proc. Int. Workshop on Digital Watermarking*, Seoul, Korea, November 2002, pp. 147-169.
 - [28] D. Kirovski, Y. Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis solutions," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, San Jose, California, USA, November 1998, pp. 194-198.
 - [29] D. Kirovski, Y. Y. Hwang, M. Potkonjak, and J. Cong, "Protecting combinational logic synthesis solutions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2687-2696, December 2006.
 - [30] M. M. Khan and S. Tragoudas, "Rewiring for watermarking digital circuit netlists," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1132-1137, July 2005.

- [31] A. J. Cui, C. H. Chang, and S. Tahar, "IP watermarking using incremental technology mapping at logic synthesis level," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1565-1570, September 2008.
- [32] E. Charbon, and I. Torunoglu, "Watermarking layout topologies," in *Proc. ASP-DAC*, Hong Kong, China, January 1999, pp. 213-216.
- [33] N. Narayan, R. D. Newbould, J. D. Carothers, J. J. Rodriguez, and W. T. Holman, "IP protection for VLSI designs via watermarking of routes," in *Proc. IEEE International ASIC/SOC Conference*, 2001, pp. 406-410.
- [34] M. Potkonjak, C. H. Chang, L. Zhang, "Hardware IP Watermarking," to appera in "Secure System Design and Trustable Computing", ed. by C. H. Chang and M. Potkonjak, Springer, 2015.
- [35] G. Qu "Publicly Detectable Watermarking for Intellectual Property Authentication in VLSI Design", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 21, No. 11, pp. 1363-1368, Nov. 2002.
- [36] I. Torunoglu, and E. Charbon, "Watermarking-based copyright protection of sequential functions," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 3, pp. 434-440, March 2000.
- [37] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "A public-key watermarking technique for IP designs," in *Proc. Design, Automation and Test in Europe*, vol. 1, Munich, Germany, March 2005, pp. 330-335.
- [38] A. Cui, C. H. Chang, S. Tahar, and A. T. Abdel-Hamid, "A Robust FSM Watermarking Scheme for IP Protection of Sequential Circuit Design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 5, pp. 678-690, May 2011.
- [39] A. L. Oliveira, "Techniques for the creation of digital watermarks in sequential circuit designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1101-1117, September 2001.
- [40] D. Kirovski, and M. Potkonjak, "Intellectual property protection using watermarking partial scan chains for sequential logic test generation," in *Proc. IEEE High Level Design, Verification, and Test*, CA, USA, November 1998.

- [41] A. Cui, and C. H. Chang, "Intellectual property authentication by watermarking scan chain in design-for-testability flow," in *Proc. IEEE Int. Symposium on Circuits and Systems*, Seattle, Washington, USA, May 2008, pp. 2645-2648.
- [42] A. Cui, C. H. Chang, and L. Zhang, "A Hybrid Watermarking Scheme for Sequential Functions," in *Proc. IEEE Int. Symp. on Circuits and Syst.*, Rio de Janeiro, Brazil, May 2011, pp. 2333-2336.
- [43] C. H. Chang, and A. J. Cui, "Synthesis-for-Testability Watermarking for Field Authentication of VLSI Intellectual Property," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 8, pp. 1618-1630, July 2010.
- [44] A. E. Caldwell, C. Hyun-Jin, A. B. Kahng, S. Mantik, M. Potkonjak, Q. Gang, and J. L. Wong, "Effective iterative techniques for fingerprinting design IP," in *Proc. Design Automation Conference*, New Orleans, LA, USA, June 1999, pp. 843-848.
- [45] A. E. Caldwell, H. J. Choi, A. B. Kahng, S. Mantik, M. Potkonjak, G. Qu, and J. L. Wong, "Effective iterative techniques for fingerprinting design IP," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 2, pp. 208-215, February 2004.
- [46] G. Qu, and M. Potkonjak, "Fingerprinting intellectual property using constraint-addition," in *Proc. Design Automation Conference*, CA, USA, June 2000, pp. 587-592.
- [47] H. J. Patel, J. W. Crouch, Y. C. Kim and T. C. Kim, "Creating a unique digital fingerprint using existing combinational logic," in *Proc. International Symposium on Circuits and Systems*, Taipei, May 2009, pp. 2693-2696.
- [48] Y. Jin, and Y. Makris, "Hardware Trojan detection using path delay fingerprint," in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust*, Anaheim, CA, USA, June 2008, pp. 51-57.
- [49] C. Dunbar and G. Qu, "A practical circuit Fingerprinting method utilizing observability don't-care conditions", in *Proc. International Workshop on Logic and Synthesis*, pp. 1-8, June 2014.
- [50] C. Dunbar and G. Qu, "Satisfiability Don't Care Condition based Circuit Fingerprinting", in *Proc. Asia and South Pacific Des. Auto. Conf.*, Jan. 2015.

-
- [51] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "FPGA fingerprinting techniques for protecting intellectual property," in *Proc. IEEE Custom Integrated Circuits Conference*, CA, USA, May 1998, pp. 299-302.
- [52] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting techniques for field-programmable gate array intellectual property protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1253-1261, October 2001.
- [53] C. H. Chang, and L. Zhang, "A Blind Dynamic Fingerprinting Technique for Sequential Circuit Intellectual Property Protection," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 1, pp. 76-89, January 2014.
- [54] Transparency Market Research. (Oct 2013). *Field-Programmable Gate Array (FPGA) market - global industry analysis, size, share, growth, trends and forecast, 2013-2019*. [Online]. Available: [http://twileshare.com/uploads/Field-Programmable_Gate_Array_\(FPGA\)_Market_-_Global_Industry_Analysis,_Size,_Share,_Growth,_Trends_and_Forecast,_2013_-_20191.pdf](http://twileshare.com/uploads/Field-Programmable_Gate_Array_(FPGA)_Market_-_Global_Industry_Analysis,_Size,_Share,_Growth,_Trends_and_Forecast,_2013_-_20191.pdf).
- [55] E. Simpson, and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," in *Proc. Workshop on Cryptographic Hardware and Embedded Systems*, Yokohama, Japan, October 2006, pp. 311-323.
- [56] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Proc. Workshop on Cryptographic Hardware and Embedded Systems*, Vienna, Austria, Sept 2007, pp. 63-80.
- [57] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "Physical Unclonable Functions and Public-Key Crypto for FPGA IP Protection," in *Proc. International Conference on Field Programmable Logic and Applications*, Amsterdam, The Netherlands, August 2007, pp. 189-195.
- [58] C. H. Chang, L. Zhang, "Secure licensing of IP cores on SRAM based FPGAs," to appear in "Secure System Design and Trustable Computing", ed. by C. H. Chang and M. Potkonjak, Springer, 2015.

-
- [59] J.-B. Note, and E. Rannaud, "From the bitstream to the netlist," in *Proc. International ACM/SIGDA symposium on Field programmable gate arrays*, Monterey, California, USA, February 2008, pp. 264-264.
- [60] F. Benz, A. Seffrin, and S. A. Huss, "Bil: A tool-chain for bitstream reverse-engineering," in *Proc. Int. Conf. Field Programmable Logic and Applications*, Oslo, The Netherlands, 2012, pp. 735-738.
- [61] "FPGA Design Security Issues: Using the ispXPGA Family of FPGAs to Achieve High Design Security," Lattice white paper, December 2003.
- [62] M. Tehranipoor, and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10-25, February 2010.
- [63] S. McNeil, "Solving today's design security concerns (v1.2)," Xilinx white paper 365, July 2012.
- [64] F. Khelil, M. Hamdi, S. Guilley, J. L. Danger, and N. Selmane, "Fault analysis attack on an fpga aes implementation," in *Proc. New Technologies, Mobility and Security*, Tangier, November 2008, pp. 1-5.
- [65] D. M. Dhiman Saha, Dipanwita Roy Chowdhury, "A diagonal fault attack on the advanced encryption standard," *Cryptology ePrint Archive* 2009: 581 (2009).
- [66] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RAS, DSS, and other systems," in *Proc. Int. Cryptology Conf. Advances in Cryptology (CRYPTO)*, Santa Barbara, CA, USA, September 1996, pp. 104-113.
- [67] J. J. Paul Kocher, and Benjamin Jun, "Differential power analysis," in *Proc. Int. Cryptology Conf. Advances in Cryptology (CRYPTO)*, Santa Barbara, California, USA, August 1999, pp. 388-397.
- [68] E. De Mulder, P. Buysschaert, S. B. Ors, P. Delmotte, B. Preneel, G. Vandenbosch, and I. Verbauwhede, "Electromagnetic Analysis Attack on an FPGA Implementation of an Elliptic Curve Cryptosystem," in *Proc. Int. Conf. Computer as a Tool*, Belgrade, November 2005, pp. 1879-1882.
- [69] D. G. Abraham, G. M. Dolan, G. P. Double, and J. V. Stevens, "Transaction Security System," *IBM Systems Journal*, vol. 30, no. 2, pp. 206-229, 1991.

-
- [70] L. Bossuet, G. Gogniat, and W. Burleson, "Dynamically configurable security for SRAM FPGA bitstreams," *Int.J. of Engineering Science*, vol. 2, no. 1/2, pp. 73-85, 2006.
- [71] "Security Solutions Using Spartan-3 Generation FPGAs (v1.1)," Xilinx white paper 266, April 2008.
- [72] NIST, U.S. Dept. of Commerce: FIPS 197: Advanced encryption standard. (2001).
- [73] H. P. Martin Stigge, Wolf Müller, Jens-Peter Redlich, "Reversing CRC – Theory and Practice," Technical Report SAR-PR-2006-05, Humboldt University Berlin, May 2006.
- [74] NIST, U.S. Dept. of Commerce: FIPS 180-3, Secure Hash Standard (SHS). (2008).
- [75] E. Peterson, "Developing tamper resistant designs with Xilinx Vertex-6 and 7 series FPGAs," Xilinx application note, August 10, 2012.
- [76] W. Adi, R. Ernst, B. Soudan, and A. Hanoun, "VLSI design exchange with intellectual property protection in FPGA environment using both secret and public-key cryptography," in *Proc. IEEE Computer Society Annual Symposium on VLSI*, Karlsruhe, March 2006, pp. 24-29.
- [77] S. Malipatlolla, and S. A. Huss, "A novel method for secure intellectual property deployment in embedded systems," in *Proc. VII Southern Conference on Programmable Logic (SPL)*, Cordoba, April 2011, pp. 203-208.
- [78] T. Kean, "Cryptographic rights management of FPGA intellectual property cores," In *Proc. ACM/SIGMA Symp. Field Programmable Gate Arrays*, CA. USA, February 2002, pp. 113-118.
- [79] S. Drimer, T. Guneyusu, M. G. Kuhn, and C. Paar. (2008). *Protecting multiple cores in a single FPGA design*. [Online]. Available: www.saardrimer.com/sd410/papers/protect_many_cores.pdf
- [80] R. Maes, D. Schellekens, and I. Verbauwhede, "A pay-per-use licensing scheme for hardware ip cores in recent SRAM-based FPGAs," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 98-108, February 2012.

- [81] L. Gaspar, V. Fischer, T. Guneyusu, and Z. Cherif Jouini, "Two IP protection schemes for multi-fpga systems," in *Proc. International Conference on Reconfigurable Computing and FPGAs*, Cancun, Mexico, Dec 2012, pp. 1-6.
- [82] T. Kean, "Method of protecting intellectual property cores on field programmable gate array," U.S. Patents 20020199110 A1, 2002.
- [83] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards trojan-free trusted ICs: Problem analysis and detection scheme," in *Proc. Design, Automation and Test in Europe*, Munich, March 2008, pp. 1362-1365.
- [84] X. X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust*, Anaheim, CA, USA, June 2008, pp. 15-19.
- [85] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39-46, October 2010.
- [86] Y. Alkabani, and F. Koushanfar, "Designer's hardware Trojan horse," in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust*, Anaheim, CA, USA, June 2008, pp. 82-83.
- [87] Y. Jin, N. Kupp, and Y. Makris, "Experiences in hardware trojan design and implementation," in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust*, Francisco, CA, USA, July 2009, pp. 50-57.
- [88] X. H. Zhang, and M. Tehranipoor, "Case study: Detecting hardware Trojans in third-party digital IP cores," in *Proc. IEEE Int. Symposium on Hardware-Oriented Security and Trust*, San Diego, CA, USA, June 2011, pp. 67-70.
- [89] M. Tehranipoor, H. Salmani, X. H. Zhang, "Hardware trojan detection: Untrusted third-party IP cores," in *Integrated Circuit Authentication: Hardware Trojans and Counterfeit Detection*, ed. by M. Tehranipoor, H. Salmani, X. H. Zhang, Springer, 2014, pp. 19-30.
- [90] S. Jha, and S. K. Jha, "Randomization based probabilistic approach to detect trojan circuits," in *Proc. IEEE High Assurance Systems Engineering Symposium*, Nanjing, China, December 2008, pp. 117-124.

-
- [91] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "MERO: A statistical approach for hardware trojan detection," in *Proc. Cryptographic Hardware and Embedded Systems*, Lausanne, Switzerland, September 2009, pp. 396-410.
- [92] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proc. IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, May 2007, pp. 296-310.
- [93] H. V. Poor, *An introduction to signal detection and estimation*, 2nd ed. New York: Springer-Verlag, 1994.
- [94] L. Jie, and J. Lach, "At-speed delay characterization for IC authentication and Trojan Horse detection," in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust*, Anaheim, CA, USA, June 2008, pp. 8-14.
- [95] W. Xiaoxiao, H. Salmani, M. Tehranipoor, and J. Plusquellic, "Hardware trojan detection and isolation using current integration and localized current analysis," in *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, Boston, MA, USA, October 2008, pp. 87-95.
- [96] R. M. Rad, W. Xiaoxiao, M. Tehranipoor, and J. Plusquellic, "Power supply signal calibration techniques for improving detection resolution to hardware Trojans," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, USA, November 2008, pp. 632-639.
- [97] M. Banga, M. Chandrasekar, L. Fang, and M. S. Hsiao, "Guided test generation for isolation and detection of embedded trojans in ICs," in *Proc. ACM Great Lakes Symp. on VLSI*, Orlando, Florida, USA, 2008, pp. 363-366.
- [98] M. Banga, and M. S. Hsiao, "A region based approach for the identification of hardware Trojans," in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust*, Anaheim, CA, USA, June 2008, pp. 40-47.
- [99] M. Banga, and M. S. Hsiao, "A novel sustained vector technique for the detection of hardware trojans," in *Proc. International Conference on VLSI Design*, New Delhi, India, January 2009, pp. 327-332.
- [100] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware trojan horse detection using gate-level characterization," in *Proc. ACM/IEEE Design Automation Conf.*, San Francisco, CA, USA, July 2009, pp. 688-693.

-
- [101] M. Nelson, A. Nahapetian, F. Koushanfar, and M. Potkonjak, "SVD-based ghost circuitry detection," in *Proc. International Workshop on Information Hiding*, Darmstadt, Germany, June 2009, pp. 221-234.
- [102] S. Wei, S. Meguerdichian, and M. Potkonjak, "Malicious Circuitry detection using thermal conditioning," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1136-1145, September 2011.
- [103] S. Wei, and M. Potkonjak, "Scalable hardware trojan diagnosis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1049-1057, June 2012.
- [104] Y. Alkabani, and F. Koushanfar, "Consistency-based characterization for IC Trojan detection," in *Proc. IEEE/ACM Int. Conference on Computer-Aided Design*, San Jose, CA, USA, November 2009, pp. 123-127.
- [105] S. Wei, and M. Potkonjak, "Scalable consistency-based hardware trojan detection and diagnosis," in *Proc. International Conference on Network and System Security*, Milan, Italy, September 2011, pp. 176-183.
- [106] S. Wei, and M. Potkonjak, "Self-consistency and consistency-based detection and diagnosis of malicious circuitry," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1, September 2013.
- [107] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. S. Hsiao, J. Plusquellic, and M. Tehranipoor, "Protection against hardware trojan attacks: towards a comprehensive solution," *IEEE Design & Test*, vol. 30, no. 3, pp. 6-17, March 2013.
- [108] S. Narasimhan, D. D. Du, R. S. Chakraborty, S. Paul, F. G. Wolff, C. A. Papachristou, K. Roy, and S. Bhunia, "Hardware trojan detection by multiple-parameter side-channel analysis," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2183-2195, November 2013.
- [109] H. Salmani, M. Tehranipoor, and J. Plusquellic, "New Design Strategy for Improving Hardware Trojan Detection and Reducing Trojan Activation Time," in *Proc. Ieee International Workshop on Hardware-Oriented Security and Trust*, Francisco, CA, USA, July 2009, pp. 66-73.

-
- [110] V. Agrawal, and K.-T. Cheng, "Finite state machine synthesis with embedded test function," *Journal of Electronic Testing*, vol. 1, no. 3, pp. 221-228, October 1990.
- [111] S. Kanjilal, S. T. Chakradhar, and V. D. Agrawal, "A test function architecture for interconnected finite state machines," in *Proc. International Conference on VLSI Design*, Kolkata, India, January 1994, pp. 113-116.
- [112] S. Kanjilal, S. T. Chakradhar, and V. D. Agrawal, "Test function embedding algorithms with application to interconnected finite-state machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 9, pp. 1115-1127, September 1995.
- [113] S. Kanjilal, S. T. Chakradhar, and V. D. Agrawal, "A partition and resynthesis approach to testable design of large circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 10, pp. 1268-1276, October 1995.
- [114] Z. Kohavi, and N. K. Jha, *Switching and finite automata theory*, 3rd ed. New York: Cambridge University Press, 2010.
- [115] C. Pixley, S. W. Jeong, and G. D. Hachtel, "Exact calculation of synchronization sequences based on binary decision diagrams," in *Proc. ACM/IEEE Design Automation Conf.*, CA, USA, June 1992, pp. 620-623.
- [116] D. Chaum, "Blind signatures for untraceable payments," in *Proc. Advances in Cryptography*, D. Chaum, R.L. Rivest, & A.T. Sherman Eds., Plenum, 1982, pp. 620-623.
- [117] National Institute of Standards and Technology (NIST). Secure Hash Standard (SHS) (March 2012). [Online] Available:
<http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>
- [118] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. Boca Raton: CRC Press, 1997.
- [119] M. R. Garey, and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1990.
- [120] T. H. Cormen, *Introduction to algorithms*, 3rd ed. Cambridge, Mass.: MIT Press, 2009.

-
- [121] G. D. Hachtel, and F. Somenzi, *Logic synthesis and verification algorithms*, New York: Springer, 2006.
- [122] R. K. Ranjan, V. Singhal, F. Somenzi, and R. K. Brayton, "On the optimization power of retiming and resynthesis transformations," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, CA, USA, November 1998, pp. 402-407.
- [123] J. K. Rho, G. D. Hachtel, F. Somenzi, and R. M. Jacoby, "Exact and heuristic algorithms for the minimization of incompletely specified state machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 2, pp. 167-177, February 1994.
- [124] M. Lewandowski, "A Novel Method For Watermarking Sequential Circuits," M.S. thesis, Department of Computer Science and Engineering, University of South Florida, 2013.
- [125] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovannivincetelli, "Sequential-circuit design using synthesis and optimization," in *Proc. IEEE International Conference on Computer Design: VLSI in Computers & Processors*, MA, USA, October 1992, pp. 328-333.
- [126] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs: State-of-the-art implementations and attacks," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 534-574, August 2004.
- [127] T. Guneyusu, B. Moller, and C. Paar, "Dynamic intellectual property protection for reconfigurable devices," in *Proc. International Conference on Field-Programmable Technology*, Kitakyushu, Japan, December 2007, pp. 169-176.
- [128] M. A. Gora, A. Maiti, and P. Schaumont, "A flexible design flow for software ip binding in fpga," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 719-728, November 2010.
- [129] S. Chow, P. Eisen, H. Johnson, and P. C. Van Orschot, "White-box cryptography and an AES implementation," in *Proc. Workshop on Selected Areas in Cryptography*, Newfoundland, Canada, August 2002, pp. 250-270.
- [130] Z. Cherif, F. Flament, J. L. Danger, S. Bhasin, S. Guilley, and H. Chabanne, "Evaluation of white-box and grey-box noekeon implementations in FPGA,"

- in *Proc. International Conference on Reconfigurable Computing and FPGAs*, Quintana Roo, Mexico, December 2010, pp. 310-315.
- [131] "Overview datasheet - High performance AES (Rijndael) cores for Xilinx FPGA (Rev 2.4.0)," Helion Technology, January 2009.
 - [132] "Full datasheet - Fast hash core family for Xilinx FPGA (Rev 1.9)," Helion Technology, November 2011.
 - [133] S. Pappala, M. Niamat, and S. Weiqing, "FPGA based key generation technique for anti-counterfeiting methods using Physically Unclonable Functions and artificial intelligence," in *Proc. International Conference on Field Programmable Logic and Applications*, Oslo, The Netherlands, August 2012, pp. 388-393.
 - [134] T. Kean, "Secure configuration of a field programmable gate array," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, CA, USA, March-April 2001, pp. 259-260.
 - [135] L. B. N. Mentens, B. Preneel, I. Verbauwhede, N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, "An FPGA Implementation of Rijndael: Trade-offs for side-channel security," in *Proc. IFAC Workshop on programmable devices and systems*, Cracow, Poland, November 2004, pp. 493-498.
 - [136] N. Kamoun, L. Bossuet, and A. Ghazel, "SRAM-FPGA implementation of masked S-Box based DPA countermeasure for AES," in *Proc. International Design and Test Workshop*, Monastir, Tunisia, December 2008, pp. 74-77.
 - [137] Y. W. Francesco Regazzoni, and Francois-Xavier Standaert, "FPGA implementations of the AES masked against power analysis attacks," in *Proc. International Workshop on Constructive Side-Channel Analysis and Secure Design*, Darmstadt, Germany, February 2011, pp. 56-66.
 - [138] M. Nassar, S. Bhasin, J. L. Danger, G. Duc, and S. Guilley, "BCDL: A high speed balanced DPL for FPGA with global precharge and no early evaluation," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, March 2010, pp. 849-854.
 - [139] T. Güneysu, and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *Proc. Cryptographic Hardware and Embedded Systems*, Nara, Japan, September 2011, pp. 33-48.

-
- [140] R. McEvoy, M. Tunstall, C. C. Murphy, and W. P. Marnane, "Differential power analysis of HMAC based on SHA-2, and countermeasures," in *Proc. International conference on Information security applications*, Jeju Island, Korea, August 2007, pp. 317-332.
- [141] T. G. Malkin, F. -X. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Proc. International conference on Fault Diagnosis and Tolerance in Cryptography*, Yokohama, Japan, October 2006, pp. 159-172.
- [142] Q. Y. Chang N. Zhang, and Xiao Wei Liu, "A hybrid fault tolerant approach for AES," *International Journal of Network Security*, vol. 15, no. 1, pp. 263-269, January 2013.
- [143] M. Joye, P. Manet, and J. B. Rigaud, "Strengthening hardware AES implementations against fault attacks," *Information Security, IET*, vol. 1, no. 3, pp. 106-110, 2007.
- [144] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, and R. Van Keer, "Keccak implementation overview, version 3.2," May 2012.
- [145] S. Wei, A. Nahapetian, M. Nelson, F. Koushanfar, and M. Potkonjak, "Gate Characterization Using Singular Value Decomposition: Foundations and Applications," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 765-773, April 2012.
- [146] M. Potkonjak, "Non-invasive leakage power device characterization of integrated circuits using device grouping and compressive sensing," USA Patent US8286120 B2, October 2012.
- [147] L. N. Trefethen, and D. Bau, *Numerical linear algebra*. Philadelphia: Society for Industrial and Applied Mathematics, 1997.
- [148] Y. Alkabani, T. Massey, F. Koushanfar, and M. Potkonjak, "Input vector control for post-silicon leakage current minimization in the presence of manufacturing variability," in *Proc. ACM/IEEE Design Automation Conference*, Anaheim, CA, USA, June 2008, pp. 606-609.
- [149] D. Shamsi, P. Boufounos, and F. Koushanfar, "Noninvasive leakage power tomography of integrated circuits by compressive sensing," in *ACM/IEEE*

-
- International Symposium on Low Power Electronics and Design*, Bangalore, India, August 2008, pp. 341-346.
- [150] N. Beckmann, and N. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *Proc. International Workshop on Information Hiding*, Darmstadt, Germany, June 2009, pp. 206-220.
- [151] M. Potkonjak, S. Meguerdichian, A. Nahapetian, and S. Wei, "Differential Public, Physically Unclonable Functions: Architecture and Applications," in *Proc. ACM/EDAC/IEEE Design Automation Conference*, New York, USA, June 2011, pp. 242-247.
- [152] S. Meguerdichian and M. Potkonjak, "Matched public PUF: Ultra low energy security platform," in *International Symposium on Low Power Electronics and Design*, Fukuoka, Japan, August 2011, pp. 45-50.
- [153] S. Meguerdichian and M. Potkonjak, "Using standardized quantization for multi-party ppuf matching: foundations and applications," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, USA, November 2012, pp. 577-584.
- [152] X. Teng, J. B. Wendt, and M. Potkonjak, "Digital bimodal function: An ultra-low energy security primitive," in *IEEE International Symposium on Low Power Electronics and Design*, Beijing, China, September 2013, pp. 292-296.