Constraint-based watermarking techniques for VLSI IP protection

Cui, Aijiao

2009

Cui, A. (2009). Constraint-based watermarking techniques for VLSI IP protection. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/46983>

<https://doi.org/10.32657/10356/46983>

Nanyang Technological University

Downloaded on 26 Apr 2023 19:05:19 SGT

296
4X
Black
Color

Constraint-based Watermarking Techniques for VLSI IP Protection

CUI AIJIAO

School of Electrical & Electronic Engineering

A Thesis Submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2009

Acknowledgements

First, I would like to express my deeply heartfelt thanks to my supervisor, Associate Professor Chang Chip Hong, for his intellectual guidance, constructive and patient direction as well as the unceasing attention, support and encouragement. It is he who led me into a new wonderful palace of science, which made me enjoy the pursuit of the Ph. D. all the time. I wish to thank him for the open, interactive and relaxed academic environment he provided for me. For many times, the discussions with him have triggered my ideas on the research topic, justified my deviation from the right way and ignited my passion and interest in the work. I wish to thank him for teaching me the invaluable skills, necessary attitudes towards the academic research. In particular, I would like to thank him for all his kind consideration and care for me in my daily life and all the special time and effort he has spent to support me to fulfill the Ph. D. What I benefited from him has contributed much to my current career and I believe it will assist me more in my way ahead. I'd like to extend my sincere thanks to Mrs. Chang. Her kind care and considerations help me much in my private life in Singapore. I always feel fortunate to have known such a kind, wise and nice couple in my life.

I would like to thank Professor Sofiène Tahar and Dr. Amr Talaat Abdel-Hamid, who are glad to share their valuable ideas with me and contribute their algorithm and tool to facilitate my research work.

To all the members in Chip's family, I wish to convey my thanks to all of them for their kind and friendly assistance. I would like to thank the staff and the students in the Center for High Performance Embedded Systems for their kind help for my research work and all the happy days I have had with them. I would like to thank Mr. Xu Shouheng and Mr. Qu Hui for their tedious but indispensable work to speed up the research schedule.

I would like to acknowledge the School of Electrical & Electronic Engineering, Nanyang Technological University for providing the research facility and the research scholarship.

Acknowledgements

I wish to express my thanks in my deep heart ocean to my boyfriend, Li Hao for his consistent care and support. His words are always companying with me to shrink the tough time and double the happiness in my life.

My parents, my brother and my sister-in-law, are always caring and loving me. They unceasingly input encouragement, comfort and support to me. I can do nothing to express my deep gratitude to them but dedicate the thesis to them.

Table of Contents

Acknowledgements	i
Table of Contents	iii
Summary	vii
List of Figures.....	ix
List of Tables	xi
List of Abbreviations	xii
Chapter 1 Introduction	1
1.1 Preliminaries and Motivation	1
1.1.1 Intellectual Property in VLSI Design.....	4
1.1.2 IP Categories.....	5
1.1.3 IP Protection	7
1.1.4 Digital Watermarking for IPP	8
1.2 Objectives of the Thesis	9
1.3 Thesis Contributions.....	11
1.4 Organization of the Thesis	13
Chapter 2 Preliminaries and Literature Review.....	15
2.1 Basic Percepts of Constraint-based IP Watermarking.....	17
2.1.1 A Motivating Example.....	17
2.1.2 Context for Nonintrusive IP Watermarking	19
2.2 Desiderata for Watermarking-based IP Protection Technique	19
2.3 Generic Approach and Analysis of Constraint-based IP Watermarking.....	21
2.3.1 Generic Procedure for Watermark Synthesis	21
2.3.2 Generic Procedure for Watermark Verification	22
2.3.3 Proof of Authorship	22
2.3.4 Typical Attack Scenarios	23
2.3.4.1 Ghost Searching	24
2.3.4.2 Tampering	24
2.3.4.3 Forging	26
2.4 Static IP Watermarking Schemes.....	26
2.4.1 IP watermarking at System Synthesis Level.....	26
2.4.2 IP watermarking at Behavioral Synthesis Level	28
2.4.3 IP watermarking at Logic Synthesis Level	30

Table of Contents

2.4.4	IP watermarking at Physical Synthesis Level	34
2.4.4.1	Preprocessing of Path-timing Constraints	34
2.4.4.2	Watermarking for Row-based Placement	35
2.4.4.3	Watermarking at Standard-cell Routing	35
2.5	Dynamic Watermarking for IP Protection	36
2.5.1	Dynamic Watermarking at Algorithm Level	36
2.5.2	Dynamic Watermarking on Finite State Machine	36
2.5.3	Dynamic Watermarking at Design-for-Testability Stage	39
2.6	Watermarking for FPGA IP Protection	40
2.7	Hierarchical Watermarking	43
2.8	Watermarking for Design Tools and Algorithms	45
2.9	Summary	45
Chapter 3	IP Watermarking at Logic Synthesis Level	47
3.1	Introduction	47
3.2	Preliminaries	47
3.3	Watermark Insertion and Detection	51
3.3.1	Watermark Insertion	51
3.3.1.1	Watermark Encryption	51
3.3.1.2	Generation of Disjoint Closed Cones	52
3.3.1.3	Qualification of Closed Cones	54
3.3.1.4	Insertion of Watermark	57
3.3.2	Watermark Detection	59
3.4	Watermark Strength and Resilience Analysis	61
3.4.1	Strength of Watermark	61
3.4.2	Watermark Resilience Analysis	62
3.4.2.1	Watermark Denial	62
3.4.2.2	Watermark Removal	63
3.4.2.3	Watermark Addition	64
3.5	Experimental Results	65
3.6	Summary	73
Chapter 4	Synthesis-for-Testability Watermarking	75
4.1	Introduction	75
4.2	Percept of SfT-Based IP Watermarking	76
4.3	Scan Chain Ordering Problem for SfT Watermarking	79
4.4	Proposed Algorithm and Example	82
4.4.1	Preliminary Vectors Generation	82
4.4.2	Watermark Insertion	83
4.4.3	Watermark Detection	85
4.5	Public and Field Authentication of Multiple IP Authorships	87

Table of Contents

4.6	Watermark Resilience Analysis.....	90
4.6.1	Ghost Searching.....	91
4.6.2	Denial Attacks.....	92
4.6.3	Removal Attacks.....	92
4.6.3.1	Combinational Logic Re-synthesis	92
4.6.3.2	Scan Chain Exchange.....	93
4.6.3.3	Flip-Flop Exchange.....	94
4.6.4	Unauthorized Addition.....	97
4.7	Experimental Results and Discussions.....	98
4.8	Summary	105
Chapter 5	A New Hybrid Watermarking Scheme.....	107
5.1	Introduction.....	107
5.2	Finite State Machine Watermarking.....	108
5.2.1	Preliminaries	108
5.2.2	Generation of Watermark and Random Sequence	110
5.2.3	Watermarking the STG	111
5.2.4	Detection of Watermark.....	116
5.2.5	An Example of the Proposed FSM Watermarking.....	116
5.3	Watermarking on Scan Chain Ordering	118
5.3.1	Generation of Watermark Constraints	120
5.3.2	Construction of Weighted Connected Graph	121
5.3.3	Watermark Insertion	121
5.3.4	Verification of Authorship	123
5.3.5	A Simple Example	123
5.4	Watermark Resilience Analysis.....	125
5.4.1	Authorship Credibility	125
5.4.2	Resilience Analysis.....	127
5.4.2.1	Combinational Logic Re-synthesis	127
5.4.2.2	Circuit Retiming.....	127
5.4.2.3	State Recoding (or Assignment).....	128
5.4.2.4	Combinational and Sequential Redundancy Removal	128
5.4.2.5	State Reduction	129
5.4.2.6	Transition Elimination.....	130
5.4.2.7	Removal of Circuitries with Pseudo Inputs.....	130
5.4.2.8	Ghost Searching	130
5.4.2.9	Addition of Watermark.....	131
5.5	Experimental Results.....	132
5.6	Summary	140
Chapter 6	Conclusions and Future Work.....	142
6.1	Conclusions.....	142
6.2	Suggestions for Future Work.....	144
6.2.1	Combination of Static Watermarking with Dynamic Watermarking.....	144
6.2.2	Constraining Design Overheads of Watermarking	145
6.2.3	General Watermarking Scheme on FSM	145
6.2.4	Soft IP Protection.....	146
6.2.5	Birthmarking Method for CAD Tools and Algorithms.....	146

Table of Contents

6.2.6	Watermarking Industrial Size Designs.....	146
6.2.7	Watermarking Analog Integrated Circuits	147
Author’s Publication.....		148
Bibliography		150

Summary

With the increase in the popularity of reusable intellectual property (IP) cores for System-on-a-Chip (SoC), piracy is also on the rise in this new design area. The lure of getting a new product ahead of the competitor at a fraction of, or at no cost, is often too much a temptation to ignore, which results in an increasingly buoyant IP theft. To safeguard the IP owner against piracy and illegal redistribution of the IP cores, constraint-based watermarking has been used as a pivotal technology to protect the copyright of the very large scale integration (VLSI) IPs. In this thesis, a comprehensive framework of constraint-based watermarking techniques has been studied and explored at different design abstraction levels. The goal is to develop robust watermarking schemes and techniques for VLSI IP protection that can facilitate easy detection and tracking of IP misappropriation.

A novel adaptive watermarking technique on combinational logic network has been proposed. In this method, each disjoint local subcircuit in an originally optimized design is deemed as a potential watermark host. The headroom of each host is evaluated based on its slack and slack sustainability. The new notion of slack sustainability enables subcircuits in the critical path to be qualified as watermark hosts if their slacks can be better preserved upon remapping and the unfit watermark hosts are pruned by an embedding threshold. The watermark is embedded by remapping only qualified subcircuits randomly selected and template constrained by the signature. This parametric formulation provides a means to capitalize on the headroom of a design to increase the signature length or strengthen the watermark resilience. An appealing watermark recovery scheme has also been proposed to avoid the revelation of the constraint generator. The experimental results on ISCAS and LGSyn93 benchmarks show that the combinational networks can be marked efficiently by the proposed method with good timing performance and low area overhead. The authorship can be verified with very low probability of coincidence and the scheme is resilient to several attack scenarios.

Fraudulence tracking is easier if the embedded watermark can be detected publicly after the IP cores have been packaged. To bridge the gap between IP protection and IP

management, an on-site publicly detectable watermarking scheme has been proposed. The IP core is watermarked by a synthesis-for-testability (SfT) methodology, which makes it more resilient against removal attacks than a watermark imposed at the post-synthesis design-for-testability (DfT) stage. The watermarked constraints are imposed on the scan chain ordering problem so that the author's signature can be retrieved from the output sequence under a specific test vector. A nearest neighbor search algorithm has been used for the flip-flop reordering to minimize the overhead due to the watermark insertion. A provable mechanism has also been proposed to enable multiple authorships of different IP cores in a single chip to be publicly authenticated in the field. The experimental results show that the watermarked designs have low area overheads, and the probabilities of coincidence and removal reduce rapidly with increased watermark and scan chain lengths.

Lastly, a hybrid IP watermarking scheme has been proposed for the protection of sequential circuit designs. At the behavioral level, a new and efficient method is proposed to watermark a finite state machine (FSM) by interweaving the watermark bits into the outputs of its existing and free transitions. The authorship can be verified directly by running the watermarked FSM with a specific code sequence. The overheads of watermarking can be reduced by increasing the verification code length without compromising the authorship credibility. During the DfT process, the scan chain that incorporates the watermarked FSM as one of the test kernels is further watermarked to provide a second level of protection. The watermark on the FSM is publicly detectable by the legitimate end users through the scan test even after the watermarked IP has been encapsulated. Analysis shows that the authorship of dual-mode watermarking technique is harder to repudiate and the infringement is easier to trace. The watermarked circuits from the ISCAS89 and IWLS93 benchmark suites show lower or acceptably low overheads with lower probability of removal and stronger authorship proof in comparison with other watermarking schemes for sequential functions.

List of Figures

Figure 1.1	IP creation in SoC design flow	6
Figure 1.2	Comparison of different IP formats	7
Figure 2.1	Watermark synthesis for constraint-based watermarking	22
Figure 2.2	Watermarking on DSP code partitioning	27
Figure 2.3	Generic approach to watermarking at behavioral synthesis level	29
Figure 2.4	An example of watermarking technology mapping solution	31
Figure 2.5	Redundancy addition/removal	33
Figure 2.6	FSM watermarking in [ABD05b]	38
Figure 2.7	Watermarking during partial sequential scan logic test generation	40
Figure 2.8	Watermarking on non-critical paths of the FPGAs	43
Figure 2.9	Architectural level circuit transformations	44
Figure 3.1	Examples of a DAG network, a maximum cone (MC_v), a closed cone (C_v'), an open cone (C_v), and a complex cone ($C_{v,x}$)	49
Figure 3.2	Required time for the arrival of a signal at a node	50
Figure 3.3	Generation of distance- τ disjoint closed cone	53
Figure 3.4	Generation of (a) simple and (b) complex closed cones	54
Figure 3.5	Pruning of disqualified disjoint closed cones	57
Figure 3.6	Watermark insertion by incremental technology mapping	59
Figure 4.1	Scan chain design used in DfT	76
Figure 4.2	Merging of normal and scan functions	77
Figure 4.3	Scan function overheads and edge costs of dependency graph for SfT	80
Figure 4.4	SfT cost graph of a sequential circuit	81
Figure 4.5	Nearest neighborhood algorithm for watermark insertion	85
Figure 4.6	A simple example of watermark insertion.	86
Figure 4.7	Watermarking process on the example design	87
Figure 4.8	A scenario for scan chain exchange and flip-flop exchange	94
Figure 5.1	Flowchart of FSM watermarking	115
Figure 5.2	Example of watermarking on FSM	117

List of Figures

Figure 5.3 Watermark insertion by power-driven scan chain ordering algorithm 122

Figure 5.4 Example for scan chain watermarking..... 124

Figure 5.5 Nearest neighbor search for watermark insertion. 125

Figure 5.6 Retiming of the FSM 128

Figure 5.7 Watermarking with third party keeping a time-stamped signature..... 132

List of Tables

Table 3.1 Performance reference of master design..... 67

Table 3.2 Watermarking results on ISCAS85 benchmarks 68

Table 3.3 Watermarking results on LGSynth93 benchmarks 68

Table 3.4 Watermarking results on ISCAS89, ISCAS99 benchmarks 70

Table 3.5 Comparison with logic synthesis watermarking method in [KIR06] 71

Table 3.6 Comparison with logic synthesis watermarking method in [MEG00] 72

Table 4.1 Performance overheads on ISCAS89, ISCAS99 benchmarks..... 100

Table 4.2 Performance overheads on LGSynth93 benchmarks..... 101

Table 4.3 Comparison of dynamic watermarking methods 102

Table 4.4 Watermark strength and probability of removal..... 103

Table 5.1 Intersection of two ternary variables..... 112

Table 5.2 Statistics for ISCAS89 benchmark designs 133

Table 5.3 Statistics for IWLS93 benchmark designs..... 135

Table 5.4 Comparison with FSM watermarking method in [ABD05b] 137

Table 5.5 Comparison with FSM watermarking method in [OLI01] 138

Table 5.6 Comparison with FSM watermarking method in [TOR00] 139

Table 5.7 Scan power overheads and watermark strength of the hybrid watermarking 140

List of Abbreviations

ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit
ATPG	Automatic Test Pattern Generation
CAD	Computer-aided Design
CDFG	Control Dataflow Graph
CFE	Computational forensic engineering
CLB	Configurable Logic Blocks
CPU	Central Processing Unit
CSP	Constraint Satisfaction Problem
CUT	Circuit Under Test
DAG	Directed Acyclic Graph
DfT	Design-for-Testability
DSP	Digital Signal Processor
EDA	Electronic Design Automation
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
HDL	Hardware Description Language
IC	Integrated Circuit
IP	Intellectual Property
IPP	Intellectual Property Protection
MD	Message Digest
MSU	Mississippi State University
MUX	Multiplexer
ND	Non-Distinguishable
NN	Nearest Neighbor
NoC	Network-on-a-Chip
NP	Nondeterministic Polynomial-time
PI	Primary Input
PKC	Public-Key Cryptosystem
PNG	Pseudorandom Number Generator

List of Abbreviations

PO	Primary Output
RAR	Redundancy Addition/Removal
RTL	Register-Transfer Level
SAT	Satisfiability
SDF	Standard Delay Format
SFF	Scannable Flip-Flops
SfT	Synthesis-for-Testability
SI	Scan In
SIS	Sequential Interactive Synthesis
SO	Scan Out
SHA	Secure Hash Algorithm,
SNE	Sequentially Non-Excitable
SoC	System-on-a-Chip
STG	State Transition Graph
TSP	Traveling Salesman Problem
UART	Universal Asynchronous Receiver/Transceiver
VC	Virtual Component
VHDL	Very-High-Speed Integrated Circuit Hardware Description Language
VSI	Virtual Socket Interface
VLSI	Very Large Scale Integration

Chapter 1

Introduction

1.1 Preliminaries and Motivation

More than 700 years ago [MOH99], watermarks were used in Italy to indicate the paper brand and the mill that produced it. By the 18th century, watermarks began to be used as anti-counterfeiting measures on money and vital documents. The term watermark was introduced near the end of the 18th century, so named probably because the marks resembled the effects of water on paper. The first example of the technology to digital watermarking is a patent filed in 1954 by Emil Hembrooke for identifying musical arts [HEM54]. In 1988, Komatsu and Tominaga [KOM88] appeared to be the first to use the term “digital watermarking”. In around 1995, when the lost of revenue in the recording industry was escalating due to piracy, interest in digital watermarking began to mushroom.

Digital watermarking is the process of embedding possibly irreversible information into a digital object. It can be classified into two categories: visible watermarking and invisible watermarking. In visible watermarking, the information, which is typically text or a logo identifying the owner of the media, is visible in the primary object. In invisible watermarking, the information is added as bit stream into digital audio, picture or video covertly. It cannot be perceived although it is possible to detect the hidden information. An important application of invisible watermarking is the copyright protection, which is intended to prevent or deter unauthorized copying of digital arts. Unless otherwise specified, the terms “digital watermarking” and “watermarking” in this thesis refer to the “invisible digital watermarking”.

Watermarking is an information hiding technique. Steganography and watermarking are two important subdisciplines encompassed in the general term of information

hiding. Steganography achieves the information hiding by keeping the existence of the information secret while watermarking tries to make the information imperceptible. Watermarking is the practice of imperceptibly altering a cover (original article or media) to embed a message about that cover. The watermarks are inseparable from the cover in which they are embedded so that they cannot be easily erased or manipulated. The protection mechanism is different from cryptography in that the watermark remains even after the encrypted content has been decrypted and the watermarks can withstand normal processing that preserves the worth of the cover media. Over the last few decades, invisible or inaudible digital watermarking leveraging on the human sensory imperfection has evolved into a mature technology to protect the copyright of multimedia content [CHAN05, BONE96, BRAS96, BRAU96].

According to the definition provided by World Intellectual Property Organization, the term Intellectual Property (IP) refers to a bundle of legal property rights over creations of the mind, both artistic and commercial [WIPO]. The former is covered by copyright laws, which protect creative works, such as articles, images, films, music, paintings and software, and give the copyright holder exclusive right to control reproduction or adaptation of such works for a certain period of time. The latter is collectively known as “industry properties”. Typically, a patent is granted for a new, useful and non-obvious invention to provide creators of original works economic incentive to develop and share ideas through a form of temporary monopoly. IP is a unifying concept that governs the specific legal rights such as copyrights, trademarks, patents, and trade secret. Many derivatives of IP have evolved over centuries. In integrated circuit (IC) design, a semiconductor IP core or IP block is a reusable unit of logic, cell or chip layout design and is also the intellectual property of one party. The term is derived from the licensing of the patent and source code copyright that subsist in the design.

Very Large Scale Integration (VLSI) is a term created in the 1970s when complex semiconductor and communication technologies were being developed. It is used to describe the process of creating integrated circuits by combining thousands of transistor-based circuits into a single chip. The definition of VLSI is now somewhat

quaint, given that the chips have increased complexity over time into the hundreds of million of transistors. The huge number of gates that can be built into a single chip today has rendered the calibration of different levels of large-scale integration moot. In the System-on-a-Chip (SoC) era, the increased integration density and the shorter product life-cycle impact the product competitiveness directly. As the size and complexity involved in designing an SoC have outpaced the design team's productivity, reuse-based and platform-based design methodologies [HCH99] are introduced to speed up new product development. Reuse-based methodology involves partitioning of a complex design into smaller IP blocks with well-defined functionality that can be reused across multiple designs. Platform-based design methodology extends the reuse of IPs to architecture level paradigm by catering for an architectural independent system design. Although significant time and effort have been invested in the creation of reusable IP cores to shorten the time to market of new products, piracy can extort sizable revenue from the IP producers and pose a severe threat to reuse methodologies. Encrypting an IP block can only ensure its secured delivery by deterring adversaries from hijacking the copyrighted information. It does not safeguard an IP against piracy and its illegal redistribution upon decryption. Without a provable detection mechanism, such lawsuit is onerous and controversial. The IP owner must be prepared to divulge his/her design documents and methods to demonstrate his/her ownership in the court. This has in the late 1990's led to the thought of using fingerprinting¹ or watermarking to provide an undeniable and robust proof of IP authorship for VLSI designs.

Visible watermarking has been used to identify the origins of VLSI IPs for many years. Many chips have a "signature" area where the type of chip, designer credits, and company logo are displayed and these areas are usually found in the voids between registers, series of buses and caches or near the bonding pads. Such signature is built into the IC during the manufacturing process, probably requiring several masks to create [FSU]. As these areas can be examined without damaging the silicon surface, the signature is easily exposed or forged. This thesis will focus on the application of the invisible watermarking to the protection of VLSI IP instead.

¹ Fingerprinting deals with taking each copy of the protected content and making it unique to the person who receives it. By this way, if the work is shared, the owner knows exactly which recipient is responsible for the distribution of the copies of the work originally.

VLSI circuit watermarking is different from the media watermarking but its main objective is the same, which is to discourage intellectual property theft, or when such theft has occurred, allows the IP author to prove his/her ownership. When digital watermarking is applied to VLSI design [ABD05a], it augments the IP owner's opportunity to reclaim his/her loss of revenue upon infringement. Watermarking for VLSI IP protection (IPP) is still in its infancy and remains a hot topic both within industrial and academic circles after slightly more than a decade of research. Despite the challenges, there are some promising works recently, which suggest that efficient tools and methods are emerging to make the cost of both implementation and detection of watermarks economically feasible in the not-too-distant future [VSI00].

1.1.1 Intellectual Property in VLSI Design

The complexity of SoC is growing rapidly with the advance of IC fabrication technology. It is common to have the number of gates per chip in the order of several million. Intel announced the design of a two-billion-transistor computer chip for supercomputers in the beginning of 2008 [INTEL'08]. This integration density has exceeded the projection of Moore's law which stated that "The capacity of electronic circuits doubles every 18 months." [MOO65]. The maximum integration density is growing at more than 60% per year but the design productivity only has an average growth rate of 21% per year [BURG99]. The ever-increasing logic density has created a design productivity gap between what can be built in a chip and what can be designed. The productivity gap is becoming even worse because of generally decreasing time-to-market. A more intuitive interpretation of design productivity gap is the IC manufacturing industry is now faced with a problem that there are more transistors on silicon than the designer's ability to design them meaningfully within the product turnaround time. According to the reported data by the Interactive Business Systems, Inc. [IBS], in a fast moving market, being only three months late in a product delivery can cost over a quarter of the product's potential revenue.

In order to cope with the increased IC complexity and shorter design turnaround time, VLSI design methodology has undergone a paradigm shift. SoC refers to integrating all components of a computer or other electronic system into a single integrated chip.

The arrival of the era of SoC is marked by the principle of design reuse. The emphasis is on design of reusable functional modules and design of system with reusable components. Over the last decade, reuse-based design methodology has been adopted in all levels of the design flow, for both the hardware and the software. Utilizing the predesigned modules accelerates the development of new products to meet the time-to-market demand. In an IP-driven SoC design, large predesigned blocks, such as bus controller, central processing units (CPUs), memory subsystems, specialized peripheral functions like universal asynchronous receiver/transceiver (UART), etc., will be integrated into an application-specific integrated circuit (ASIC) architecture to implement routine functions so that the designers can focus on the design of new blocks, value-added functions and design innovation based on the system requirements. The newly developed blocks will also be tested, documented and deposited as design IPs for future use. Reusing the existing blocks that have been rigorously verified and tested also reduces the possibility of failure of a system based on the use of a newly designed and verified block for the first time. These reusable predesigned hardware and software modules are easily accessible, and easily integrable “virtual components” (VCs), which are commonly called the IP cores. They hold values as tradable as other commodity, and hence equally vulnerable to copying as other IPs.

1.1.2 IP Categories

In an SoC design, the system level model is constructed by introducing the requirements in the architectural and algorithmic designs. The system specification is decomposed and mapped into behavioral and system level models. These models are generated by assigning a specific hardware or software resource to every function. The partitioned modules are developed separately in the lower levels. The software part is developed using different programming languages. The hardware design follows the same hierarchical approach as used in the traditional IC design. IP cores can be used as building blocks within ASIC chip designs or field programmable gate array (FPGA) logic designs. In this design flow, as shown in Figure 1.1, to provide various levels of flexibility for reuse and optimization, three major forms of IP cores are categorized. They are soft, firm and hard IP cores.

Figure 1.1 IP creation in SoC design flow.

Soft IP cores are synthesizable version of IP cores. They are delivered either as generic gate netlists or in a hardware description language (HDL) such as very-high-speed integrated circuit hardware description language (VHDL) or Verilog at the register-transfer level (RTL). A generic netlist is a Boolean-algebra representation (gates, standard cells) of the IP's logic function. It protects the vendor against reverse engineering, while maintaining portability to multiple foundry targets. HDL cores permit customer modification at the functional level. These cores offer maximum flexibility and reconfigurability to match the requirements of a specific design application. Although soft IP cores provide the maximum flexibility for changing their features, they must be synthesized, optimized and verified by their users before integration into the SoC designs. Some of these tasks could be performed by the IP providers. However, it is not possible for the provider to support all the potential libraries. Therefore, the quality of a soft IP is highly dependent on the effort needed in the IP integration stage of the SoC design.

Hard IP cores consist of layouts using particular physical design libraries and are usually delivered in masked-level designed blocks (GDSII format [CIR91]). These cores offer optimized implementation and the highest performance for their chosen physical library. The integration of hard IP cores is quite simple and the cores can be

dropped into an SoC physical design with minor integration effort. However, hard cores are technology dependent and provide minimum flexibility and portability in reconfiguration and integration across multiple designs and technologies.

Firm IP cores are the watershed. Like the hard cores, firm (sometimes called semi-hard) IP cores also carry placement data but are configurable to adapt to various applications. They balance the high performance and optimization properties of hard IP cores with the flexibility of soft IP cores. These cores are delivered in the form of targeted netlists to specific physical libraries after going through the synthesis but without performing the physical layout. A firm IP has a certain degree of technology dependence, but remains relatively generic. The users are responsible for the physical design steps (placement and routing) and their verification.

The characteristics of these three formats of IP are summarized in Figure 1.2.

	Soft IPs	Firm IPs	Hard IPs
Representation	RTL	Targeted Netlist	GDSII
Optimization	Low	High	Very High
Technology	Technology Independent	Technology Generic	Technology Dependent
Reusability	Very High	High	Low
Distribution Risk	High	High (with RTL)	Low

Figure 1.2 Comparison of different IP formats.

1.1.3 IP Protection

The goals of IPP include the protection of IPs against unauthorized use, the protection of design data used to generate and deliver the IPs and the detection and tracing of the use of IPs. There are three approaches [VSI00] to the problem of securing an IP core. The *deterrent* approach provides external communication of legal protection in an attempt to deter an illegal act from occurring. It does not provide any physical protection. The available means include: patents, copyrights, trade secrets, contracts

and lawsuits. The *protection* approach involves taking active steps to try to prevent the unauthorized use of IP's from occurring. The protection mechanisms include such tangibles as: licensing agreements and encryption. The *detection* method involves the ability to determine that an unauthorized use has occurred and then, tracing the source of theft. Detection and traceability methods that are becoming available include: foundry IP tracking or tagging, digital signatures such as digital fingerprinting and digital watermarking and noise fingerprinting. Detection-based IPP such as watermarking is more efficient and economical than the deterrent and protection approaches like copyright, patent and licensing agreements.

1.1.4 Digital Watermarking for IPP

Digital watermarking is an indirect protection scheme in that it deters illegal IP copying by offering the originator of a design the ability to demonstrate his/her ownership. IP watermarking enables self-protection by embedding covert and indelible authorship information into the design for ownership authentication. The imperceptibility of multimedia watermark stems from the imperfection of human sensory system. In contrast, watermarking for IPP imposes that the watermarked IP must also remain functionally correct. The transparency of IP watermarking is achieved by making the changes induced by the watermark "invisible" to an experienced designer so that they could not be easily detected or erased. At the same time, the cost and performance (quality) of the watermarked IP shall not be unduly compromised. The watermarked IP should also be resilient to malicious attempt to delete or replace the watermark information without deteriorating the design functionality or performance to an extent that renders the design unusable. The key challenges in this area are the development of tools and methods that are extremely difficult to defeat, have low cost/performance penalties, do not impede the native operation of the system, and can be provably accepted as a proof of ownership in a court of law. Besides, a direct and effective detection scheme is indispensable for a promising IP watermarking approach. Compared to watermark embedding, relatively less attention and effort have been devoted to the efficient implementation of watermark detection.

The prevailing VLSI design IP watermarking techniques fall under the static constraint-based watermarking paradigm [ABD05a]. Under this popular class of techniques, to enact that the presence of the watermark in an IP is a consequence of some deliberate actions, the watermarked design is usually reverted to the level of abstraction in the watermark embedding process to prove that the additional constraints generated by the secret signature are also satisfied. This reverse engineering process is expensive. It exposes the grammar used to generate the constraints and creates potential security leak to other similarly marked designs. On the contrary, the class of dynamic watermarking schemes [HON98, TOR00, OLI01, ABD05b] allow the ownership information to be verified from the output response of the watermarked design by injecting a specific input vector. Finite state machine (FSM) is a popular target for dynamic watermarking. The main advantage of these approaches is the direct traceability of the watermarked IP at the lower design levels, and the increase in reverse engineering cost for the aggressor due to the high computational complexity needed to extract the state transition graph (STG) from the circuit netlist for large machines [OLI99]. However, direct authentication of the embedded IP core after chip fabrication and packaging is still impossible for the legal IP buyer. Besides deterring high-tech piracy, another incentive of IPP that has often been overlooked is the ease of identification and tracking of IP distribution by lawful IP providers, buyers and consumers [ABD05a]. Such IP management and risk assessment are to be carried out directly on the packaged ICs and publicly in the field, a challenge that has not been fully resolved by the current static and dynamic VLSI IP watermarking techniques.

1.2 Objectives of the Thesis

The purpose of watermarking is to provide an ability to prove the ownership of a pirated design, and hence facilitates effective tracking of the IP licensees that are ultimately responsible for the fraudulence. The objective of this research is to propose novel and robust constraint-based watermarking methods for the copyright protection of VLSI IPs. The scope of this research involves the development of static and dynamic watermarking schemes at different levels of design abstraction. The following specific goals have been set to achieve this research theme.

Owing to the additional constraints, watermarking-based IPP will inevitably introduce design overhead. As combinational circuits possess ample opportunities for logic minimization, a new static watermarking technique targeting the logic synthesis level will be proposed. The challenge is to safeguard the grammar used to generate the watermarked constraints in the watermark detection process. Besides reducing the security leak, it is also desirable to alert on design tampering by tracking partially modified watermarked subcircuits in a manner similar to fragile watermarking in image watermarking [LI04], in which a watermark is embedded within an image such that subsequent alterations to the watermarked image can be detected with high probability.

Due to its advantage of intuitive and direct watermark detection, dynamic watermarking scheme will also be explored at the design-for-testability (DfT)² process to enable field authentication of authorship after the IP cores have been integrated into SoC and packaged. As IP cores have had a profound impact on the design of SoCs, it is important that multiple IP cores of different authorships can be independently and publicly authenticated without compromising the watermark strength or the security of each IP. This thesis will also research into a provable protocol for versatile IP authentication that can facilitate the tracking of licensees who are ultimately responsible for the illegal IP distribution.

As test architecture is independent of the design functionality, it is vulnerable to removal attack. To develop a hard-to-remove and easy-to-trace watermarking scheme, the merits of watermarking at multiple design abstraction levels will also be explored. Watermarking at higher design level can better resist typical attacks performed at lower levels if the watermark can survive the optimization carried out at the design level where the watermark is embedded. To combine the easy-to-trace merit enabled by the backend DfT process and the watermark robustness by frontend watermarking, some high level design representation of sequential circuits will be exploited to develop a seamless and cohesive hierarchical watermarking scheme.

² DfT is a name for the design techniques that add certain testability features to a microelectronic hardware product design. The added features make it easier to develop and apply manufacturing tests for the designed hardware. The purposed of manufacturing tests is to validate that the product hardware contains no defects that could, otherwise, adversely affect the correct function of the product.

1.3 Thesis Contributions

The work reported in this thesis aims to provide better IP watermarking methodologies with higher credibility, stronger resilience against tampering, lower design overheads and new protection features not offered by the existing watermarking solutions. The attempt has led to the following major contributions and original results.

(1) IP Watermarking using Incremental Technology Mapping at Logic Synthesis Level

A new adaptive watermarking technique at logic synthesis level has been proposed. This was the first proposal that makes use of the concept of an originally optimized master design to extract the excess bandwidth relative to the original timing specifications for watermark embedding. Incremental technology mapping is applied on the disjoint closed cones to overcome the limitation of poorer and less predictable watermarked solution quality obtained by technology mapping of overlapping maximum cones. Both the timing slack and the proposed slack sustainability metrics to qualify disjoint closed cones for watermark embedding have prevented the watermark bits from being conspicuously hosted in only the non-critical paths. The revelation of the grammar used to generate the constraints has also been avoided by the proposal of an alternative signature correlation verification method. This direct watermark retrieval method also allows partially corrupted watermark bits to be independently detected.

(2) Synthesis-for-Testability³ Watermarking for Field Authentication of VLSI IP

To facilitate authorship authentication of an encapsulated IP core in the field, a new dynamic watermarking technique has been proposed. The watermark is embedded as implicit constraints to the scan chain ordering problem to enable the watermark

³ Originally, synthesis-for-testability (SfT) identified gate level optimizations that could preserve or enhance circuit testability for a selected fault class without the need for more specific testability insertion techniques. In the last decade, research in SfT has focused on the generation and processing of HDL specifications of system function. EDA vendors and researchers are exploring the coupling of the HDL descriptions with testability structures, since it is attractive to apply synthesis and optimization technology directly to a testable HDL description to optimize functional and test logic concurrently [WAG96].

sequence to be observed at the output pin by the verifiers that possess the correct input code. The proposed SfT watermarking method has not only reduced the area overhead of the design, but also made attempts to obfuscate the embedded watermark more costly and fragile as such attempts are more likely to impact the design specification and optimality and leave clearer misappropriation traces for forensic analysis than when the attacks are mounted on existing DfT-based watermarking. The method is also applicable to the programmable logic devices and blends well with other dynamic watermarking to withstand different types of attacks.

(3) Identification Protocol for Verification of Multiple Authorships of Different Watermarked IP Cores in an SoC

An identity-based watermark verification scheme has been proposed to enable multiple authorships of different watermarked IP cores in a packaged chip to be independently and publicly authenticated in the field by the IP buyers. Different IP ownership can be verified publicly under this identification protocol without compromising the strength or security of the watermark. The proposed scheme also allows the IP owner to trace illegal IP redistribution by assigning different authentication codes of the same IP to different customers. As dynamically reconfigurable FPGA [MAC01] is a key technology driver of software defined radio, this unique identity-based watermarked IP authentication scheme supports the vision of anti-trust protection and licensed hardware update management by providing multiple ownership declaration and in-system authenticity verification for the reconfigurable bitstream downloaded over the air [DRE01].

(4) A Hybrid Watermarking Scheme for IP Protection of Sequential Circuit Designs

A new robust watermarking scheme that spans two design abstraction levels has been proposed for the copyright protection of sequential functions. At behavioral synthesis level, the watermark is embedded in the STG of the FSM to enable the authorship to be verified directly by running the watermarked FSM with a specific code sequence. In the backend, the scan chain that incorporates the watermarked FSM as one of the test kernels is further watermarked with the scan chain developer's signature to provide a secondary protection with a fragile watermarking feature. The verification

code sequences that come piggyback with the scan vectors allow the end users to directly authenticate the authorship of the encapsulated IP in the field. With complementary protection modes at different abstraction levels, the proposed watermarking scheme makes the authorship proof harder to erase and the IP forgery easier to trace.

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 motivation and objectives of the research work. The achievements resulting from the work presented in this thesis are summarized into four main contributions in this chapter.

Chapter 2 presents the fundamentals of constraint-based watermarking scheme for IPP and surveys the related IP watermarking techniques proposed at different design levels. The core concepts, merits and limitations of several important static, dynamic and hierarchical watermarking methods are discussed.

The original contributions of this research are presented in Chapters 3 through 5. Chapter 3 proposes an adaptive watermarking technique on combinational logic network by incremental technology mapping. The watermark is embedded by modulating some qualified closed cones for technology mapping. An alternative watermark detection scheme is also detailed in this chapter. Some attack scenarios are analyzed. At the end of the chapter, the experimental results are presented and compared with other existing watermarking methods at logic synthesis level [KIR06, MEG00] to demonstrate the merits of the proposed method.

Chapter 4 commences with the introduction of scan chain watermarking technique based on the SfT concept. The scan chain ordering problem for SfT watermarking is formulated. An illustrative example is provided to explain the watermark insertion and detection methods. Several conceivable attack scenarios are analyzed and discussed.

The probability of coincidence and the probability of watermark removal by random permutation of the scan flip-flops are also derived. Experiments are performed on benchmark circuits to evaluate the design overheads, watermark strength and robustness. The results are compared with a recent dynamic watermarking technique [OLI01].

Chapter 6 proposes a hybrid IP watermarking scheme for sequential circuit designs. At behavioral level, an efficient method is proposed to watermark an FSM by interweaving the watermark bits into the outputs of its existing and free transitions. The second level of protection is provided at the DfT stage, where the scan chain that incorporates the watermarked FSM as one of the test kernels is further watermarked. The probability of coincidence for each watermarking scheme is derived and the false positive rate is formulated. The performance overhead and authorship proof of the FSM watermarking are evaluated and compared with three other FSM watermarking schemes [OLI01, TOR00, ABD05b]. Besides, the scan power overheads and watermark strength of the hybrid watermarking are discussed.

Finally, Chapter 7 reviews the methods proposed in this thesis by highlighting their features and merits, discusses some of the remaining problems and suggests seven potential future research directions.

Chapter 2

Preliminaries and Literature Review

The VLSI design automation tool flow involves a large number of optimization tasks. Each optimization task is NP-hard and usually can be performed by a heuristic solver capable of generating many different solutions that satisfy the design specific constraints. These optimization tasks can hence be treated as constraint satisfaction problems (CSPs). In short, CSP is a generalization of the first proven NP-complete decision problem known as Boolean or propositional satisfiability (SAT) problem [COO71]. Efficient and scalable SAT solvers are the backbone that endows modern electronic design automation (EDA) tools with the ability to automatically solve problem instances involving tens of thousand of variables and millions of constraints. CSP solvers virtually explore the exponentially-sized space of variable assignments to locate satisfying assignments. The excess solution space of CSP has been exploited by Lach *et al.* [LAC98a, LAC98b] to pioneer the research in *Constraint-based Watermarking* for IPP.

Under the framework of constraint-based watermarking, many IP watermarking techniques have been developed at different design abstraction levels throughout the VLSI tool flow [CAS07]. Some of them have been proposed for dedicated IP cores on digital signal processors (DSPs) and FPGAs with consideration of their specific programmable and configurable hardware structures. A complete IPP scheme based on watermarking consists of two phases: watermark synthesis (or more commonly known as watermark embedding) and watermark detection. The watermark synthesis phase is characterized by an algorithm or a set of transformations that maps some abstract design features onto a functionally equivalent and/or topology invariant instances under the watermarked constraints which are cryptographically generated by a secret owned by the IP creator. The watermark detection phase involves the

verification that the transformed design satisfies the unique watermarked constraints. Based on the watermark detection criteria, the prevailing watermarking methods for IPP can be classified into static watermarking and dynamic watermarking [ABD05a]. In static watermarking [KIR98b], to enact that the presence of the watermark in an IP is a consequence of some deliberate actions, the watermarked design is usually reverted to the level of abstraction in the watermark embedding process to prove that the additional constraints generated by the secret signature are also satisfied. This reverse engineering process is expensive. It exposes the grammar used to generate the constraints and creates potential security leak to other similarly marked designs. Dynamic watermarking schemes, on the contrary, allow the ownership information to be verified from the output response of the watermarked design by injecting a specific input vector. Static watermarking has been found in different levels of design flow whereas dynamic watermarking is typically applied to only certain design abstractions. There are merits and limitations of the techniques in both categories. Recently, hierarchical watermarking schemes have also been proposed to concurrently watermark the design at multiple design levels.

The research work covered in this thesis focuses mainly on the constraint-based watermarking for IPP. In this chapter, some preliminaries on this scheme are introduced and the constraint-based watermarking techniques are surveyed. In Section 2.1, a motivating example is used to illustrate the key ideas behind the constraint-based watermarking for IPP. Section 2.2 outlines the desiderata for a given watermarking-based IPP technique. This is followed by the percepts and a general approach to constraint-based IPP. Section 2.4 reviews several static IP watermarking techniques developed at different design abstraction levels with emphasis placed on the watermarking techniques at combinational logic synthesis level. The dynamic watermarking schemes are surveyed in the next section. Section 2.6 reviews some watermarking schemes proposed for IP cores implemented on FPGAs. The emerging hierarchical watermarking schemes and some new watermarking proposals to protect the design tools are briefly introduced in Sections 2.7 and 2.8, respectively. The chapter is concluded by a summary in Section 2.9.

2.1 Basic Percepts of Constraint-based IP Watermarking

2.1.1 A Motivating Example

Satisfiability is the problem of determining if there exist some assignments of T (TRUE) or F (FALSE) values to the variables of a given Boolean expression such that the entire expression is evaluated to be T . What is equally important is the implication when such assignment does not exist, which means that the function expressed by the formula is identically F for all possible variable assignments. In this latter case, the function is said to be unsatisfiable. Otherwise, it is satisfiable. To emphasize the binary nature of this problem, it is frequently referred to as Boolean or propositional satisfiability. The shorthand “SAT” is also commonly used to denote it, with the implicit understanding that the function and its variables are all binary-valued. An SAT problem can be formulated as:

SAT (U, C):

Literal: A literal is a variable or the negation of a variable. For example, u_1 is a positive literal and $\bar{u}_1$ is a negative literal.

Clause: A clause is a disjunction of literals. For example, $\{u_1, \bar{u}_2\}$ (operation u_1 OR $\bar{u}_2$) is a clause.

Instance: A finite set of variables U and a collection $C = \{c_1, c_2, \dots, c_m\}$ of clauses over U .

Question: Is there a truth assignment for U that satisfies all the clauses in C ?

Problems from many application domains have been modeled as SAT instances. In VLSI computer-aided design, SAT formulations have been used in testing [CHAK93], logic synthesis and physical design [DEV89]. In this section, a 2SAT, where each clause has exactly two variables, is used as an motivating example of the key idea behind the whole framework of constraint-based IP watermarking techniques.

Suppose a 2SAT instance is given by: $U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7\}$ and $C = \{\{\bar{u}_1, \bar{u}_2\}, \{\bar{u}_1, \bar{u}_4\}, \{\bar{u}_2, u_5\}, \{u_6, u_7\}\}$. There are totally 2^7 different assignments of T or F values to the variables in the set U . For this satisfiable instance, an exhaustive

enumeration of the solution space shows that 48 different assignments satisfy all the clauses in C . One of the satisfying truth assignments is given by $t(u_1) = F$, $t(u_2) = F$, $t(u_3) = T$, $t(u_4) = F$, $t(u_5) = F$, $t(u_6) = T$, $t(u_7) = T$.

Suppose the process of IP creation involves solving a 2SAT problem. The IP can be watermarked by modifying the 2SAT instance such that the solution to the modified instance is a solution to the original instance and the modified instance with its solution contains the signature information. An encoding rule (case-insensitive) is set as $A - u_1$ (Here, the symbol “-” represents that the alphabet at its left, “A”, is encoded into the literal at its right, “ u_1 ”), $B - \bar{u}_1$, $C - u_2$, $D - \bar{u}_2$, $E - u_3$, $F - \bar{u}_3$, $G - u_4$, $H - \bar{u}_4$, $I - u_5$, $J - \bar{u}_5$, $K - u_6$, $L - \bar{u}_6$, $M - u_7$, $N - \bar{u}_7$. The additional constraints to be imposed in the form of extra two literal clauses are generated using the above rule. Suppose the signature is “Lihcaj”. As $L - \bar{u}_6$, $I - u_5$, $H - \bar{u}_4$, $C - u_2$, $A - u_1$, $J - \bar{u}_5$, the signature would be encoded using three extra clauses: $\{\{\bar{u}_6, u_5\}, \{\bar{u}_4, u_2\}, \{u_1, \bar{u}_5\}\}$. The addition of these clauses into the collection of clauses C reduces the number of satisfying assignments from 48 to 8 by exhaustive enumeration. It is then claimed that any satisfying assignment for this modified 2SAT instance contains the signature and that the probability that such a solution is coincidentally generated by others is 8 in 48. The additional clauses force the solution space to shrink. They are nonintrusive in that any existing solver remains applicable to the augmented 2SAT instance. In addition, the larger a given instance, the more amount of information can be embedded into its solution as watermark.

The SAT problem can be significantly broaden while still remaining NP-complete. Satisfiability modulo theories are an extension to the conjunctive normal form of SAT formalation that has gained popularity since 2003. They can enrich the restrictive Boolean formulas with linear constraints, arrays, all-different constraints, uninterpreted functions, etc. These NP-complete constraint satisfaction problems are omnipresent in the VLSI IP design flow. This simple example of watermarking the 2SAT instance can be extended to many variants of such problem that deal with the number of variable assignments making the formula true by viewing the watermark embedding as a way of limiting the possible solutions to the problem to a smaller

subset. The existence of the watermark is proved by the exceedingly low probability that a solution satisfies the augmented (watermarked) instance. Besides, the desired functionality should not be altered or the design performance should not be unduly degraded. Otherwise, the watermarking scheme will not be applicable in practice.

2.1.2 Context for Nonintrusive IP Watermarking

To form the context for a nonintrusive watermarking procedure, the following ingredients must be present. First, there must be an *optimization* problem, which has multiple acceptable solutions to accommodate a digital watermark. Such problems, usually corresponding to some design synthesis tasks which are so complex that minimal solutions are prohibitively costly to enumerate and some heuristic algorithms must be employed to generate an acceptable solution. Second, the solutions of the referred *optimization* problem must have well-defined specifications to be properly *interpreted* as an IP. Third, there exist appropriate algorithms and/or software tools that can solve the original *optimization* problem and are compatible for performing the optimization before and after the addition of the steganographic constraints during the watermarking process. Finally, it must possess the general requirements of robustness and credibility as for the currency watermarking. The protection method must ensure that the effort to remove or forge a watermark is as difficult as the repetition of the specified optimization (synthesis) task. Copying and unauthorized tampering with a watermarked design shall be provable in court.

2.2 Desiderata for Watermarking-based IP Protection Technique

From the research and implementation points of view, IPP invokes a unique set of new requirements that must be addressed by mathematically sound and practical techniques. In general, a complete IPP scheme based on watermarking is fully characterized by the watermark synthesis and watermark detection phases. Beyond theoretical significance of CSP, the watermarks can be designed with inherent redundancy to combat partial forging and/or deletion.

The general requirements of IPP for VLSI design are very similar to those of multimedia watermarking. In order to prevent the attacker from detecting the presence

of the watermark, the watermark must be subtle but obvious to the owner when it is required to prove the ownership or authenticity. It must be imperceptible in the cover media into which it is embedded, such that it does not disrupt the original form of the media. The imperceptibility in multimedia watermarking is accomplished by exploiting the imperfection of human visual or auditory system. On the contrary, for VLSI IPP, the modified IP must maintain the original specified functionality. The embedded watermark should be “invisible” to an experienced designer to add barrier to its disruption or erasure. At the same time, the design performance of the watermarked IP shall not be overly compromised. Depending on the type of IP and objective of the protection scheme, the watermarked IP should be robust against various possible attacks to remove the watermark.

The desiderata for a given watermarking-based IPP technique suggested by the Virtual Socket Interface (VSI) Alliance [VSI97] are outlined as follows:

1) Maintenance of Functional Correctness.

Design functionality should not be altered during the application of the IPP technique.

2) Transparency to Existing Design Flows.

The IPP technique should be transparent to the already complex design and verification process.

3) Minimal Overhead Cost.

Both the cost of the application of the protection technique and the hardware overhead due to it should be kept low.

4) Enforceability.

The technique should provide strong and undeniable proof of authorship when called upon in court.

5) Flexibility in Providing a Spectrum of Protection Levels.

It should enable a spectrum of protection levels corresponding to variable cost overheads.

6) Persistence.

The removal of the embedded watermark should result in a task of the difficulty comparable to the complete redesign of the specified functionality.

7) Invisibility.

The embedded watermark should be invisible to the machine or tool inspection.

8) Proportional Component Protection.

Ideally, a good watermark should be effective all over the software or design in order to protect all parts of it.

2.3 Generic Approach and Analysis of Constraint-based IP Watermarking

2.3.1 Generic Procedure for Watermark Synthesis

The general strategy is to transform an author's signature to a set of constraints that can independently hold for a particular solution or a small subset of solutions over an enormously large solution space. The signature is said to be present in a design if disproportionately many of these extra constraints are satisfied by the design. The assumption of independence implies that the amount of watermarking constraints to be imposed on a design stage is a trade-off between the odd that other design carries the same watermark unintentionally and the quality of the watermarked design. Therefore, choosing the type of constraints and the tactic (e.g., preprocessing or post-processing) can dramatically affect the strength of the authorship proof and the degradation of solution quality caused by watermarking.

The generic procedure for watermark synthesis is depicted in Figure 2.1. The signature to be embedded is encrypted and translated into to a set of additional design constraints. The constraint generator is designed in such a way that these extra constraints are generated without conflicting with the constraints of the original problem. The original and additional constraints are then combined to form a more constrained stego-problem. The black-box optimizer solves the stego-problem to satisfy both the cover and embedded constraints to produce a watermarked solution.

The embedded constraints limit the set of possible solutions to a smaller set. The watermark becomes stronger as the solution space for the stego-problem is smaller than that for the original problem. For a good scheme, the probability that another optimized design satisfying both the original and extra constraints by coincidence must be extremely small. This generic watermark synthesis makes no assumption about the type of optimizer or constraint satisfaction problem. Different schemes can therefore be developed to embed authorship information at different design abstraction levels in the VLSI tool flow, from system synthesis, behavioral synthesis down to the physical placement and routing processes.

Figure 2.1 Watermark synthesis for constraint-based watermarking.

2.3.2 Generic Procedure for Watermark Verification

To demonstrate that a particular stage was watermarked, the IP designer must show that the solution satisfies a disproportionate number of constraints generated by his/her signature. To verify that the IP designer's signature is present in the solution, the probability of that many or more of the constraints being satisfied by coincidence needs to be calculated. If this probability is sufficiently low, it means that the satisfiability of these extra constraints is a result of an act of due diligence carried out by the designer in the process of IP creation. To prove the ownership of a watermarked design in this manner requires the IP designer to divulge his/her signature, and hence, the chosen constraints.

2.3.3 Proof of Authorship

A strong proof of authorship should be provided to claim the legitimacy of the

protected object. In the watermarking scheme, the probability that the watermarked constraints are satisfied by coincidence, which is commonly denoted as P_c , serves as the watermark's proof of authorship. The lower the value of P_c , the stronger the proof of authorship. In most VLSI IP watermarking schemes, the P_c value cannot be computed exactly. Under this circumstance, it is acceptable to overestimate P_c so that an upper bound on P_c can be reported. A straightforward computation of such upper bound on P_c was derived in [KAH01]. Let p denote the probability that a single random constraint is satisfied by coincidence. This value or an asymptotical upper bound on it can be readily estimated with reasonable statistical justification from the definition of a constraint. For example, if a constraint is defined as "AND gate is used in a local circuit", p can then be set as 1/2. To simplify the analysis, p is assumed to be independent of whether the other constraints were satisfied. If the assumption is not true for some schemes, as long as the number of additionally imposed constraints is sufficiently small, this probabilistic formulation is still a valid approximation [KAH01]. Suppose C is the total number of imposed constraints and X is a random variable that represents the number of the C constraints that are not satisfied by a design that meets the original specifications. Then, P_c can be mathematically formulated as a sum of binomials based on the probability that out of the C constraints, coincidentally only b or less of them are not satisfied. The expression of P_c is given by [KAH01]:

$$P_c \equiv P(X \leq b) = \sum_{i=0}^b C_i^C \cdot p^{C-i} \cdot (1-p)^i \quad (2.1)$$

where $C_r^n = \frac{n!}{r!(n-r)!}$.

2.3.4 Typical Attack Scenarios

There are many different strategies an aggressor can think of to attack a watermarking scheme. The comment that "All methods have limitations. The goal of achieving protection of large amounts of embedded data against intentional attempts at removal may be unobtainable." [BEN96] is also valid for VLSI design watermarking. It is impossible to imagine or analyze all possible attacks. Fortunately, most of these attacks are fundamentally related and it would be reasonable to discuss a few major

and more prominent scenarios of attacks, such as finding “ghost signatures”, tampering, and forging. Derivatives of these attack scenarios specific to the watermarking schemes will be especially analyzed for the proposed watermarking in Chapters 3 to 5. “Alice” is referred to as the IP owner and “Bob” as the aggressor. The IP authored by Alice has been protected by constraint-based watermarking and Bob attempts to subvert such protection from the watermarked copy of Alice’s IP.

2.3.4.1 Ghost Searching

Bob steals the IP from Alice and claims that the IP belongs to him. To demonstrate that the IP contains his own watermark, Bob needs to find a set of constraints that correspond to a *ghost signature* (an imaginative signature that was not originally conceived in the watermarked IP) and show that these constraints can yield a favorable P_c . Bob has only two possible approaches to make this happen. He may choose a set of constraints (presumably ones that yield a good proof of authorship P_c) and then attempt to find a signature that corresponds to this set. For a convincing authorship proof, the signatures have to be generated using a cryptographically secure one-way function, which means that Bob has to reverse the hash function to find a matching signature. This is computationally intractable due to the provable security of the cryptographic hash function. Alternatively, Bob may try a brute-force approach to find a signature that corresponds to a set of constraints that yields a convincing proof of authorship, P_c . However, this brute-force attack becomes computationally infeasible due to the NP-complete nature of the stego-problem. It is extremely difficult, if not impossible, for such attempt to succeed in meeting an acceptably low threshold for a convincing proof of authorship for a well-designed constraint-based IP watermarking scheme.

2.3.4.2 Tampering

Bob may *tamper* with Alice’s solution to nullify Alice’s claim of authorship. Ideally, such tampering would completely remove Alice’s signature and better still, allow Bob to add his own signature. If Bob can obtain the design representation at the stage in which Alice’s watermark is embedded, Bob can simply process the design from scratch to embed his own watermarked design constraints and continue with all the

subsequent processing procedures. It is believed that in realistic scenarios, the effort of Bob to carry out all of the subsequent phases of the design and verification is not trivial. The time and cost required can be as high as a complete redesign of Alice's IP in the absence of understanding of the design concept and optimization strategies adopted by the designer, particularly if the watermarking was carried out relatively early in the design process. Having said that, direct access to raw watermarked design shall be refrained by distributing only the post-processed watermarked IP or by keeping some vital design information proprietary so that considerable reverse engineering effort is necessary to revert it to the stage at which the watermark was embedded. If Bob would like to tamper the watermarked solution without redoing or resolving every subsequent phase of the design process, he will have to transform the solution output by the last phase of the design process. In the view of the effect it has on the final design solution, the modifications that Bob made to the final solution are likely to belong to either (1) local perturbations of the solution to the watermarked phase or (2) global-scale transformations such as those that exploit a symmetry of the design representation [KAH01]. Special effort is then also required by Bob to guarantee that the design will not be unduly degraded. Understanding the limitation of Bob's tampering attack, it is important that Alice makes her watermarking technique resilient to such transformations. As Bob does not have the knowledge of the correspondence between the constraints and Alice's signature, tampering attacks might not be able to ruin the proof of authorship before they degrade the quality of the final solution significantly and render the tampered solution worthless. Unless Bob knows Alice's signature, he may not be able to remove it by tampering. In addition, he can also add an entirely new signature. For the re-embedded watermarked design to be useful, the addition generally cannot modify the already watermarked design much and it is unlikely to subvert a large proportion of Alice's watermark. Given that watermarking based IPP cannot stop any one to re-embed another watermark, to prevent Alice's signature from being overthrown by Bob's re-embedded signature, Alice can time-stamp her signature with a third party using a public-key cryptographic protocol.

2.3.4.3 Forging

Bob may attempt to *forge* Alice's signature by inappropriately watermarking other solutions with Alice's watermark. This is possible if Bob is able to generate a signature that can be convinced as Alice's. If a signature corresponds simply to a text message, then Bob may simply choose a text message resembling one that Alice would use. However, such attacks can be easily prevented by using a public-key encryption system [MEN96, KAH01]. Any message actually signed by Alice could only be encrypted with her private key, yet verifiable with her public key. Based on the tenet of public-key cryptography, the private key is not compromised even if the ciphertext that is encrypted with it is compromised. Thus, Alice can still demonstrate the presence of her watermark in her watermarked design to anyone who knows her public key without compromising her private key. In a nutshell, Bob is able to forge a message from Alice only if he knows her private key.

2.4 Static IP Watermarking Schemes

2.4.1 IP watermarking at System Synthesis Level

In [HON98], a scheme was proposed to embed watermark to the system synthesis tasks such as cache line coloring and multiprocessor DSP code partitioning. The example in [HON98] is used to demonstrate the key idea underpinning this scheme of watermarking.

The extra constraints due to watermark are imposed by forcing particular pairs of nodes into the same partition. Suppose the letter "A", encoded as a binary string "1000001" in ASCII code, is to be embedded in a design problem modeled by a graph. Each node in the graph is selected randomly and then numbered sequentially in an ascending order of integers, as shown in Figure 2.2(a). Starting from the node (from among those that have not been set as the origin node) with the smallest number, one of the nodes will be selected to pair with it according to the watermark bit to be embedded. To embed a "0" bit, an even numbered node is selected for pairing. Conversely, to embed a "1" bit, an odd numbered node is selected for pairing. The first feasible node (node that has not been paired) in the increasing order of node

numbers, starting from the node number right after the last paired terminal node, is selected as the terminal node to embed the bit value. A balanced partitioning is made such that the paired nodes remain in the same partition. The balanced requirement is attained with one partition having at most 20% more nodes than the other partition. Consider the graph in Figure 2.2 (a). The pairs of nodes that have to be kept in the same partition are (1,3), (2,4), (3,6), (4,8), (5,10), (6,12) and (7,13). These pairs are connected by dotted lines. The resultant watermarked partition is shown in Figure 2.2(b).

(a) Pairs of nodes with encoded values (b) Partitioned graph with watermark

Figure 2.2 Watermarking on DSP code partitioning.

This scheme can be easily carried out at the system synthesis level. However, the performance overhead caused by watermarking is not evaluated in [HON98]. It is important to restrain the design overhead due to the extra watermarking constraints [YUA06]. If the overhead is high, the watermarked design becomes too burdensome to be useful. Also, it is not stated how a verifier can renumber each node out of numerous nodes of a real design in the same manner as it was numbered during watermarking. If this cannot be performed effectively and precisely, it is hard to demonstrate how the extra constraints are satisfied by the watermarked designs.

This watermarking scheme on graph partitioning can be extended to the application for the graph coloring problem [QU98]. The idea is to label the vertices of a graph with a minimal number of colors such that two adjacent vertices connected by an edge are not labeled with the same color. Some behavioral synthesis tasks have been formulated as graph coloring problems. Therefore, a similar watermarking scheme can be also applied at the behavioral synthesis level.

2.4.2 IP watermarking at Behavioral Synthesis Level

Behavioral synthesis transforms a given behavioral specification into a register-transfer level description that can implement the given behavior. Behavioral synthesis consists of a variety of synthesis tasks such as operation scheduling, resource allocation, binding and transformations.

In [KOU05], the watermarking is applied on the synthesis task of registers allocation. Given the control dataflow graph (CDFG) of a design, the values generated during each control step of the scheduled CDFG must be stored in registers so as to be used in latter steps. The lifetime of a variable is defined as the interval from the birth of a variable to the time its value is consumed. The variables with non-overlapping lifetimes can then be stored in the same register. An interval graph can be constructed with each vertex denoting a variable and an edge between two vertices is used to indicate the overlap of the lifetimes of two variables, i.e., these variables cannot share a register. Register allocation can then be performed by coloring the interval graph [KOU05, QU98]. To watermark the CDFG, extra constraints are imposed by adding extra edges in the interval graph as in [HON98] to force some variables in the CDFG to be stored in different registers, hence a specific solution of register allocation can be obtained. If the probability that another good register allocation method returns the same watermarked solution is sufficiently low, the authorship is proved.

The proposed technique can also be applied to several other behavioral synthesis tasks such as partitioning [KOU03], scheduling [KIR03, KOU03], transformations [KOU03] and template matching [KIR03, KOU03]. A generic approach to perform watermarking at each behavioral synthesis task [KOU03] is summarized in Figure 2.3.

Figure 2.3 Generic approach to watermarking at behavioral synthesis level.

To improve the reliability of the proposed IPP technique, an error detection and correction code [LIN83] was adopted in [KOU03] for the embedded signature. The application of the error detection and correction code provides a mechanism to detect malicious attempts to remove the signature and permit partially modified watermark to be recovered.

In the evaluation of the watermark strength and resilience, the probability of coincidence, P_c and the probability of removing one or more signature bits by modifying the color of one node, P_T were shown to be closely correlated to the length of the signature. According to the derivation of P_c and P_T , the use of more extra colors leads to a smaller P_c but a higher P_T . The experimental results in [KOU03] show that when none or few extra colors are incurred due to watermarking on graph coloring, the approach provides a low P_c and P_T . For a given threshold of P_c and P_T , the overheads in terms of the number of extra colors and the number of parity bits for the error correction codes were found to be acceptably low for reasonably large interval graphs with a large number of variables present in the CDFG.

Another scheme was also proposed to watermark the graph coloring problem [QU98]. It uses the fact that the vertices in one maximal independent set (MIS) can be labeled with only one color. The theoretical framework of these two techniques was established in [QU98]. It proved their capability to provide high credibility watermarked design with at most 1-color overhead for large graphs.

As behavioral synthesis appears higher up in the design hierarchy, the watermark embedded at this abstraction level will carry forward to all the post-processing steps and enables the downstream designs to be protected, from the logically synthesized design to the physical layout design. However, the modification in the register allocation will affect the interconnection of these registers to their inputs and outputs, hence the overheads on the latter designs, such as placement and routing, are not easily assessable and are hard to predict.

No watermarking scheme is complete without a signature detection procedure. As proposed in [KOU03], before the signature verification, it is necessary to collect the information about the relevant structural properties of a design for the complete or partial reconstruction of the design specification at one or more levels of abstraction. Several alternatives were referred by [KOU03] to complete this task, such as reverse engineering, time delays of gate-level signals and energy consumption measurements, accessibility via the *don't care* states in the FSM, etc. Take the last referred alternative as an example. FSMs of many designs typically have one or more *don't care* states. These states can be used to initialize the sequence of state traversals of an FSM. With the knowledge about the FSM controller that reads and writes to the registers, reconstructing the coloring of the conflict graph is an easy task. These methods are feasible theoretically but the time and monetary investments increase rapidly with the size of the design. Besides, some subtle problems should also be considered. For example, reverse engineering may cause the method to be intrusive, as it is inevitable to expose crucial design information in this process. For the access using *don't care* states, the FSM must be available. For moderately large FSMs, it is practically infeasible to extract the FSM from the netlist, since the FSM itself is too large to be represented explicitly [OLI99]. The efficiency and viability of these detection schemes remains a moot question.

2.4.3 IP watermarking at Logic Synthesis Level

Logic synthesis is a process by which an abstract form of desired circuit behavior (typically RTL or behavioral) is turned into a design implementation in terms of logic gates. The first constrained-based watermarking method for IPP at the combinational

logic synthesis level was proposed by [KIR98a].

The goal of logic synthesis at this level is to map the logic network to as few cells as possible from a given cell library. Finding an optimum solution to the technology mapping problem is NP-hard [MIC94]. Consider a simple cell library of two different types of cells in Figure 2.4(a) and a simple logic network of eleven gates in Figure 2.4(b). In this example, the optimum cover uses six cells in the provided cell library while the total number of possible solutions is 49. The nodes of the network are uniquely identified. The data to be embedded is a set of distinct numbers. The additional constraints are imposed by making the output nodes of those gates with the same identifiers as the numbers in the signature set to be the pseudo primary outputs. For example, if the embedded message is $\{3, 4\}$, then node 3 and node 4 will be designated as the additional primary outputs to the synthesis tool. The constrained network can still be solved using six standard cells. However, the total number of possible solutions has reduced to 4. The probability that another algorithm applied to the initial problem specification returns a solution from the set $S_{3,4}$ is equal to $P_c = |S_{3,4}|/|S| = 4/49$. This probability of coincidence is a measure of the strength of the authorship proof. From the experiments of [KIR98a], for real-life examples, very strong authorship can be expected from the infinitesimally small P_c estimated. Owing to the versatility of logic optimization and Boolean matching, low hardware overhead is an important attribute of watermarking at logic synthesis level.

Figure 2.4 An example of watermarking technology mapping solution.

To detect the watermark [KIR06], the function of each node with respect to the primary inputs needs to be computed and compared with the list of functions of those potential nodes in the original design. This establishes the node identifiers for the

design under test. If the nodes corresponding to the pseudo primary output nodes of the original design satisfy the specified watermarked constraints, the authorship is proved. The function of a node is computed recursively from the node functions of its predecessor nodes. Therefore, the computation time increases exponentially with the size of the circuit. Hence, the detection of watermark becomes very time-consuming for large designs and may be impractical for real industrial designs. Under attack, certain nodes may disappear in the pirated design. It was observed in [KIR06] that in the case of a low-effort attack, as many of the nodes are retained in the pirated design, the design, even with several tens of pseudo primary outputs, can produce sufficiently strong proof of authorship. However, how the disappeared nodes will affect the computation of the output functions of nodes present in the design were not discussed. *It should be noted that the embedded watermark will not survive combinational logic re-synthesis.* Since no node can be guaranteed to reappear in the re-synthesized design, the watermark may not be detected by the above procedure.

Another watermarking scheme is proposed in [MEG00] to protect the logic circuit IP while preserving the critical paths in the technology mapping phase. This is achieved by applying the watermark constraints on a set of signals that are not on the critical path. The watermark constraints are introduced by breaking the selected signals into a pair of primary output and primary input signals, in order to produce a unique solution (the solution is deemed to be unique if the IPP has very low P_c) after the technology mapping. The experimental results showed that the watermarked solutions exhibited small area overhead. Unfortunately, the probability of coincidence was not evaluated and no verification process was described. It is unclear how fragile or efficient it is to demonstrate the satisfiability of the watermark constraints.

Because the nodes hosting watermark have been judiciously selected in the non-critical paths, it is easy for an attacker to identify all the nodes in the non-critical paths by timing analysis. It was not discussed how the possible operations on the non-critical nodes will affect the watermark information. When the timing constraints are tight, which means that there are few non-critical paths and few logic gates in them, the probability of removing the watermark by randomly altering the local circuit of non-critical paths becomes high. Similarly, the watermarked design cannot survive combinational logic re-synthesis operation.

Rewiring is a synthesis technique used primarily for circuit optimization tasks like logic optimization, delay optimization, elimination of wires with high load or switching, etc. In [MOI05], rewiring is explored for watermarking.

Given a gate level netlist, a certain number of nodes based on the signature are selected. The redundancy addition/removal (RAR) based rewiring scheme is performed on these nodes to watermark the netlist. Consider the circuit of Figure 2.5 [MOI05]. If a connection “R” from the output of g_5 is added to the input of g_9 (delineated with dashed line), the functionality of the circuit will not change. In other words, “R” is a redundant connection. However, addition of this connection makes two other wires which were originally irredundant to become redundant. These connections and their associated logic are thus removed to reduce the logic complexity of the circuit since the aim of RAR is to perform logic minimization during watermarking without violating the test or timing related specifications. The algorithm processes the circuit nodes one by one to achieve the final watermarked solution.

Figure 2.5 Redundancy addition/removal.

The watermarking scheme can resist several typical attacks [MOI05]. As it is based on the rewiring at the combinational logic synthesis level, the re-synthesis of combinational logic will remove all the embedded information. To detect the watermark, reverse engineering is required to revoke the design to the logic synthesis level. Then, the IP owner needs to show that the original design can evolve into the

revoked watermarked design by repeating the watermarking process on his/her proprietary original design. This is not efficient and may trigger the revelation of some vital design information.

2.4.4 IP watermarking at Physical Synthesis Level

Physical synthesis implements the netlist on a minimal floorplan and maintains or improves quality of physical characteristics, such as timing, power, testability, signal integrity, routability and manufacturability of the design. The optimization tasks aim at enhancing the yields for the device. In the context of physical design, several synthesis tasks were explored for watermarking in [KAH01].

2.4.4.1 Preprocessing of Path-timing Constraints

In the scheme [KAH01], a set of path timing constraints are selected and replaced with “subpath” timing constraints. For example, if a path has the timing constraint as $t(C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow \dots \rightarrow C_{10}) \leq 50$ ns, where C_i is a cell in the path, the timing bound can be allocated between two subpaths and replaced by two constraints $t(C_1 \rightarrow \dots \rightarrow C_5) \leq 20$ ns and $t(C_5 \rightarrow \dots \rightarrow C_{10}) \leq 30$ ns. Other path timing constraints can be further constrained in the similar way. The chance that the original constraint happens to satisfy both of these subpath constraints is at most one-half [KAH01]. The additional timing constraints are transparent to the timing-driven design tools. As only a small part out of a large pool of the timing paths are concerned, this scheme can afford a strong proof of authorship.

This is a preprocessing watermarking scheme, which inserts the extra constraints before the design undergoes the specified synthesis. These constraints will affect the following placement and routing and cause some design overhead. As the path timing constraints are specified in the standard delay format (SDF) file, to verify the watermark, the IP owner must demonstrate how the SDF file is modified according to his/her watermarked constraints and the final design can be generated using the SDF file as the specification of the timing constraints. Thus, the information in SDF is exposed. Alternatively, the designer can analyze the timing performance on the watermark-related path and show that it complies with his/her extra path timing

constraints. Once the watermarked paths are known, it is easy for an attacker to modify (by replacement or routing) them or their related paths to remove the subpath constraints in the similarly watermarked design while maintaining the original constraints. Some reasonably small effort is needed as this operation can be carried out directly on the layout design that is delivered to the IP buyers.

2.4.4.2 Watermarking for Row-based Placement

In this approach [KAH01], some individual cells are constrained to be placed with specified cell row parity, i.e., some cells are constrained to be placed in a cell row with an even index while some other cells are forced to locate in an odd-index row.

This scheme needs just a few constraints to achieve a low probability that the same watermarked placement can be accomplished by mere coincidence. The signature is not easy to be tampered with by local perturbations as many cells are needed to be perturbed to nullify the signature and it is difficult to perturb them without worsening the solution quality. It was not stated in [KAH01] if there is a specific way to select the cells for watermarking. If the cells are selected randomly in this scheme, it is possible for a successful ghost searching attack to be mounted by selecting a group of cells with the cell row indices satisfying the constraints due to his/her watermark. The operation can be easily carried out without touching the design. Likewise, using the Bob versus Alice scenario, Alice cannot convince others of her authorship without a provable way to endorse that the cells are deliberately selected by her. In this case, Bob can deny that Alice has watermarked the design by repudiating that she could simply pick those cells arbitrarily to match her signature.

2.4.4.3 Watermarking at Standard-cell Routing

One approach in [KAH01] is based on adding watermarking constraints on the (per-net) costing of the underlying routing resource. Unusual costs are imposed on “wrong-way” routings and/or via resources for each watermark net so that the watermark nets are provably unusual in their utilization of such resources. This scheme has similar advantages as the above scheme of watermarking for row-based placement. The proving process is a double-edged sword. If the net proving scheme is

open, it is possible for an attacker to apply the proving process exhaustively for each net to find out the unusual ones as long as such process is easy. On the other hand, if such proving process is complex, more effort is needed for the authorship proof.

2.5 Dynamic Watermarking for IP Protection

2.5.1 Dynamic Watermarking at Algorithm Level

In [HON98], it was proposed to embed a covert channel into a DSP design so that only the author of a design can observe and interpret information obtained through the channel. Thus, the signature can be encoded into the information transmitted in the covert channel and then used by the DSP designer to claim his/her authorship. The key idea is illustrated by an example using a digital bandpass filter, where the additional design constraints are added into the filter structure which encodes a designer's signature. By observing the outputs for the specific input segments, the embedded message can be identified.

The technique is attractive as it is carried out at a higher design abstraction level, which makes the embedded information naturally unaffected no matter how many implementation details have been performed by an unauthorized user. However, the method is design-dependent as it is known to be applicable only to the hardware designs which contain the structure of digital filter. With regard to the covert channel, it is a parasitic communication channel that draws bandwidth from another channel in order to transmit information and is hidden within the medium of a legitimate communications channel. As all embedded covert channels draw their bandwidth from a legitimate channel, it reduces the capacity of the latter. The greater the bandwidth used by the covert channel, the more likely it will be "visible" to users of the legitimate channel.

2.5.2 Dynamic Watermarking on Finite State Machine

FSM is a popular target for dynamic watermarking. In [OLI01], the STG is implicitly manipulated so that the sequence of states traversed by an input sequence specified by the watermark exhibits a unique property that is rare in a non-modified STG. All states

and transitions in the STG are duplicated. In addition, some states are duplicated with one edge on it modified such that all newly duplicated states can only be reached from these states. This STG modification can be obtained by a relatively simple manipulation of the circuit that realizes the original FSM. The experimental results show that for circuits of reasonable size, the watermarking procedure has a negligible impact on both the timing performance and the silicon area. There is a low probability that a random unintended watermark is detected in the watermarked design. However, the watermarking process introduces redundant states in the STG, which can be removed if a state reduction operation is applied on the modified STG [OLI01]. The watermark can hence be deleted. When the STG is available to the user, it is vulnerable to the removal attack by the state reduction operation.

Another FSM watermarking scheme was proposed in [TOR00] by inserting redundant transitions into the original STG after the unspecified transitions in the STG are searched and associated with the user-defined input/output sequence. The weakness of this scheme is the monotonous use of only the unspecified transitions with the specified outputs of the STG for watermark insertion. The embedding capacity is limited by the number of free input combinations. For FSM with limited unspecified transitions, the probability of coincidence is high. If the watermark length is increased beyond the available number of unspecified transitions to boost the authorship proof, the overhead aggravates rapidly.

To increase the robustness of FSM watermarking, besides the unspecified transitions, existing transitions are also utilized in an output mapping algorithm to watermark FSM [ABD05b]. This method takes advantage of the original transitions in the STG to lower the overhead of watermarking. The embedding process is fast as no special effort is made to search the states of STG. The watermark bits are embedded at large by a random walk of the STG. When all output bits of an existing transition of a visited node coincide with a substring of the watermark, that transition is automatically watermarked. Otherwise, extra watermarked transition will be added to the STG. A similar algorithm based on the coinciding transitions approach, input comparison algorithm, is also proposed in [ABD05c]. The following example illustrates the output mapping algorithm.

Take the STG in Figure 2.6(a) as an example. Assume a signature sequence of [(11/00), (00/11), (00/01)]. In Figure 2.6(b), starting from the state S_0 of the original STG, a coinciding output “00” is found and moves to S_3 . In S_3 , no coinciding output “11” exists but input “00” is free. A transition “00/11” emanating from S_3 is created, as shown in Figure 2.6(c). The next state in this case will be decided randomly and the algorithm advances to state S_2 . In S_2 , no output coincidence can be found and all inputs are being used. An extra input bit is added to extend the primary input space of the STG. This bit will be forced to be equal to “0” for all existing transitions out of S_2 and “1” for the added transition, “001/01”. This extra transition will drive the STG to a randomly selected state, S_0 in this illustration. This step is shown in Figure 2.6(d).

Figure 2.6 FSM watermarking in [ABD05b].

When the number of outputs of the FSM increases or the FSM is completely specified, output coincidence of existing transition with the watermark bits becomes rare. The watermarked FSM is susceptible to removal attack if the ratio of augmented transitions to coinciding transitions is high. When only unspecified transitions are watermarked, the scheme becomes as vulnerable as [TOR00], as in the benchmark designs of “scf”, “s1488” and “s832” [ABD05b], the removal probability, $P_r^a = 1$. If no unspecified transitions are available for watermarking, a pseudo input variable is added. This input variable is assigned a fixed logic value of “0” for all existing transitions, and a fixed “1” for the added transitions. This discrimination between the existing transitions and added transitions is conspicuous. Moreover, the addition of new input variables with fixed assignments on all transitions increases the decoder logics and hence the overhead of the watermarked FSM substantially.

The main advantage of these approaches is the traceability of the watermarked IP at the lower design levels. Extraction of STG from circuit netlist is computationally impractical for large FSMs [OLI99]. Thus the watermark inherited in the downstream design is resilient against removal attack. The detectability advantage of FSM watermarking ceases when the watermarked FSM is integrated with other circuits in an SoC and packaged.

2.5.3 Dynamic Watermarking at Design-for-Testability Stage

Dynamic watermarking was also applied at the DfT process after the circuit is synthesized and optimized. Kirovski [KIR98b] proposed a protocol for watermarking the selection of scan cells during the generation of test structure. User-specific constraints are used to force some specific scan registers to appear in the final chain, as shown in Figure 2.7. The watermark is detected by iteratively injecting the test vectors into the watermarked IP and the output responses are compared with the unique values generated by simulating the design. Existence of the watermark in the design is proved if these output values are contained in those uniquely selected scan flip-flops. The drawbacks of this scheme are that it requires a readable design to be simulated for authorship verification and it is only applicable to partial scan design. The probability of coincidence and the watermark resilience are not analyzed in

[KIR98b]. As the scan test structure is easily identified from the design and the watermark embedding is independent of the design functionality, the deletion of the entire test scan chain will also remove the watermark on the scan design.

Figure 2.7 Watermarking during partial sequential scan logic test generation.

2.6 Watermarking for FPGA IP Protection

An FPGA is a special semicustom programmable logic device that can be configured by the customer after manufacturing. An FPGA contains clusters of programmable logic components called the configurable logic blocks (CLBs) and a hierarchy of reconfigurable interconnects that allow the CLBs to communicate. The CLBs can be configured to perform complex combinational functions, or primitive Boolean functions. FPGAs can be used to implement any logical function that an ASIC could perform, but the ability to update the functionality after shipping offers advantages for many applications. The IPs designed for implementation on FPGA are typically delivered in the form of configuration files. The uniqueness of a large configuration

bitstream file for a given set of constraints forms the basis of the desired authorship proof.

One method of watermarking an FPGA at the physical level involves manipulating the unused portions of the configuration bitstream. Informed parties can then extract the mark from the bitstream. There is no effect on the function of the design during the insertion or extraction of the watermark because only unused portions of the design are altered. This approach can be implemented through preprocessing, iterative, or post-processing techniques. The advantage of post-processing is that it does not impact other computer-aided design tools and has zero impact on design performance, area or power consumption. The disadvantage of this approach is that the watermark is not embedded in the functional part of the design. Given enough information, the watermark can be removed without affecting the design functionality.

An example of a post-processing approach is given in [KAH01]. In this scheme, the watermark is inserted into the control bits for some unused outputs from the CLBs. Certain bits in the configuration bitstream that control the multiplexers for the CLB outputs can be replaced by the watermark bits if the CLB outputs are not used. The process of watermark insertion in this approach is carried out entirely as a post-processing step and requires very little added design effort. The tool methodically scans the bitstream, searching for the unused outputs by finding the CLB output pinwires that do not attach to any external CLB interconnect. Upon the detection of the unused outputs, the bits of the watermark are inserted in place of the corresponding multiplexer configuration bits. The length of the watermark is limited by the number of bits made available by this approach. Extracting the watermark is an almost identical process. The tool finds the unused CLB outputs in the same way as it was done for the watermark insertion process and pieces the watermark back together by examining the corresponding multiplexer configuration bits. This FPGA watermarking approach requires little extra design effort. It can store fairly large watermarks, allows for easy watermark extraction, and has no overhead in terms of design area or performance. However, because a mark is nonfunctional, it may be removed by reverse engineering a design to a stage in the flow before the mark has been applied.

It is shown in [LE02] that these watermarked IPs [LAC98a, LAC98b] are susceptible to misappropriation by running a deletion algorithm to nullify the creditability of the original watermark. The attack assumes that the cost of reverse engineering is less than the cost of specifying, designing and debugging a circuit from scratch due to the relatively simpler design process of FPGAs. This is rarely the case for complex non-FPGA designs. The time and monetary costs to redo all consequent synthesis and validation steps for the altered design render such attack unworthy.

The scheme proposed in [JAI03] modifies only the net delays in the non-critical path during the place-and-route phase of the FPGA design cycle. This is based on the fact that the timing on the non-critical paths is more flexible and a slight change will not affect the design's overall performance but different delays on paths will force the place and route result to be different. This in turn will require a different set of switches on the FPGA to be opened or closed. Therefore, the configuration bitstream generated by the Xilinx Integrated Software Environment (ISE) software, which programs these switches, will be different. It is shown that changing the delay on only one net will affect a significant number of configuration bits. This produces a rather unique bitstream file as a strong proof of authorship. This scheme is outlined in Figure 2.8. It has zero overhead because the watermark is embedded in the interconnect switches of FPGA devices instead of the configurable logic. While reducing the net delay by less than a hundredth of a nanosecond can create a substantial change in the wiring configurations on FPGAs, the same randomized routings cannot be ensured by the physical synthesis process of other custom circuit designs.

In short, constraint-based watermarking schemes at logic synthesis level can usually be extended to FPGAs by embedding the encrypted signature into unused CLBs [LAC98a, LAC98b, LE02]. On the contrary, watermarking schemes dedicated for FPGA designs [JAI03] may not be applicable to non-FPGA designs.

Figure 2.8 Watermarking on non-critical paths of the FPGAs.

2.7 Hierarchical Watermarking

The notion of hierarchical watermarking was presented in [CHAR98] to describe an IPP scheme for watermarking multiple design levels independently. Hierarchical watermarking is designed to be inherently redundant to fortify against forging and deletion and boost the confidence in positive watermark identification. Since the deletion of a watermark at a certain abstraction level will not impair the watermarks at most other abstraction levels, such scheme is robust.

In [RAS99], a hierarchical watermarking approach is developed which incorporates the watermarking of DSP filter IP cores at both the algorithm level and the architectural level. At the algorithm level, the owner's information is hidden in the filter specification during the development of the filter transfer function. The signature is encoded into the filter's magnitude response by manipulating the magnitude vector within the passband region. At the architectural level, a method is proposed to watermark the transposed direct form finite impulse response (FIR) filter structure, which is a pipelined structure with each pipeline stage consisting of a

multiplier-adder-delay basic block functions [RAS99]. The watermarking is achieved by using the basic blocks of each pipeline stage in different structure but without altering the transfer function of the filter.

Figure 2.9 shows three basic block transformations. In Figure 2.9(a), (b) and (c) the filter function can be expressed respectively as

$$y_j = a_i \cdot x[n-1] + a_j \cdot x[n]$$

$$y_j = a_j \cdot \{a_i / a_j \cdot x[n-1] + x[n]\} = a_i \cdot x[n-1] + a_j \cdot x[n]$$

$$\text{and } y_j = a_i \cdot \{a_j / a_i \cdot x[n] + x[n-1]\} = a_i \cdot x[n-1] + a_j \cdot x[n].$$

It can be seen that the three basic blocks in different structures possess the same transfer function. Thus, they can be used to encode the 2-bit code: 00, 01 and 10, respectively.

Figure 2.9 Architectural level circuit transformations.

Only about seven percent of area overhead is incurred by the watermarking at the algorithmic level. The area cost due to watermarking at both levels is nearly 40 percent. This is regarded as an impractically high performance overhead to the original design. The quality of the watermarked solution will discourage the use of this technique in commercial applications.

The hierarchical watermarking scheme in [CHAR98] extracts the topological features in multiple design levels. With specific topological constraints imposed, the implementation of the design at each level is reduced. The watermarking example shows that this scheme can provide a very low probability that an identical watermark is created by coincidence. There is no account for whether the aggregated overhead

due to the watermarking at multiple levels is acceptable [CHAR98].

For the schemes in [RAS99, CHAR98], it is not stated how the watermarking at the higher level will influence or be influenced by the posterior design processes. Also, the design levels for signature insertion are selected without considering the complementary properties or counteractive constraints of the different design levels in which the watermarks are embedded.

2.8 Watermarking for Design Tools and Algorithms

Conceptually, the tools and algorithms^a used to create an IP can also be protected by constraint-based watermarking in computational forensic engineering. Computational forensic engineering (CFE) is proposed in [WON01] with aims to identify the synthesis tool or compiler that created a particular IP. Rather than relying on watermarking content or designs, the generic CFE methodology analyzes the statistics of certain features of a given IP and quantizes the likelihood that a well-known source has created it. In [WON01], the generic methodology of CFE is described and a set of techniques are presented to identify the one used to generate a particular hardware/software design given a set of compilation tools. The generic CFE approach can be applied in four phases: 1) feature and statistics data collection; 2) feature extraction; 3) entity clustering; and 4) validation. Besides the IP protection, the developed CFE paradigm can have other potential applications: optimization algorithm selection and tuning, benchmark selection, and source-verification for mobile code. However, to protect existing tools and algorithms, they need to be redesigned. Moreover, the quality of the solutions obtained by such protected tools and algorithms cannot be guaranteed [YUA06].

2.9 Summary

In this chapter, background knowledge of constraint-based watermarking and a comprehensive literature survey on different watermarking techniques under this framework are provided.

This chapter has shown by means of a simple illustration on how a generic satisfiability problem can be exploited for constraint-based watermarking for VLSI IP

protection. Due to the nature of VLSI design, the desiderata of watermarking-based IPP are summarized and generic approaches to watermark synthesis and verification for constraint-based watermarking are introduced. The proof of authorship and typical attack scenarios are also analyzed.

From the perspective of watermark detection, constraint-based watermarking has been classified into static watermarking and dynamic watermarking. Different static watermarking schemes at different design abstraction levels have been reviewed. Multiple watermarking approaches at logic synthesis level are elaborated and discussed in greater detail. Due to the prolific number of cells, watermarking combinational circuit has high potential to achieve strong proof of authorship with low overhead. At the time of embarking on this research, it was found that existing watermarking techniques at logic synthesis level did not fully explore the redundancy of combinational circuits and the embedding overhead could be further minimized. The proof requires that the CSP instance be revoked to demonstrate that the additional constraints are satisfied by the watermarked design. This process tends to expose the signature and secrecy of the well-formed grammar used to generate the constraints, making other similarly marked designs vulnerable to attack. Therefore, watermarking at logic synthesis level with better performance and direct detection scheme is to be explored in this research work.

Dynamic watermarking schemes are mainly found in the FSM design and DfT processes. The advantages and disadvantages of different dynamic watermarking schemes have been analyzed. In this thesis, a new dynamic watermarking technique is pursued to overcome the disadvantages of current schemes. Some emerging watermarking schemes that have been proposed for the protection of FPGA IP cores, hierarchical watermarking schemes and IP protection methods for design tools and algorithms have also been surveyed. Hybrid dynamic watermarking scheme is to be studied to combine the merits of both dynamic watermarking schemes and hierarchical watermarking schemes.

Chapter 3

IP Watermarking at Logic Synthesis Level

3.1 Introduction

In most watermarking techniques for combinational circuit designs, as all watermark bits are closely coupled into a set of design constraints, they cannot be independently extracted to detect the locality of design changes due to partial obliteration. A watermark typically contains information about the origin, status, or recipient of the data. Even though not used directly for copy protection, it can help to identify the source and/or destination of the data in suspected copyright violations. While some watermarks are robustly embedded into the host circuit, it is also possible to exploit fragile watermarks that are impaired or destroyed by manipulation for lucrative data authentication. Such idea has been commonly used in watermarking multimedia artifacts but not in VLSI designs.

This chapter presents a new constraint-based watermarking technique at logic synthesis level. The remainder of this chapter is organized as follows. Some definitions on a logic network and timing analysis are presented in Section 3.2 to facilitate the exposition of the proposed watermarking technique. Section 3.3 presents the detailed procedures of watermark insertion and detection. The watermarking strength and its resilience to the typical attacks are analyzed in Section 3.4. Section 3.5 presents the experimental results and the chapter is summarized in Section 3.6.

3.2 Preliminaries

To aid the exposition of the proposed watermark embedding technique, some definitions pertaining to the structural representation of a bound logic network and timing analysis are presented in this section. A bound logic network is a technology

mapped logic circuit. It can be represented as a directed acyclic graph (DAG), $G(V, E)$ where each node, $v \in V$ represents an instance of a cell library. A directed edge, $e_{i,j} \in E$ exists if $\exists v_i, v_j \in V$ such that the output of cell v_i is an input of cell v_j . The sets of fanin and fanout nodes of v_j are denoted by $FI(v_j)$ and $FO(v_j)$, respectively. A *primary input/output* (PI/PO) has no fanin/fanout node.

Definition 3.1: A node u is said to be a predecessor of v if there exists a directed path from u to v . A *cone* at node v , denoted as C_v , is defined as a subgraph consisting of v and at least one of its predecessors. If all the predecessors of v are contained in the cone, C_v is called the *maximum cone* at v [CON00].

Similar notations of nodes are applicable to cones. For example, the set of nodes outside C_v which drive the nodes (or are driven by the nodes) in C_v will be denoted by $FI(C_v)$ (or $FO(C_v)$). Any node in C_v that drives (is driven by) at least a node outside C_v or a PO (PI) is called the output (input) node of C_v .

Definition 3.2: A *simple cone*, C_v has only one explicit output node, v . Any output node of C_v that drives at least one internal node in C_v , if it exists, is known as an implicit output node. A cone is *closed* if it has no implicit output nodes. If a cone contains any implicit output node, it is said to be *open*.

Definition 3.3: A *complex cone*, $C_{v_1, v_2, \dots, v_n} = C_{v_1} \cup C_{v_2} \cup \dots \cup C_{v_n}$ is a conjunction of two or more connected cones with at least one common transitive fanin node between any two cones.

Definition 3.4: The distance, $d(v, u)$ between two nodes, u and v is the minimum number of transitive edges among the paths connecting them. The distance, $d(C)$, of a cone, C , is defined as the maximum distance among all pairs of input and output nodes of C .

A simple open cone can be made closed by either excluding all its implicit output

nodes, or including the transitive fanin and fanout nodes of the implicit output nodes until the complex cone so formed contains no implicit output node.

Figure 3.1 shows the DAG of a logical network. It has seven PIs, i_1 to i_7 and two POs, o_1 and o_2 . By definition, a distance-2 cone, C_v can be obtained by including the node v , and all its predecessors within the distance, which are nodes s , t , p and u . C_v is an open simple cone due to the only explicit output node v and an implicit output node u (Node u is an implicit output node of C_v as it drives an internal node t). The simple cone C_v can be made closed by eliminating u from C_v to obtain the closed cone C'_v . Alternatively, by including the transitive fanout nodes x of the implicit output node u , C_v can be made closed to obtain a complex cone, $C_{v,x}$, which has no implicit output node. MC_v is a maximum cone at node v as all the predecessors of v are contained in the cone.

Figure 3.1 Examples of a DAG network, a maximum cone (MC_v), a closed cone (C'_v), an open cone (C_v), and a complex cone ($C_{v,x}$).

A synchronous logic network needs not be acyclic, but it can be simplified to a number of combination networks when no registers are present, as shown in Figure 3.2. The design is broken into several data paths. Each data path has a start point and an end point for timing analysis [SYN04]. The former includes the input ports and clock pins of registers, while the latter includes output ports and all input pins except the clock pins of sequential devices. The data paths are grouped by the clocks controlling their endpoints and the default data path group contains all data paths not

captured by a clock. In this way, a synchronous design can be modeled by a multi-graph, where different timing constraints can be imposed on the output nodes of default data path group, and other data paths are constrained by their respective clock periods.

Figure 3.2 Required time for the arrival of a signal at a node.

The watermark information is embedded into a design by exploring the timing path slacks in a bound logic network from the initial timing specifications. The following definitions are adopted from [JOU98] for timing slack analysis.

Definition 3.5: The *arrival time* of the signal at node v , denoted by $t_a(v)$, is the latest time when the signal at node v reaches its final stable state.

If $t_d(u, v)$ is the delay from node u to node v , where $u \in FI(v)$, then $t_a(v)$ can be calculated recursively from the maximum cone of v .

$$t_a(v) = \max_{u \in FI(v)} [t_a(u) + t_d(u, v)] \quad (3.1)$$

where $t_d(u, v)$ of each cell v can be obtained from the target technology library.

Definition 3.6: The *required time* of the signal at node u , denoted by $t_r(u)$, is the earliest time when the signal at node v is required to be stable to meet a given timing specification.

$t_r(u)$ can also be calculated recursively from the fanout nodes of u given the timing constraint of the data path involving the node u .

$$t_r(u) = \min_{v \in FO(u)} [t_r(v) - t_d(u, v)] \quad (3.2)$$

Definition 3.7: The *slack* of a node v , denoted by $t_s(v)$, is the difference between the required time and the arrival time of the signal at node v , i.e.,

$$t_s(v) = t_r(v) - t_a(v) \quad (3.3)$$

The slack of a node, v , indicates the delay margin of the most critical path in the maximum cone at node v . The smaller the slack of a node, the more critical is the slowest path to this node in the maximum cone.

3.3 Watermark Insertion and Detection

3.3.1 Watermark Insertion

To mimic the non-oblivious media watermarking scheme [CHAN05], a master design is used for the insertion and detection of the watermark. A master design is an originally optimized circuit generated by a logic synthesis tool with a cell library. The four major steps involved in the adaptive watermarking method are elaborated as follows:

3.3.1.1 Watermark Encryption

The watermark, W is a meaningful text string or logo. This arbitrary length binary string is encrypted using a public-key cryptosystem (PKC). If the encrypted watermark exceeds the embedding capacity, it is reduced by a message digest (MD) to a constant length sequence.

$$W_e = MD(PKC(W, K)) \quad (3.4)$$

PKC and MD disperse the spatial relationship of the watermark information, making the encrypted watermark W_e resemble random noise with balanced number of ones and zeros. The advantage of using a PKC is the ease of generating and distributing

authenticable time-stamped signatures by a secure third party to circumvent the re-embedding attack [ABD05b].

3.3.1.2 Generation of Disjoint Closed Cones

To minimize and isolate the topology perturbations to small regions of selected cells in incremental technology mapping, the cones selected for watermark embedding must be disjoint and closed. There must be at least one substitutable template for any cell in a selected cone to enable remapping but more substitutable templates help to preserve or improve the quality of the remapped solution. Besides, there must be sufficient number of such local closed cones that are spatially uncorrelated to reduce the odds that a randomly selected cone coincides with those that host the watermark. In these perspectives, a single cell is a trivial candidate and offers limited optimization. A single cell tends to be translated into more than one cell. Such redundancy is readily detectable, causing the embedded watermark bit susceptible to erasure. A maximum cone or large cone is not a good choice either, as it limits the circuit embedding capacity. The larger the cone, the higher the probability that the watermark bit can be corrupted by random attack. To provide a pool of fair candidates for selection, each cell in the master design is first expanded to a disjoint closed cone of distance- τ , where τ is empirically determined based on the complexity of the circuit, the length of watermark, and the resources in the cell library.

Figure 3.3 shows the procedure to expand a cell, v , into a simple or complex closed cone of distance- τ . It can be implemented by a depth first traversal from v to include all its predecessor nodes into C_v until the distance from the predecessors to v is τ (line 3), the primary input is reached, or when an implicit output node (see Definition 3.2) is met. Once a cell is included in the closed cone (line 6), it will be masked (line 5) to refrain it from selection into an adjacent closed cone. When an implicit output node is encountered (line 9), a decision has to be made to determine if a simple closed cone excluding the implicit node, or a complex closed cone covering the implicit node and all its fanout nodes, is to be generated. As multi-vertex mapping requires more effort to preserve the timing constraints, to justify for a complex cone, all implicit output nodes of the closed cone must have greater slack than node v . Therefore, a complex

cone, $C_{v_1v_2\dots v_n}$ of distance- τ is generated only if the following criterion is met.

$$t_s(v_i) > t_s(v_1) \quad \forall v_i \in FO(C_{v_1v_2\dots v_n}), v_i \neq v_1 \quad (3.5)$$

where v_1 is the cell to be expanded into a distance- τ closed cone. The function **unmask**(u) in line 11 is used to mark the release of the node, u from the closed cones of C_v . Trivial closed cones containing only one cell or only inverters are discarded, as indicated in line 23.

```

1: disjoint_closed_cone( $v, \tau$ ) {
2:    $C_v = \{v\}$ ;
3:   for each ( $u \in FI(C_v)$  and  $d(u, v) \leq \tau$  and  $u$  is unmasked) {
4:     if ( $u \neq PI$ ) {
5:       mask( $u$ );
6:        $C_v = C_v \cup \{u\}$ ;
7:     }
8:     flag = 1;
9:     for each ( $x \in FO(u)$  and  $x \notin C_v$ ) {
10:      if ( $t_s(x) \leq t_s(v)$  or  $x$  is masked) {
11:        unmask( $u$ );
12:         $C_v = C_v - \{u\}$ ;
13:        flag = 0;
14:        break; // the loop of "for" is terminated
15:      }
16:    }
17:    if (flag = 1) {
18:      mask( $FO(u)$ );
19:       $C_v = C_v \cup FO(u)$ ;
20:    }
21:  }
22:  if ( $C_v$  is trivial)
23:    return  $\emptyset$ ;
24:  else
25:    return  $C_v$ ;
26: }
```

Figure 3.3 Generation of distance- τ disjoint closed cone.

Using the DAG of Figure 3.1 as an example, if $\tau = 2$, node v can be expanded into either a simple closed cone or a complex closed cone. The complex closed cone of Figure 3.4(b) is generated if the slack of node x is greater than the slack of v . Otherwise, node u is excluded and a simple closed cone of Figure 3.4(a) is generated even though u is distance-2 from v .

Figure 3.4 Generation of (a) simple and (b) complex closed cones.

3.3.1.3 Qualification of Closed Cones

When applying incremental technology mapping, it is imperative to maintain the quality of the remapped design to be as close as possible, if not better than the master design. The locations of the disjoint closed cones and the number of closed cones to be modulated are influential to the remapped result. The number of closed cones is dictated by the length of the signature. The embedding capacity is limited by the ability of a design to preserve its timing constraints in remapping. To maximally utilize the available embedding capacity for robust watermarking, a metric is developed to formally qualify the disjoint closed cones generated for watermark embedding.

The slack of a cone, C_v , is defined as the slack of its explicit output node, $t_s(v)$. It provides a global view of the margin available for remapping the cells along the critical path of the max cone rooted at v . However, a distance- τ closed cone, C_v , can preclude cells in the critical path of the max cone of v that are further than distance τ from v . To determine how well a closed cone, C_v preserves its output timing, only C_v of G can be remapped to obtain a remapped solution, G' . The *sustainability* of a closed cone C_v is defined as:

$$S(C_v) = 1 + \frac{\Delta t_a(v)}{t_a(v)} = 1 + \frac{t_a(v) - t'_a(v)}{t_a(v)} \quad (3.6)$$

where $t_a(v)$ and $t'_a(v)$ are the arrival times of the same node v in G and G' , respectively. $\Delta t_a(v)$ is the difference in the critical path delays of the maximum cone of v before and after remapping. A negative difference means the critical path is retarded but it does not necessarily imply that the original timing constraint is violated. A positive difference implies an improvement in timing performance by remapping the cone. If $\Delta t_a(v) = 0$, there is no change in the critical path delay. A bias of one is added to the fractional delay in (3.6) so that when $S(C_v) = 1$, the original slack is retained, hence the term slack sustainability. Since it is unlikely that an incremental technology mapping doubles the critical path delay of a remapped cone, it can be derived from (3.6) that $S(C_v) \geq 1 + \frac{t_a(v) - 2 \cdot t_a(v)}{t_a(v)}$, i.e. $S(C_v) \geq 0$. Timing is improved by remapping if $S(C_v) > 1$ and aggravated if $S(C_v) < 1$.

A re-synthesis is needed for each closed cone to evaluate its sustainability. The computation for all disjoint closed cones in a design by exhaustive simulation is very time consuming and impractical. To increase the efficiency, a correlated metric to the sustainability is statically derived from the master design without repeating the physical re-synthesis process. To account for the fractional delay that a cell v contributes to its critical path through its fanin node, $u \in FI(v)$, the following term is defined.

$$\delta(v, u) = \frac{t_d(v, u)}{t_a(u) + t_d(v, u)} \quad (3.7)$$

where $t_d(u, v)$ is the delay from node u to node v .

Ignoring the constant bias, the slack sustainability, $S(C_v)$ can be approximated by a recursive evaluation of δ as follows:

$$\hat{S}(C_v) = \hat{S}(v) = \frac{1}{|FI(v)|} \sum_{u \in FI(v)} (\hat{S}(u) + \delta(v, u)) \quad (3.8)$$

where $|FI(v)|$ is the number of fanin nodes of v . $0 < \hat{S}(C_v) \leq 1$. The recursion of $\hat{S}(u)$ terminates at $u = FI(C_v)$, and $\hat{S}(u) = 0 \forall u \in FI(C_v)$. If C_v is a maximum cone, $\hat{S}(C_v) = 1$.

The higher the sustainability, the better the closed cone preserves its timing slack upon remapping. The slack, $t_s(v)$ is reciprocal to the criticality of C_v . The greater the slack, the easier the delay constraint is met by remapping. When two cones possess the same sustainability, the cone with the greater slack is less time critical and has more headroom for remapping. The headroom of a closed cone is thus defined as:

$$w(C_v) = t_s(v) \times \hat{S}(C_v) \quad (3.9)$$

This headroom is used to qualify closed cones for watermark embedding. An empirical threshold, w_T is defined such that all cones with headroom less than w_T are pruned. To reduce the probability of qualifying closed cones that could adversely affect the performance of the watermarked design, the threshold is judiciously set to α time the minimum slack of all path groups of the master design, G .

$$w_T = \alpha \times \min_i \{t_s(PO_i(G))\} \quad (3.10)$$

Since the central value of $\hat{S}(C_v)$ is 0.5, $\alpha \geq 0.5$ to minimize the degradation to the overall circuit performance. As the excess slacks on less critical path groups are expected to be traded for the diminished slacks on more critical path groups, α provides the trade-off control of the embedding capacity of a design and its watermarked solution's timing specification. In other words, (3.9) provides a design-dependent quantification of watermark embedding capacity based on the slack "imperceptibility".

The pseudo code of Figure 3.5 computes the headroom of each disjoint closed cone in the list Q and prunes those with headroom less than w_T , as indicated in lines 3 and 4 of the function **qualify_closed_cone**.


```

1: qualify_closed_cone( $Q, w_T$ ) {
2:   for each ( $C_v \in Q$ ) {
3:      $w(C_v) = t_s(v) \times \text{sustainability}(v, C_v)$ ;
4:     if ( $w(C_v) < w_T$ )  $Q = Q - C_v$ ;
5:   }
6:   return  $Q$ ;
7: }

1: sustainability( $v, C_v$ ) {
2:   if ( $v = FI(C_v)$ ) return 0;
3:   for each ( $u \in FI(v)$ ) {
4:      $\delta(v, u) = t_d(v, u) / (t_d(u) + t_d(v, u))$ ;
5:     return {sustainability( $u, C_v$ ) +  $\delta(v, u)$ } /  $|FI(v)|$ ;
6:   }
7: }

```

Figure 3.5 Pruning of disqualified disjoint closed cones.

3.3.1.4 Insertion of Watermark

To embed a logic “1” of the watermark, a selected closed cone is coerced to remap by prohibiting the use of one randomly selected template inside the cone. To embed a logic “0”, one template of the selected cone is preserved. To obfuscate the locality of the watermark, each qualified closed cone with $w(C_v) \geq w_T$ is equally probable to be selected by the signature to host either a “1” or a “0” bit. Every qualified closed cone is identified by a unique identifier of its explicit output node in the master design. The random selection of m qualified closed cones is performed by a cryptographically strong pseudorandom number generator seeded with the m -bit encrypted watermark, W_e .

Figure 3.6 shows the algorithm for watermark insertion. The input is a copy of a master design, G , a cell library, Lib , an m -bit watermark, W , a public encryption key, K , and two controlling parameters, τ and α . τ controls the size of the closed cones, and α controls the embedding space, as indicated in line 1. Small α increases the obscurity for a given signature length but if α is too small, it is difficult to preserve the slack of the master design. The initialization indicated in line 3 will select the root node for the development of the 1st cone in the set of closed cones, Q . Other cones in Q are generated from each vertex of G using the function **disjoint_closed_cone** of

Figure 3.3. The threshold, w_T , computed by **threshold**, in line 11, is used by **qualify_closed_cone** of Figure 3.5 to prune the unfit disjoint closed cones from Q . The unique identifiers of the survival cones are stored in a list id . The watermark, W is public-key encrypted using the key pair, K and then message digested into W_e in line 13. The encrypted signature, W_e , is used to generate m additional constraints and each constraint involves either coercing a change in one template of a selected cone or retaining a designated template in it. A keyed one-way function, **pseudo_random_generator** in line 16 is used to generate an index between 1 to q to select the first cone for embedding the first watermark bit. Any arbitrary message can be used as the starting seed. The identifier of the first selected cone is then used as a seed to generate another random index to select the next cone for the second watermark bit. This process continues until all watermark bits have been embedded. The keyed one-way function ensures that with high probability a different set of closed cones will be selected when the bitstream in W_e is permuted. The selected 1- and 0-watermarked cones are stored in Q_1 and Q_0 (lines 20 and 24), respectively, and their designated forbidden and preserved templates are stored in TP_1 and TP_0 (lines 19 and 23), respectively for watermark recovery. The watermarked solution, G' is generated by the function **remap** in line 28, which calls the synthesis tool to perform a technology mapping of only Q_1 and Q_0 according to their template constraints, TP_1 and TP_0 .

```

1: watermark_insert( $G, Lib, W, K, \tau, \alpha$ ) {
2:   Compute and store timing information of  $t_s, t_a$  in  $G$ .
3:    $i = 1$ ; Initialize disjoint cone array,  $Q$ ;
4:   for each ( $v \in G$ ) {
5:      $Q[i] = \text{disjoint\_closed\_cone}(v, \tau)$ ;
6:     if ( $Q[i] \neq \emptyset$ ) {
7:        $\text{cone\_id}[i] = \text{identifier of } v$ ;
8:        $i = i + 1$ ;
9:     }
10:  }
11:   $w_T = \text{threshold}(\alpha, G)$ ;
12:  qualify_closed_cone( $Q, w_T$ );
13:   $W_e = MD(PKC(W, K))$ ;
14:   $m = |W_e|$ ;  $q = |Q|$ ;
15:  Initialize arrays  $TP_1, TP_0, Q_1$  and  $Q_0$  to  $\emptyset$ ;
16:   $\text{index}[1..m] = \text{pseudo\_random\_generator}(W_e, q)$ ;
17:  for ( $i = 1$  to  $m$ ) {
18:    if ( $W_e[i] = 1$ ) {
19:       $TP_1 = TP_1 \cup \text{any template in } Q[\text{index}[i]]$ ;
20:       $Q_1 = Q_1 \cup Q[\text{index}[i]]$ ;
21:      mark the selected  $Q[\text{index}[i]]$  and its forbidden
        template in  $G$ ;
22:    } else {
23:       $TP_0 = TP_0 \cup \text{any template in } Q[\text{index}[i]]$ ;
24:       $Q_0 = Q_0 \cup Q[\text{index}[i]]$ ;
25:      mark the selected  $Q[\text{index}[i]]$  and its preserved
        template in  $G$ ;
26:    }
27:  }
28:   $G' = \text{remap}(G, Q_1, Q_0, TP_1, TP_0, Lib)$ ;
29:  return  $G'$ ;
30: }

```

Figure 3.6 Watermark insertion by incremental technology mapping.

3.3.2 Watermark Detection

To enact that the presence of the watermark in an IP is a consequence of some deliberate actions, the original CSP instance is recalled to verify that the additional constraints imposed by the signature are satisfied by the watermarked design [KAH01]. This process exposes the well-formed grammar used to generate the constraints, which could weaken other watermarked designs. This is unlike the detection procedure of media watermarking, where the encrypted watermark is physically recovered from the watermarked design by a reverse process. Such

recovery does not, in principle, pose a threat to another media marked with the same signature and algorithm.

To augment the protection model against piracy, another new watermark detection method is presented, which possesses some features of fragile watermarking. Other designs similarly marked with the same signature will not become more vulnerable by the information divulged in this verification process. If the marked design has not been tampered, the recovered watermark will perfectly match the embedded one. The high degree of correlation between the recovered and embedded watermarks provides the design authenticity even if the design has been partially modified. Thus, the detection process can also track the intentional tampering or unauthorized changes to the watermarked object, if any.

To facilitate the retrieval of the watermarked closed cones, correspondences between internal nets of the watermarked design and designated nets of the master design are identified by functional equivalence. The logic cones, C' and C of two networks, G' and G , where $C' \in G'$, $C \in G$, are said to be equivalent if and only if there exists a bijection between $FI(C')$ and $FI(C)$, and between $FO(C')$ and $FO(C)$. Let $f(v)$ be the logical function of cell v , then $C' \equiv C$ iff $|FI(C')| = |FI(C)|$ and $|FO(C')| = |FO(C)|$, and $\exists u \in FI(C')$ such that $f(u) \equiv f(v) \forall v \in FI(C)$, and $\exists u \in FO(C')$ such that $f(u) \equiv f(v) \forall v \in FO(C)$, where “ $\equiv$ ” denotes the equivalence of two logical functions.

Since incremental technology mapping preserves the functionality of the interface ports of the remapped cones, the logic functions of these nets can be retrieved from the fanin and fanout nodes of the closed cones saved in the master design, G . To recover the watermarked cone C' from a marked design, G' , nodes with the same logic functions are extracted. Let $C' \equiv C$ be a cone extracted from the watermarked design, G' with n fanins and k fanouts. $f(WP_i) \equiv f(P_j)$ for $i, j = 1, 2, \dots, n+k$, where WP_i and P_j are the interface ports of C' and C , respectively. If C is used to embed a logic “0” and the designated template is found in C' , a valid logic “0” is retrieved. If C is used to embed a logic “1” and the designated template is not found in C' , a valid logic “1” is recovered. Otherwise, the embedded bit has been corrupted. When equivalent fanin and fanout nodes of a watermarked cone cannot be found, it implies that either

the cells within the cone or its neighboring cells have been modified. When all watermarked cones in G' have been checked, the detection process terminates. The authorship is proved by a perfect or high match between the recovered bit stream and the embedded watermark, W_e .

3.4 Watermark Strength and Resilience Analysis

3.4.1 Strength of Watermark

The strength of a watermarking scheme is evaluated by the probability of coincidence, P_c , which depends on the design and how the design space is explored to solve the chosen CSP by the design tools. It is difficult to unambiguously or unconditionally measure the credibility of IC IPPs in practice. Therefore, the probabilistic analysis conservatively considers only the potential high quality solutions that are less likely to violate the design constraints. The closed cones that span this subspace have been qualified by the threshold, w_T in this watermarking flow. Thus, the probability of m uniquely labeled closed cones selected from q closed cones that satisfy the headroom constraints is given by $1/C_m^q$. Let p_0 and p_1 be the probabilities that a template in the selected closed cone is preserved or changed in the remapping process, respectively. Assume that W_e has an equal number of “1” and “0” bits, the probability that a solution carries the watermark by coincidence is given by:

$$P_c = \frac{1}{C_m^q} (p_0)^{\frac{m}{2}} (p_1)^{\frac{m}{2}} \approx \frac{1}{C_m^q} \left(\frac{1}{2}\right)^m \quad (3.11)$$

Given a reasonably abundant cell library and a small number of different templates in a small closed cone, it is assumed that it is equally probable that a designated template is extricated or preserved when a closed cone is remapped. Hence, $p_0 = p_1 \approx 0.5$.

Equation (3.11) shows that the strength of the watermarking scheme is dependent on m and q . It is obvious that increasing the length of the signature increases the strength of the watermark but this is possible only if the design itself offers sufficient redundancy. Given a fixed signature length, m , a high exploitable redundancy space, q , also increases the strength of the watermarking scheme, provided that α in (3.10) is judiciously selected to preserve the original timing constraints. If manufacturing

variability or a lax in the device and process models causes the constraint to be tighter than originally specified, q becomes smaller than analyzed, and P_c increases. Thus, experienced designers can leverage on α to indirectly modulate the timing safety margin based on the observed discrepancy between the simulated result and actual measurement for a given design tool and cell library.

3.4.2 Watermark Resilience Analysis

Since the design is watermarked at the logic synthesis process, the watermark carries over to the posterior processes of placement, routing and physical layout. The marked design is made available as a technology-specific mapped firm or hard IP. As with most combinational circuit watermarking method, the watermark will not be guaranteed to exist upon radical re-synthesis of the entire circuit. Information that can be directly exploited for re-synthesis without substantial design and verification effort shall not be revealed. Depending on the type of IP, the watermarked design can be distributed with some or all of the followings: a high-level behavioral model, a full or partial netlist, a floorplan, a test list, physical and timing models and final layout but not the RTL code. To analyze how well the proposed watermarking scheme stands up to the following attacks, the IP attacker, Bob is assumed to have illegally acquired a watermarked IP produced by Alice. To comply with the fundamental tenet on IPP, any attempt to obliterate the watermark shall result in a task of difficulty equal to a complete repetition of the specified optimization task [VSI97]. This will exclude reverse engineering of the complete circuit for radical re-synthesis.

3.4.2.1 Watermark Denial

Bob denies the existence of Alice's watermark. This implies that Alice saw the IP and was able to claim her ownership with the proposed watermarking scheme. Bob's claim cannot be substantiated. First, Alice can prove that she could generate the marked IP from her master copy. If there is no watermark, Alice must have reverse engineered this process to create her master copy. Since the closed cones that host the "1" and "0" bits are ordered by a keyed one-way function, it is difficult to create a master copy that can map to a correct sequence of watermarked cones without Alice's key. Even if Alice can assume some watermarked cones to reverse engineer a master

copy, she has no control over the embedded bit sequence obtained from the forged watermarked cones. Since this bit stream is produced by (3.4), it is computationally infeasible for Alice to generate her signature from this sequence. If Alice did not mark the design, she must have also reversed the optimization process to obtain the original design from the master design. Again, this effort is at least as hard as the complete redesign of the specified functionality.

3.4.2.2 Watermark Removal

Bob tries to erase Alice's watermark from her watermarked netlist. To do so, Bob has to search for the remapped closed cones. He could either manipulate or replace the cells in the suspected cones or change the logic functions of their fanin and fanout nodes without violating the functionality or unduly degrading its performance. Node manipulation is not easy as it will affect the circuit timing even if Bob can reverse engineer parts of the circuit netlist to recover some local functions with reasonable effort. If Bob tries to add or replace cells or nets without changing circuit function at only the non-critical paths based on his assessment, it is unlikely that he could successfully erase a sufficient number of watermark bits to lower the authorship proof below a predetermined standard due to the slack sustainability which is defined to quality closed cones for watermark embedding. To delete a large portion of the watermark bits and still preserve the solution quality, Bob has to substantively perturb the watermarked netlist which results in a task of effort comparable to complete circuit optimization.

With no obvious cue of the watermark locality, Bob will manipulate a limited number of circuit nodes at random to remove the watermark. Assume that Bob can perform a brute-force attack to alter γ cells followed by a timing analysis to ensure that the constraints are still satisfied with reasonable effort and without changing the functionality of the circuit. To simplify the analysis, it is conservatively presumed that Bob needs only to alter any cell of a watermarked cone to successfully erase one bit of watermark. Thus, the probability that he will succeed in erasing i watermark bits is given by:

$$\begin{aligned}
P_r(n=i) &= C_i^\gamma \underbrace{\left(\frac{m}{N}\right)\left(\frac{m-1}{N-1}\right)\dots\left(\frac{m-i+1}{N-i+1}\right)}_i \\
&\quad \underbrace{\left(\frac{N-m}{N-i}\right)\left(\frac{N-m+1}{N-i-1}\right)\dots\left(\frac{N-m-\gamma+i+1}{N-\gamma+1}\right)}_{\gamma-i} \\
&= C_i^\gamma \prod_{j=0}^{i-1} \left(\frac{m-j}{N-j}\right) \cdot \prod_{j=0}^{\gamma-i-1} \left(\frac{N-m-j}{N-i-j}\right)
\end{aligned} \tag{3.12}$$

where m is the bit length of the encrypted watermark. N is the number of disjoint cones in the watermarked design. When $m \ll N$, the probability can be approximated by a simple binomial.

$$P_r(n=i) \approx C_i^\gamma \left(\frac{m}{N}\right)^i \left(1 - \frac{m}{N}\right)^{\gamma-i} \tag{3.13}$$

Therefore, the probability of removing more than k ($k \leq m$) watermark bits is given by:

$$P_r(n \geq k) \approx \sum_{i=k}^m C_i^\gamma \left(\frac{m}{N}\right)^i \left(1 - \frac{m}{N}\right)^{\gamma-i} \tag{3.14}$$

3.4.2.3 Watermark Addition

Bob tries to embed his watermark in Alice's watermarked design. Bob will search for a ghost signature, which he can use to modulate several closed cones in the marked design without changing the functionality or severely violating the timing constraints. If Bob succeeded in re-embedding his signature, the master copy from which Bob used to retrieve his watermark will contain Alice's encrypted watermark with which Alice could prove her ownership. Conversely, Bob can never prove his ownership from Alice's master copy since the proprietary master copy kept by Alice is inaccessible to Bob. On the other hand, if Bob creates his own master copy to forge the watermarked design, he must ensure that his master copy contains no trace of Alice's signature to repudiate Alice accusation. This reverts to the same scenario of removal attack.

The use of public-key cryptography to encrypt the signature also helps to solve the re-embedding problem, if a trusted agency time-stamps Alice's signature. Alice could

sign her ownership information using her private key and send this signature to the agency for time-stamping. The agency decrypts Alice's signature using her public key and then generates a message digest of the time-stamp signature for Alice to embed it into her design.

3.5 Experimental Results

In the following experiments, Synopsys Design Compiler and Artisan SAGE-Modeler standard cell library for the TSMC 0.18 μ m process are used to synthesize the circuits from ISCAS85, ISCAS99 and LGSynth93⁴ benchmark suites. All experiments were run on a 750-MHz Sun UltraSPARC-III with 2 GB of memory running Solaris operating system. The master design of each circuit was obtained by using the timing constraints derived by the synthesis tool from the initially unconstrained designs. Table 3.1 shows the area in μm^2 , the critical path delay in ns and the power consumption in mW of the master design. The last three columns show the total number of combinational cells, number of disjoint closed cones and number of qualified closed cones with $\tau=1$ for each design.

The first 16, 32, 64 or 128 bits of the MD5 hash value of the signature were inserted into each master design. The average results were reported by repeating the experiments for 100 different signatures. The amount of watermark bits embedded was determined by the embedding capacity according to the value of q . For the two large designs, ex1010 and B22, $\tau=2$ was also experimented with 64-bit and 128-bit signatures. When $\tau=2$, $N=q=1082$ for ex1010 and $N=1172$, $q=1150$ for B22. Tables 3.2, 3.3 and 3.4 show the area (A), delay (D) and power (P) values of the watermarked designs from different benchmark suites. ΔA , ΔD and ΔP are calculated as the percentage increases in the area, delay and power, respectively due to watermarking. The columns " P_c " and " P_r " correspond to the probability of coincidence and the probability of successfully removing more than three quarters of the watermark bits by randomly altering m cells, i.e., $k=3m/4$, $\gamma=m$.

⁴ These three benchmark suites include many typical combinational designs in different sizes. They have been widely used in many references [KIR06, MEG00] on watermarking at combinational logic synthesis level.

For a given master design, when more watermark bits were inserted, higher overheads were incurred. It is observed that as the number of combinational cells increases, the overheads diminish and become negligible. From Table 3.3, it is also observed that the area and power overheads of the 128-bit watermarked “pdc” were both less than 2.6%, and the timing was actually improved (negative percentage difference). For an even larger circuit “B22”, the area and power overheads reduced to only 0.07% and 0.11%, respectively, while the delay increased slightly by 1.87% for $\tau = 1$. For $\tau = 2$, the watermark introduced only a negligible 0.01% area overhead and the timing was improved. If a design can tolerate a higher overhead, more information can be inserted for a stronger proof of authorship. For most designs, as more watermark bits were inserted, lower P_c was obtained. If there are more than 1000 qualified cones in a master design and a 128-bit watermark is embedded, $P_c < 10^{-200}$. It is more difficult to remove a sufficient number of watermark bits successfully from a larger design. The results show that when a design possesses more than 600 disjoint cones, $P_r < 10^{-50}$ for $m = 128$. It has also been experimented with $k = m/2$. It is found that the attacker can easily remove at least half of the watermark bits for a small number of watermarked circuits with $N/m \leq 2$ and the P_r values remain very low when $N/m > 3$. When the design possesses more than 700 disjoint cones, the probability of successfully removing more than half of the 128 watermark bits is less than 10^{-19} , for example, $P_r = 6.52 \times 10^{-24}$ for “elliptic” and 2.71×10^{-37} for “ex1010”. τ can be increased to increase the probability of lowering the overhead of a watermarked solution. However, as τ increases, the number of qualified disjoint closed cones is reduced and P_c and P_r become higher.

Table 3.1 Performance reference of master designs.

Benchmark Set	Circuit	Area	Delay	Power	#cells	N	q
ISCAS85	C2670	7218	2.84	8.23	430	92	92
	C3540	7158	3.22	8.01	465	93	51
	C5315	12790	2.35	17.25	721	125	125
	C7552	16914	4.62	24.05	1069	243	243
	C6288	26006	9.37	38.85	1979	466	145
LGSynth93	i7	3972	1.38	4.13	269	61	61
	i2	2009	0.53	0.95	190	57	57
	i9	3815	1.58	3.77	239	30	30
	alu4	12015	2.03	8.97	748	122	120
	frg2	5801	1.71	4.57	389	58	58
	rot	5113	2.44	4.55	374	76	76
	apex6	5412	1.25	4.30	381	72	72
	x3	5761	0.96	4.70	430	96	91
	k2	7431	2.15	3.11	542	100	76
	i8	6443	1.94	6.00	416	80	74
	dalv	5292	2.53	3.00	356	93	93
	pair	13645	1.43	10.55	1087	314	314
	des	25517	1.85	27.73	1680	267	173
	i10	18448	4.15	13.49	1431	335	335
	apex5	6280	1.25	3.67	420	78	78
	mm30a	15704	2.16	1.64	844	215	215
	pdc	33384	3.24	6.87	2288	452	403
	spla	29336	2.72	6.76	2015	364	265
	ex1010	85951	2.37	8.79	5411	1316	1316
	elliptic	115586	5.09	13.48	3626	793	682
ISCAS89	S5378	20787	1.35	12.40	734	140	84
	S9234	24516	2.17	6.64	896	191	187
	S6669	44890	10.18	2.37	2071	603	508
	S13207	56130	2.40	17.12	1619	370	363
	S38417	160775	3.92	31.67	4927	1223	1127
	S35932	176356	1.94	58.37	5185	805	758
	S38584	170575	3.31	48.38	6520	1204	1141
ISCAS99	B22	233759	12.33	9.37	5499	1313	1290
	B21	155160	12.33	4.90	3605	886	875

Table 3.2 Watermarking results on ISCAS85 benchmarks.

Circuit	m	A	D	P	$\Delta A(\%)$	$\Delta D(\%)$	$\Delta P(\%)$	P_c	P_r
C2670	32	7297	2.78	8.31	1.09	-2.11	0.97	4.10E-35	4.97E-09
	64	7486	2.79	8.44	3.71	-1.76	2.55	1.69E-43	0.07
C3540	16	7242	3.23	8.11	1.17	0.31	1.25	2.13E-18	6.58E-09
	32	7327	3.22	8.20	2.36	0.00	2.37	4.80E-24	3.75E-09
C5315	32	13198	2.34	17.56	3.19	-0.43	1.80	3.76E-40	1.78E-12
	64	13476	2.35	17.95	5.36	0.00	4.06	1.85E-56	3.81E-08
C7552	32	17082	4.60	24.22	0.99	-0.43	0.71	2.37E-50	9.25E-20
	64	17216	4.47	24.28	1.79	-3.25	0.96	1.33E-79	3.30E-23
	128	17581	4.46	24.74	3.94	-3.46	2.87	5.75E-111	9.30E-14
C6288	32	26138	9.41	39.12	0.51	0.43	0.69	1.70E-42	9.90E-27
	64	26341	9.47	39.56	1.29	1.07	1.83	4.95E-62	1.61E-37
	128	26425	9.49	39.68	1.61	1.28	2.14	5.01E-61	2.78E-43

Table 3.3 Watermarking results on LGSynth93 benchmarks.

Circuit	m	A	D	P	$\Delta A(\%)$	$\Delta D(\%)$	$\Delta P(\%)$	P_c	P_r
i7	16	4403	1.26	4.55	10.85	-8.70	10.17	7.52E-20	1.38E-06
	32	4594	1.33	4.71	15.66	-3.62	14.04	1.07E-27	2.33E-04
i2	16	2083	0.68	0.93	3.68	28.30	-2.11	2.64E-19	3.29E-06
	32	2135	0.70	0.68	6.27	32.08	-28.4	2.34E-26	1.33E-03
i9	16	4057	1.67	4.25	6.34	5.70	12.73	1.05E-13	1.40E-02
alu4	16	12316	2.02	9.24	2.51	-0.49	3.01	4.92E-25	2.22E-10
	32	12457	2.00	9.36	3.68	-1.48	4.35	1.70E-39	3.33E-12
	64	12659	1.98	9.44	5.36	-2.46	5.24	7.31E-55	1.34E-07
frg2	16	5879	1.75	4.62	1.34	2.34	1.09	1.91E-19	2.63E-06
	32	6043	1.65	4.70	4.17	-3.51	2.84	1.05E-26	8.52E-04
rot	16	5171	2.40	4.64	1.13	-1.64	1.98	1.41E-21	8.37E-08
	32	5232	2.40	4.70	2.33	-1.64	3.30	8.64E-32	7.42E-07
apex6	16	5594	1.24	4.38	3.36	-0.80	1.86	3.71E-21	1.66E-07
	32	5591	1.24	4.32	3.31	-0.80	0.47	8.16E-31	3.07E-06
x3	16	5869	0.96	4.71	1.87	0.00	0.21	5.86E-23	4.42E-09
	32	5896	0.96	4.74	2.34	0.00	0.85	6.28E-35	1.64E-09
	64	6031	0.96	4.90	4.69	0.00	4.26	5.54E-43	1.39E-02
k2	16	7504	2.16	3.13	0.98	0.47	0.64	1.41E-21	2.65E-09
	32	7560	2.16	3.13	1.74	0.47	0.64	8.64E-32	5.67E-10
	64	7605	2.16	3.11	2.34	0.47	0.00	1.75E-33	2.36E-03
i8	16	6564	1.89	6.13	1.88	-2.58	2.17	2.27E-21	4.38E-08
	32	6610	1.90	6.08	2.59	-2.06	1.33	2.60E-31	1.93E-07
dal	16	5409	2.62	3.03	2.21	3.56	1.00	4.01E-23	6.58E-09
	32	5454	2.65	3.02	3.06	4.74	0.67	2.69E-35	3.75E-09
	64	5598	2.63	3.05	5.78	3.95	1.67	5.26E-44	4.88E-02
pair	32	13799	1.43	10.73	1.13	0.00	1.71	3.94E-54	1.61E-22
	64	14012	1.51	10.94	2.69	5.59	3.70	1.08E-87	6.76E-29
	128	14294	1.58	11.03	4.76	10.49	4.55	4.21E-130	2.14E-25

des	32	25796	1.80	27.91	1.09	-2.70	0.58	3.15E-45	8.91E-21
	64	25744	2.18	27.97	0.89	17.84	1.05	2.69E-68	2.62E-25
	128	26248	1.80	28.38	2.86	-2.70	2.34	3.67E-81	5.28E-18
i10	32	18579	4.12	13.61	0.71	-0.72	0.89	4.47E-55	3.27E-23
	64	18692	4.13	13.69	1.32	-0.48	1.48	1.07E-89	2.54E-30
	128	18992	4.10	13.88	2.95	-1.20	2.89	1.14E-134	2.38E-28
apex5	16	6499	1.24	3.88	3.49	-0.80	5.72	8.87E-22	6.03E-08
	32	6589	1.24	3.88	4.92	-0.80	5.72	2.98E-32	3.75E-07
mm30a	32	15744	2.44	1.65	0.25	12.96	0.61	1.60E-48	1.96E-18
	64	15854	2.61	1.65	0.96	20.83	0.61	1.28E-75	1.84E-20
	128	16095	2.65	1.67	2.49	22.69	1.83	5.15E-101	2.00E-08
pdc	32	33620	3.20	6.90	0.71	-1.23	0.44	9.26E-58	2.09E-26
	64	33873	3.22	6.98	1.46	-0.62	1.60	2.48E-95	7.39E-37
	128	34241	3.18	7.02	2.57	-1.85	2.18	2.75E-147	6.50E-42
spla	32	29548	2.72	6.78	0.72	0.00	0.30	1.23E-51	4.23E-24
	64	29740	2.72	6.83	1.38	0.00	1.04	2.26E-82	3.85E-32
	128	30032	2.72	6.86	2.37	0.00	1.48	1.18E-117	3.96E-32
ex1010	64	87264	2.61	9.82	1.53	10.13	11.72	7.61E-130	1.09E-59
	128	87710	2.63	10.22	2.05	10.97	16.27	3.66E-220	1.15E-88
ex1010 ($\tau=2$)	64	87055	2.59	9.70	1.28	9.28	10.35	2.97E-124	1.52E-55
	128	87583	2.63	10.17	1.90	10.97	15.70	1.18E-208	2.99E-80
elliptic	64	117356	4.77	13.48	1.53	-6.29	0.00	6.32E-111	6.14E-49
	128	117692	5.70	13.51	1.82	11.98	0.22	7.28E-181	8.88E-67

Table 3.4 Watermarking results on ISCAS89, ISCAS99 benchmarks.

Circuit	m	A	D	P	$\Delta A(\%)$	$\Delta D(\%)$	$\Delta P(\%)$	P_c	P_r
S5378	32	20998	1.36	12.47	1.02	0.74	0.56	1.49E-33	9.83E-14
	64	21182	1.37	12.53	1.90	1.48	1.05	5.05E-39	1.03E-10
S9234	32	24789	2.21	6.64	1.11	1.84	0.00	2.05E-46	3.80E-17
	64	24972	2.29	6.66	1.86	5.53	0.30	5.83E-71	8.65E-18
	128	25375	2.25	6.70	3.50	3.69	0.90	1.10E-88	8.19E-04
S6669	32	44484	10.18	2.36	-0.90	0.00	-0.42	4.29E-61	1.84E-29
	64	44101	10.18	2.35	-1.76	0.00	-0.84	2.90E-102	4.46E-43
	128	43296	10.16	2.34	-3.55	-0.20	-1.27	2.09E-162	9.38E-55
S13207	32	56252	2.54	17.14	0.22	5.83	0.12	3.03E-56	2.83E-24
	64	56430	2.57	17.14	0.53	7.08	0.12	3.71E-92	1.69E-32
	128	56697	2.57	17.15	1.01	7.08	0.18	3.20E-140	7.16E-33
S38417	64	161250	3.86	31.77	0.30	-1.53	0.32	2.02E-125	3.87E-58
	128	161827	3.92	31.81	0.65	0.00	0.44	4.63E-211	1.60E-85
S35932	64	177241	1.99	58.31	0.50	2.58	-0.10	5.34E-114	2.93E-49
	128	178089	2.02	58.09	0.98	4.12	-0.48	2.51E-187	1.96E-67
S38584	64	171376	3.25	48.25	0.47	-1.81	-0.27	8.98E-126	8.29E-58
	128	171828	3.22	48.32	0.73	-2.72	-0.12	8.66E-212	7.53E-85
B22	64	233520	12.29	9.38	-0.10	-0.32	0.11	2.82E-129	1.22E-59
	128	233920	12.56	9.38	0.07	1.87	0.11	5.37E-219	1.44E-88
B22 ($\tau=2$)	64	233507	12.25	9.38	-0.11	-0.65	0.11	5.35E-126	3.08E-57
	128	233775	12.25	9.38	0.01	-0.65	0.11	2.98E-212	1.08E-83
B21	64	155776	12.25	4.98	0.40	-0.65	1.63	3.76E-118	2.66E-51
	128	156729	12.25	5.00	1.01	-0.65	2.04	5.29E-196	1.32E-71

Table 3.5 Comparison with logic synthesis watermarking method in [KIR06].

Circuit	m	Proposed		[KIR06]	
		ΔA (%)	P_c	ΔA (%)	P_c
i7	18	14.83	2.91E-21	2.88	4.22E-14
	36	15.91	1.65E-28	1.44	6.98E-28
i2	22	6.62	6.83E-23	10.74	1.87E-13
	43	8.26	1.48E-26	26.45	1.00E-23
i9	19	6.74	3.49E-14	8.57	2.33E-14
alu4	25	2.44	7.14E-34	6.82	5.84E-11
	49	4.18	1.37E-49	11.82	3.34E-20
frg2	21	0.52	1.43E-22	2.98	3.01E-10
	41	4.93	2.30E-27	8.61	4.15E-18
rot	24	2.27	1.58E-27	4.53	2.73E-12
	48	3.89	7.13E-36	8.36	1.04E-22
apex6	26	3.75	5.40E-28	5.37	6.41E-16
	51	6.58	5.75E-34	10.74	2.29E-29
C2670	29	2.49	2.63E-33	7.27	9.51E-16
	58	4.75	1.94E-43	12.42	7.22E-29
x3	28	1.68	1.67E-32	7.89	5.93E-16
	55	2.03	9.70E-43	14.66	5.12E-29
k2	33	2.02	3.24E-32	2.91	1.84E-10
	66	2.83	1.42E-32	6.05	3.00E-19
i8	34	3.46	4.24E-32	-4.45	1.06E-10
	67	6.77	3.77E-30	-10.83	4.53E-23
dalv	43	4.40	1.81E-40	4.19	1.07E-19
	86	8.24	1.36E-36	11.26	3.88E-36
C5315	55	4.53	2.24E-53	6.30	2.92E-32
	110	7.57	8.50E-53	14.57	4.40E-59
pair	58	1.95	3.39E-82	6.73	3.99E-31
	115	4.36	1.34E-123	14.62	6.88E-57
C6288	97	1.93	9.36E-69	10.72	7.01E-51
C7552	98	3.68	4.15E-100	8.25	1.48E-52
	196	6.32	2.27E-110	17.93	4.42E-96
des	112	2.12	5.23E-82	0.79	3.21E-55
i10	119	2.24	7.27E-130	4.26	5.98E-50
	238	3.88	1.34E-158	8.67	4.48E-94

Table 3.6 Comparison with logic synthesis watermarking method in [MEG00].

Circuit	m	Proposed	[MEG00]
		ΔA (%)	ΔA (%)
frg2	14	0.24	2.04
rot	20	1.88	1.53
apex6	20	3.75	11.24
C2670	19	2.08	2.55
x3	18	1.61	5.92
i8	15	2.28	0.86
pair	19	-0.27	0.38
apex5	4	1.48	0.38
S9234	19	0.38	2.61
S5378	20	1.73	2.60
mm30a	20	-0.04	1.36

In Table 3.5 and Table 3.6, the overheads of the proposed watermarking method are compared respectively with two other methods in [KIR06] and [MEG00]⁵. Due to the practical limitation of porting the program codes into different design tools, the comparison was performed by excerpting the results in [KIR06] and [MEG00] for the same benchmark circuits. In [KIR06], irregular signature lengths were used. Two different signature lengths based on the cases of 4% and 8% of gates being constrained as pseudo-primary outputs [KIR06] were used to compare the embedding overheads and P_c values with [KIR06]. In [MEG00], P_c was not evaluated. Thus only the area overhead was compared using the same watermark length equivalent to the number of additional constraints of [MEG00]. The least overhead and lowest P_c watermarked solutions are highlighted. For the 33 watermarked designs in Table 3.5, the average area overhead of the proposed method is 4.79% while that of the method in [KIR06] is 7.68%. For the 11 watermarked designs in Table 3.6, the average area overhead of the proposed method and that of the method in [MEG00] are 1.37% and 2.86%, respectively. It is found that, for most designs, the proposed method provides the strongest proof of authorship with lower overhead. Especially for large designs, such as designs “pair”, “C6288” and “C7552”, the results of the proposed method are remarkably better than those in [KIR06]. It can be seen that when a 97-bit watermark is embedded into design “C6288”, only 1.93% area overhead is incurred by the proposed method while almost 10 times more overhead is caused by [KIR06]. Also,

⁵ [KIR06], [MEG00] and [MOI05] are three main watermarking schemes proposed at logic synthesis level. As few experimental data was presented and the area overhead was not evaluated in [MOI05], the two former references are referred for the comparison with the proposed scheme.

the proposed method provides a stronger authorship proof. It is noted that the overheads due to the proposed watermarking method become negligible as the size of the design increases. This is not the case for the other two methods [KIR06, MEG00]. This proves that the proposed method is more applicable to the industrial size designs.

3.6 Summary

In this chapter, a constraint-based watermarking method at the logic synthesis level for IP protection was presented. In this approach, stealthy localities for watermark insertion are identified based on the notion of slack sustainability. The qualification of hosting cones, in conjunction with an incremental technology mapping, has several distinct advantages. First, the original design constraint can be satisfied with very low embedding overhead. Second, the signature length can be adaptively tailored to the embedding capacity of a design to optimize the ownership proof, watermark obscurity and resilience with low impact on circuit quality. Third, the existence of the watermark can be explicitly detected by a direct extraction method by comparing the master and the watermarked design. The correlation check also provides the cue on attacker's attempt of obliteration.

The method was applied to several benchmark suites of combinational and sequential circuits. The results show that the proposed watermarking approach generally performs better than other watermarking methods at the logic synthesis level in terms of the incurred overhead and probability of coincidence under the same signature length. The impact on watermarked circuit quality becomes indiscernible as the circuit size increases. The results also indicate that the probability of successful random obliteration descends rapidly with increased disjoint cones to signature length ratio. The method is found to be resilient to denial, removal and addition of watermark, which are three attack scenarios frequently analyzed in combinational circuit watermarking schemes. However, like most combinational circuit watermarking method, the watermark will not be guaranteed to exist upon radical re-synthesis of the entire circuit.

The technique proposed in this chapter requires some reverse engineering effort to prove the existence of the watermark. It does not offer the convenience to enable the authorship of an IP to be publicly detectable. In the next chapter, the scan chain at the design-for-testability stage will be studied to explore a new watermarking technique that will allow the authorship to be field authenticable.

Chapter 4

Synthesis-for-Testability Watermarking

4.1 Introduction

When a watermark is embedded at a higher level of design abstraction, ideally, it is desired that the verification of authorship can be performed at the downstream design without the need to reverse engineer the design to the level where the watermark is embedded. Dynamic watermarking schemes, such as watermarking on FSM [TOR00, OLI01, ABD05b] and watermarking during DfT [KIR98b], allow the ownership information to be verified from the output response of the watermarked design by injecting a specific input vector. The main advantage of FSM watermarking schemes is the traceability of the watermarked IP at the lower design levels. However, it is still impossible to directly authenticate the embedded IP core with these methods after the chip containing the IP core has been fabricated and packaged.

In this chapter, a new watermarking technique is proposed to enable the watermark sequence to be observed at the output pins by the IP providers and buyers that possess the correct input code. The watermark is embedded as implicit constraints to the scan chain ordering problem in the DfT process. Unlike other DfT watermarking methods [KIR98b, FAN03], the testability features are merged with the synthesis process itself so that the controllability and observability properties of the scan function for watermark detection become an integral part of the proposed watermarked IP.

The remaining sections are organized as follows. Section 4.2 provides the inspiration of this work. The SfT watermarking is presented in Section 4.3. Then, the proposed algorithm is explained in Section 4.4 with an illustrative example. In Section 4.5, a provable mechanism for public and field authentication of multiple IP authorships in a

packaged SoC is proposed. The watermarking scheme is analyzed in Section 4.6 and the experimental results are presented in Section 4.7. Finally, the chapter is summarized.

4.2 Percept of SfT-Based IP Watermarking

Scan design [ABR90] is typically used to make direct access to internal circuit nodes to overcome the difficulty of automatic test pattern generation for sequential circuits. To enable the ownership information to be output from a packaged chip upon the input of a specific validation code, the circuit controllability and observability are explored at the DfT stage. Normally, the test structure is added after the circuit under test (CUT) with PI/PO as the primary inputs/outputs has been synthesized [ABR90]. As shown in Figure 4.1, an additional *test enable* (TE) input pin is added to switch between the normal mode of operation and the test mode. All flip-flops are replaced by the special bistable scannable flip-flops (SFF) and sprung to form one or more scan chains. Two additional IO pins are required for the test mode. The *scan in* (SI) pin feeds the input test vectors serially and the *scan out* (SO) pin shifts out the corresponding response vectors. Since the test vectors can be independently generated and compacted to provide the desired fault coverage of the CUT, the scan chain registers can be ordered to minimize the test cost without affecting the fault coverage by reordering the bit streams of the scan vectors. Scan chain ordering is known to be an NP-hard problem [COX94], which makes the application of constraint-based watermarking viable in scan design.

Figure 4.1 Scan chain design used in DfT.

Watermarking directly on the basic scan architecture [KIR98b] suffers from two

problems. First, about 30% area overhead on the test structure is added in the form of extra circuitry for each SFF. The additional circuitry such as multiplexers in the scannable components can slow down signals in the critical path and degrade the system performance [COX94]. Second, from the watermark resilience perspective, since the test circuit is independent of the logic function of the IP being protected, it is easier for an attacker to substitute the watermarked test circuitry by his/her own scan design to erase the authorship information.

To overcome the weaknesses of watermarking on scan chain with SFFs [KIR98b], the registers in the scan chain are ordered for watermarking based on the cost of incorporating the test logic into the functional data path to achieve scan design with the use of standard flip-flops and a TE pin. A simple SfT example is shown in Figure 4.2, where a standard flip-flop and a multiplexer (MUX) can be used to realize an SFF in the scan chain. If F_j denote the input function of flip-flop j in the normal mode, the input function of the scannable version of flip-flop j can be expressed as:

$$D_j = F_j \cdot \overline{TE} + SI_j \cdot TE \quad (4.1)$$

Figure 4.2 Merging of normal and scan functions.

In this example, the functional logic, $F_j = PI_2 \cdot Q_i + PI_1$, where PI_1 and PI_2 are the primary inputs and Q_i is the output of flip-flop i . When the scannable design is synthesized by a logic optimizer using a standard cell library, the extra multiplexer

logic required for the testability is absorbed into the input logic of F_j and mapped to a library cell, AOI222 and two inverters. To reuse parts of the functional logic for test functions as illustrated, the information on the data paths connecting normal non-scannable flip-flops can be identified from the design description. The extracted information about the flip-flop connectivity is then exploited to construct watermarked scan paths through existing functional logic of the design. The watermarked scan chain can be established at the RTL or the logic synthesis level and carry forward to the end of the design flow. The concept of SFT-based watermarking makes testability features a part of the synthesis process in the IP creation. Unlike conventional SFF-based DfT, the test logic can be globally optimized with the functional logic by the synthesis tool to meet the global area and timing constraints. Therefore, lower area and better timing performance [COX94, HUA01, KO04, NOR96, LINC98] and delay fault coverage [HUA01, KO04] can be achieved. The combined test and functional logic circuit as a whole is easier to route than those with SFF-based DfT [NOR96]. These criteria are important to increase the reverse engineering and re-optimization effort required to maintain the desired optimality and functionality of the watermarked design by rogue obfuscation of the scan paths.

SFT can potentially yield a lower cost scan design than that by the post-synthesis scan insertion [KIR98b], but the extent to which the existing logic can be reused for the test logic depends largely on the scan order. The scan ordering problem is formulated by representing the sequential circuit as a succinct dependency graph where each vertex represents a flip-flop and a directed edge represents a combinational path from a flip-flop's output to a flip-flop's input. In the test mode, the dependency graph must also satisfy a fundamental test synthesis constraint which specifies that the input of each flip-flop must be dependent on the data output of the preceding flip-flop in the scan chain. This is not a problem in principle as each flip-flop input equation can be expressed as a function of every other flip-flop output and the primary inputs in a synchronous sequential circuit by introducing the desired test logic. Depending on the original relationship between two flip-flops, each vertex in the dependency graph has a directional scan cost to each other's vertex by making the pair of flip-flops scan dependent.

4.3 Scan Chain Ordering Problem for SFT Watermarking

In this section, the SFT watermarking scheme is modeled as a scan chain ordering problem for minimum logic overheads. For brevity, a full-scan chain of a single IP core is used to explain the concept but the idea can be extended to partial scan designs and designs with multiple IP cores and multiple scan chains.

Assume that a full-scan design involves N internal flip-flops, $R = \{r_i\}_{i=1}^N$ of a CUT. Then, a scan chain ordering, π is defined as a one-to-one mapping of a set of N scan flip-flops, R to a set of positions, $S = \{s_j\}_{j=1}^N$ where $s_j = \pi(r_i)$ such that the i -th bit of the test vector is loaded into the j -th scan flip-flop once a complete test vector has been shifted into the scan chain. The aim of scan chain ordering, when applying to SFT, is to find an optimal permutation, $\pi: R \rightarrow S$ to minimize the combinational logic required to use standard flip-flops for testability. In the process, the watermark information can be imposed by further restricting the scan chain to output a vector that embraces a unique bitstream in response to a specific test vector.

The dependency graph of a general sequential circuit can be extracted from the gate-level or RTL description and converted to an SFT cost graph by evaluating the cost of including and merging the test logic with the functional logic between two adjacent flip-flops in a scan chain. A flip-flop is said to be scan dependent if its original system logic input is functionally dependent on the flip-flop immediately preceding it in the scan chain. Scan dependence can be used to increase the sharing of system and test logic without reducing the fault coverage capability of the scan design. In order to establish a good heuristic for a low overhead watermarked scan design, the edge weights of the dependency graph are established by considering different scenarios of the existing logic functions between two flip-flops in (4.1).

Let F_i and D_i be the input functions of a standard flip-flop, FF_i of the sequential circuit before and after the test logic insertion, respectively, and Q_i be the output function of FF_i . If FF_j is selected to be the successor of FF_i , then $SI_j = Q_i$ in (4.1) and D_j can be expressed as

$$D_j = F_j \cdot \overline{TE} + Q_i \cdot TE \quad (4.2)$$

There are two cases to consider. If F_j is independent of Q_i , a multiplexer will be inserted between the two flip-flops, FF_i and FF_j . In terms of basic gates, the cost is equivalent to two AND gates and one OR gate. On the other hand, if some relationship exists between FF_i and FF_j and $F_j = f(Q_i)$, then the function F_j can be decomposed into its cofactors that are independent of Q_i by Shannon's expansion as follows:

$$F_j = Q_i \cdot F_{Q_i} + \overline{Q_i} \cdot F_{\overline{Q_i}} \quad (4.3)$$

where $F_{\overline{x}}$ and F_x are the cofactors of F with respect to $\overline{x}$ and x , respectively. Then, (4.2) can be further reduced to:

$$D_j = Q_i \cdot (TE + F_{Q_i}) + \overline{Q_i} \cdot F_{\overline{Q_i}} \cdot \overline{TE} \quad (4.4)$$

Compared to (4.2), only one AND gate and one OR gate need to be added to satisfy the test synthesis constraint in the dependency graph between FF_i and FF_j . In particular, when the Shannon cofactors in (4.3) are constants or in some specific forms that allow further Boolean simplification with the test logic, the test synthesis constraint of (4.4) can be fulfilled at a lower cost. These special cases have been studied in [COX94, NOR96] and the normal and scan functions between two flip-flops, FF_i and FF_j are summarized in Figure 4.3 along with the overheads needed to fulfill the SFT requirement.

cost ↑	Normal function	Scan function	overhead ↑
3	$F_j = g$	$D_j = g \cdot \overline{TE} + Q_i \cdot TE$	2·AND+OR
2	$F_j = Q_i \cdot F_{Q_i} + \overline{Q_i} \cdot F_{\overline{Q_i}}$	$D_j = Q_i \cdot (TE + F_{Q_i}) + \overline{Q_i} \cdot F_{\overline{Q_i}} \cdot \overline{TE}$	AND+OR
1	$F_j = Q_i^{[0,1]} + f_{\overline{Q_i}}$ $F_j = Q_i^{[0,1]} \cdot f_{Q_i}$ $F_j = Q_i^{[0,1]} \oplus f_{\overline{Q_i}}$ $F_j = s \cdot Q_i^{[0,1]} + s \cdot t$	$D_j = Q_i^{[0,1]} + \overline{TE} \cdot f_{\overline{Q_i}}$ $D_j = Q_i^{[0,1]} \cdot (TE + f_{Q_i})$ $D_j = Q_i^{[0,1]} \oplus \overline{TE} \cdot f_{\overline{Q_i}}$ $D_j = (s + TE) \cdot Q_i^{[0,1]} + s + \overline{TE} \cdot t$	AND OR AND OR
0	$F_j = Q_i^{[0,1]}$	$D_j = Q_i^{[0,1]}$	NONE
	Normal function	Scan function	

Figure 4.3 Scan function overheads and edge costs of dependency graph for SFT⁶.

⁶ s , t and g are arbitrary functions that have no relation to Q_i . $Q_i^{[0,1]}$ denotes that the inversion of Q_i is allowed.

From Figure 4.3, the overhead induced by chaining any pair of flip-flops in the scan path can be estimated from a design. There is no need to explicitly differentiate between the costs of primitive AND and OR gates as the difference has no appreciable influence to the final optimization result. Thus, it is sufficient to annotate on a fully connected dependency graph (where every vertex has a directed edge to every other vertex) a relative cost metric in terms of the number of primitive gates (0, 1, 2 or 3) according to the scan function of Figure 4.3. Figure 4.4 shows the SfT cost graph of a sequential circuit. The SI and SO pins can be saved by using the existing IO pins of the package. The scan function costs from a primary input to a flip-flop and from a flip-flop to a primary output can be similarly determined by substituting SI_j in (4.1) by the primary input and F_j by the primary output in consideration, respectively.

Figure 4.4 SfT cost graph of a sequential circuit.

If F_i directly precedes F_j in the scan chain and $cost(r_i, r_j)$ is the logic complexity of making flip-flops, F_i and F_j scannable, the test and functional logic can be minimized by finding an optimum scan chain ordering $\pi: R \rightarrow S$ that has the least value of

$$\sum_{i=1}^{N-1} cost(\pi(r_i), \pi(r_{i+1})) \quad (4.5)$$

With the establishment of the SfT cost graph, this problem becomes that of finding a minimum cost Hamiltonian path⁷ from a PI to a PO. The complexity is equivalent to that of the traveling salesman problem (TSP) [AAR97]. Let Y be a response vector obtained when a specific test vector, X , is loaded into the scan chain. The goal of watermarking is to find a minimum cost scan chain ordering, $\pi_{wm}: R \rightarrow S$ that produces an output vector, $Y' = \pi_{wm}(Y)$, where an m -bit ($m < N$) signature, $W = w_1 w_2 \dots w_m$ and $w_i \in \{0, 1\}$ can be extracted from the outputs of m randomly selected flip-flop

⁷ A Hamiltonian path is a path in an undirected graph which visits each vertex exactly once.

positions, $p_1, p_2, \dots, p_m$ when the test vector, $X' = \pi_{wm}(X)$ is applied to the scan chain. To achieve this goal, the scan flip-flops allocated to the positions, $\{p_j\}_{j=1}^m \subset S$ will be further constrained such that the scan output, $S_{out}(r_{p_i}) = w_i$, where $w_i \in W$. Without decapsulating the chip, the authorship of the embedded IP core can be proved by recovering the signature, W from Y' . Suppose there are n possible solutions to order the scan flip-flops, the number of possible solutions that can meet the additional constraints will be reduced to n_{wm} . Thus, the probability that a non-watermarked scan chain carries the same watermark as the watermarked solution by coincidence is given by n_{wm}/n . The authorship proof is strengthened if this probability is made smaller.

4.4 Proposed Algorithm and Example

Based on the formulation of scan chain ordering problem, the proposed SFT watermarking scheme is elaborated as follows:

4.4.1 Preliminary Vectors Generation

This is the preliminary step for watermark insertion. It is assumed that there is an initial ordering of flip-flops, $R = \{r_i\}_{i=1}^N$ in the scan chain. A unique input vector, $X = \{x_i\}_{i=1}^N$, $x_i \in \{0,1\}$, designated by the IP designer, is shifted into the initial scan chain to obtain an output vector, $Y = \{y_i\}_{i=1}^N$, $y_i \in \{0,1\}$. The flip-flops in the scan chain that output the “0” and “1” bits of the binary vector Y are divided into two sets, R_0 and R_1 , accordingly. To make the vector X for authorship proof a part of the scan vectors, it can be arbitrarily selected from the test vectors generated by the DfT tool for the IP core. The criterion is that the output vector Y it generated must contain sufficient number of “1” and “0” bits for the watermark, W .

The watermark, $W = \{w_i\}_{i=1}^m$ is an m -bit ($m < N$) digital signature generated by signing an ownership message MSG with a secret key, K . The same secret key is used to generate m distinct integer numbers in the interval of $[1, N]$ by a pseudorandom number generator (PNG) [MEN96]. These numbers, $P = \{p_i\}_{i=1}^m$ where $1 \leq p_i \leq N$ for

$i = 1, 2, \dots, m$ and $p_i \neq p_j \forall i \neq j$, are indices to the watermarked flip-flop positions in the final scan chain S . The flip-flops assigned to these positions are constrained to output only bit values identical to the corresponding watermark bits in W under the input vector $X' = \pi_{wm}(X)$. Secure hash algorithm, SHA-1⁸ [MEN96] is used as the keyed one-way function for the PNG. As it is computationally infeasible to find a collision of this hash function, the possibility that the same group of numbers is generated by coincidence is extremely low without the knowledge of the secret key K .

4.4.2 Watermark Insertion

A fully connected SFT cost graph, $G(V, E)$, is constructed from the CUT where each vertex represents a flip-flop, $r \in R$ in the scan chain. The weight on each directed edge is the overhead cost of the scan function in Figure 4.3 estimated from the normal function of the pair of flip-flops. The greedy nearest neighbor (NN) was among the first algorithms used to determine a solution to the TSP. It quickly yields a short tour and the NN heuristics have been mostly used in scan chain ordering problems [COX94, NOR96]. In the proposed scheme, NN heuristics for the TSP has been adapted to watermark the scan chain. The watermarked scan chain, S is treated as a row of N vacant positions numbered from 1 to N . Each position is to be filled by a vertex in the SFT cost graph.

The watermark insertion process is integrated into N local searches for the nearest neighbors to locally optimize each individual flip-flop placement. The watermark insertion procedure is shown in Figure 4.5, where PI and PO are respectively, the sets of primary inputs and primary outputs of the CUT. Initially, as indicated in line 7, a PI that has the minimum edge cost to any vertex is selected as the SI and the flip-flop to which this edge is directed is assigned to the first position, s_1 in the watermarked scan chain, provided that s_1 is not a watermark host. Otherwise, if $\{1\} \in P$, only flip-flops from either R_0 or R_1 can be allocated, depending on the value of the watermark bit to be output at this position.

⁸ SHA-1 is used as the secure hash algorithm throughout this thesis. Compared to 128-bit hash functions, the 160-bit hash-value of SHA-1 provides increased security against brute-force attacks. SHA-1 is considered stronger than MD5 [MEN96].

With reference to the SFT cost graph, let $E(a, B)$ denote the set of all weighted edges directed from a vertex a to a vertex in vertex set B , where $a \in V$ and $B \subseteq V$. Furthermore, let U be a set of vertices that are yet to be allocated to S and $r_i = \pi_{wm}^{-1}(s_j)$ denote the allocation of a position, $s_j \in S$ to a vertex, $r_i \in R$. In general, after the position, s_{i-1} has been allocated, the next vacant position s_i can be formally allocated based on the following NN criteria to minimize the overhead of the watermarked scan chain, as indicated in line 11.

$$(s_{i-1}, s_i) = \begin{cases} \min\{E(\pi_{wm}^{-1}(s_{i-1}), U)\}, & i \notin P \\ \min\{E(\pi_{wm}^{-1}(s_{i-1}), U \cap R_0)\}, & i = p_j \text{ and } w_j = 0 \\ \min\{E(\pi_{wm}^{-1}(s_{i-1}), U \cap R_1)\}, & i = p_j \text{ and } w_j = 1 \end{cases} \quad (4.6)$$

where $\min(E)$ is a local search function for finding a minimum weighted edge from an edge set, E .

This local search is repeated until every flip-flop in R has been allocated to a unique position in S . The primary output with the least cost edge from the last allocated flip-flop will be selected as SO, which is indicated in line 19. It can be seen that the codes in lines 11, 14, 15 have a computational complexity of $O(N)$ and the external loop, as indicated in line 9, also has the same computational complexity. Hence, the watermark insertion algorithm has a computational complexity of $O(N^2)$. The design description is modified to reflect the scan path watermarked by this procedure before it is technology mapped and optimized.

```

1: watermark_insertion ( $G, W, X, R, S, m, K, N$ ) {
2:   Obtain  $Y = \{y_i\}_{i=1}^N$  by applying  $X = \{x_i\}_{i=1}^N$  onto  $R$ ;
3:    $R_0 = \{r_i \in R \mid y_i = 0 \quad \forall 1 \leq i \leq N\}$ ;
4:    $R_1 = \{r_i \in R \mid y_i = 1 \quad \forall 1 \leq i \leq N\}$ ;
5:    $P = \{p_i\}_{i=1}^m = \text{PNG}(K, m, N)$ ;
6:    $W = \{w_i\}_{i=1}^m = \text{digital\_signature}(\text{MSG}, K)$ ;
7:    $(\text{SI}, s_1) = \min(E(PI, R))$ ;
8:    $U = R - \pi_{wm}^{-1}(s_1)$ ;
9:   for ( $i = 2$  to  $N$ ) {
10:    if ( $i \notin P$ ) {
11:       $(s_{i-1}, s_i) = \min(E(\pi_{wm}^{-1}(s_{i-1}), U))$ ;
12:    } else {
13:      if ( $w_j = 0$ )
14:         $(s_{i-1}, s_i) = \min(E(\pi_{wm}^{-1}(s_{i-1}), U \cap R_0))$ ;
15:      else  $(s_{i-1}, s_i) = \min(E(\pi_{wm}^{-1}(s_{i-1}), U \cap R_1))$ ;
16:    }
17:     $U = U - \pi_{wm}^{-1}(s_i)$ ;
18:  }
19:   $(s_N, \text{SO}) = \min(E(\pi_{wm}(p_N), PO))$ ;
20:  return  $\pi_{wm}$ ;
21: }

```

Figure 4.5 Nearest neighborhood algorithm for watermark insertion.

4.4.3 Watermark Detection

The NN algorithm in Figure 4.5 returns a permutation, $\pi_{wm}: R \rightarrow S$. This permutation is used to obtain an input validation vector, $X' = \pi_{wm}(X)$ for the watermark detection. By injecting the vector X' into the watermarked scan chain S through the SI in the test mode, an output vector, $Y' = \pi_{wm}(Y)$ will be shifted out of the SO. The watermark, $W = \{w_i\}_{i=1}^m$ can be retrieved from the vector, Y' , where $w_i = y'_{p_i}$. If the recovered watermark matches W perfectly, the authorship is proved. Otherwise, if it is highly correlated with W , the watermarked design has been modified but most watermark bits are still intact.

A simplified example is shown in Figure 4.6 to illustrate the watermark insertion and detection. In Figure 4.6, $r_1, r_2, \dots, r_7$ denote seven scan flip-flops in an initial scan

chain generated automatically by the standard test design tool. Assume that the IP core is to be dynamically watermarked with a digital signature, $W = "101"$. Suppose the response vector, $Y = "0110011"$ that contains W is obtained from the test vector, $X = "00101110"$ when it is loaded into the scan chain R . Then $R_0 = \{r_1, r_4, r_5\}$ and $R_1 = \{r_2, r_3, r_6, r_7\}$. If $P = \{4, 2, 7\}$ is generated by PNG, the watermark bits, $w_1 = 1$, $w_2 = 0$ and $w_3 = 1$ are to be extracted from the positions, s_4 , s_2 and s_7 , respectively, of the watermarked scan chain.

Figure 4.6 A simple example of watermark insertion.

For brevity, assume that the CUT has an SFT cost graph represented by the connection matrix of Figure 4.7(a) and that flip-flop r_2 has the least edge weight from among all primary input and flip-flop pairs. Thus, r_2 is allocated to s_1 . Although $U = \{r_1, r_3, r_4, r_5, r_6, r_7\}$, only $U \cap R_0 = \{r_1, r_4, r_5\}$ can be allocated to s_2 because s_2 must produce $w_2 = 0$. The nearest neighbor to r_2 is given by r_1 since $e(r_2, r_1) = 1$ is the minimum weight among $e(r_2, r_1)$, $e(r_2, r_4)$ and $e(r_2, r_5)$ as shown in Figure 4.7(b). So r_1 is allocated to s_2 and $U = \{r_3, r_4, r_5, r_6, r_7\}$. From Figure 4.7(c), as $e(r_1, r_5) = 0$ has the smallest weight, r_5 is allocated to s_3 . $U = \{r_3, r_4, r_6, r_7\}$ but only $U \cap R_1 = \{r_3, r_6, r_7\}$ is eligible to compete for s_4 , which hosts $w_1 = 1$. From Figure 4.7(d), $e(r_5, r_7) = 1$ has the smallest weight and r_7 is allocated to s_4 . This process continues and r_4 , r_6 and r_3 are sequentially allocated to the last three positions.

	r1	r2	r3	r4	r5	r6	r7
r1		2	1	3	0	1	2
r2	1		3	2	3	2	3
r3	2	1		3	1	1	3
r4	3	3	1		3	2	2
r5	3	2	3	2		2	1
r6	2	2	2	2	3		3
r7	1	0	1	3	1	1	

(a) Connection matrix of cost graph.

(b) Flip-flop for s_2 . (c) Flip-flop for s_3 . (d) Flip-flop for s_4 .

Figure 4.7 Watermarking process on the example design.

The scan flip-flops, R are replaced by the standard flip-flops S and their respective scan functions, and the combinational paths are re-synthesized to obtain the watermarked design. The mapping $S = \pi_{wm}(R)$ is shown at the bottom of Figure 4.6. When the permuted vector $X' = \pi_{wm}(X) = "0010011"$ is scanned into the watermarked scan chain, the signature "101" can be retrieved from the 4th, 2nd and 7th positions of the response vector, $Y' = \pi_{wm}(Y) = "1001011"$ when it is shifted out of the SO.

4.5 Public and Field Authentication of Multiple IP Authorships

While the above dynamic watermark detection provides a convenient and non-invasive authorship proof in court without uncovering the design information and rendering the IP useless, it does not allow the watermark detection to be executed publicly without compromising its strength or security. This is because the vector P needs to be made known to the verifier and the locality of the watermark can be revealed.

SoC design is characterized by heterogeneous IP adoption. More than one party are usually involved in the creation of the IPs in an SoC. What makes public and on-site authentication of multiple authorships a desired feature is the assurance given to the consumers and IP buyers that the content of the chip they bought is genuine and legitimate. On the other hand, for IP watermarking to thrive, IP owners also want the assurance that the IP sold will not be illegally redistributed by their customers and licensees. The dilemma is that the information made publicly accessible for the ease of watermark detectability by the lawful IP users is also exploitable by the adversaries for forgery. In this section, it is shown that the conflict between watermark robustness and detectability can be elegantly resolved by incorporating an identification protocol into the proposed watermarking scheme.

The proposed public watermarking scheme is inspired by the Shamir's identity-based signature scheme [SHA84]. Shamir's method was originally designed to allow any two communicators in a public channel to verify each other's signature without exchanging private and public keys or keeping key directories. The system uses an RSA scheme [MEN96] with the parameters d , e , p and q . $de \equiv 1 \pmod{\phi}$ where $\phi = (p-1)(q-1)$. Cryptographic security requires that the two large primes, p and q be kept secret and d is made known only to the key generation center. The parameters, e and $n = pq$ are made public.

An identity, ID_i that meaningfully associates an IP owner or the IP created by him, is selected for certification by the key generation center. The center generates a secret key K_i to the IP owner.

$$K_i = (ID_i)^d \pmod{n} \quad (4.7)$$

To make the watermarked IP a property of ID_i , the IP owner uses the assigned secret key K_i to generate the vectors W and P for the watermark insertion algorithm in Figure 4.5. To enable public authentication of the watermarked IP without divulging W or P , he/she signs the output vector Y' obtained from the watermarked scan chain by:

$$\begin{aligned} \alpha &= (X_{ij})^e \pmod{n} \\ \beta &= K_i \cdot (X_{ij})^{PNG(\alpha, Y'_{ij})} \pmod{n} \end{aligned} \quad (4.8)$$

where PNG is a keyed one-way hash function. The indices, i and j are used to identify different IP owners and buyers (or licensees), respectively. $X'_{ij} = \pi_{wm}(X_{ij})$ is the input vector designated to the j -th customer to produce Y'_{ij} from the watermarked IP owned by ID_i . As it is computationally intractable to compute discrete logarithm, the permutation π_{wm} will not be divulged by revealing α and X'_{ij} publicly. The signatures α and β , together with X'_{ij} can be provided to any one who wishes to authenticate the watermarked IP in a packaged chip. The association of an encapsulated IP with ID_i can be easily verified by:

$$\beta^e = ID_i \cdot (\alpha)^{PNG(\alpha, Y'_{ij})} \bmod n \quad (4.9)$$

The cohesive property [BUR89] is assumed in the integrity functions of the cryptographic protocol, which states that if part of an encrypted message comes from a known source, then the whole of it does. Thus, the correctness of the signatures of the output Y'_{ij} of the watermarked design (in response to X'_{ij}) can be checked by the verifier to testify that the claimant is in possession of a secret K_i associated by design with the genuine party ID_i , thereby preventing impersonation. The non-repudiation and notarization aspects of this identification is a unique advantage of the proposed watermarking scheme to deter and track illegal distribution of encapsulated IP cores, as exemplified by the following scenarios.

Scenario 1: Bob copies an IP core authored by Alice without licensing it. Any one can easily vouch for the actual IP ownership on the spot by applying the above test to the chip sold by Bob that encapsulates Alice's IP core. In other words, if Alice represents a large corporation, Bob is unable to exploit the marketing advantage associated with the brand name of Alice as Bob's customers are unable to verify the authenticity of the IP without the ID of Alice.

Scenario 2: Bob licenses an IP core from Alice and attempts to resell or distribute it in his name. To convince his licensees that he is the legal owner of this IP core, Bob needs to re-sign the IP with his own ID . Based on the cohesive property of the above identification scheme, without the private key K_i of Alice, it is impossible to find a

different signature (α, β) to fulfill the test (4.9) with Bob's *ID*.

Scenario 3: Since the index j to X_{ij} in (4.8) can be made different for the same IP licensed to different customers or different groups of customers. In the case of Scenario 1, Alice can trace which of her licensees is ultimately liable for the illegal distribution of her IP by finding out the correct index j to X_{ij} and its associated parameters that fulfill the test (4.9).

It is known that none of the existing IP watermarking methods provides such a provable mechanism to enable multiple IP cores in an SoC to be publicly authenticated in the field by the consumers, and at the same time, effectively track the fraudulence activities of the IP licensees.

4.6 Watermark Resilience Analysis

The probability of coincidence P_c is a key measure of the authenticity of a watermark. It is defined as the probability that an ownership watermark is found in a non-watermarked design by the watermark detector. For the proposed scheme, this probability can be analytically approximated by assuming that each flip-flop position is equally probable to be selected to host a watermark bit. Thus, the probability of selecting m uniquely labeled watermarked positions from N candidate positions is given by $1/C_m^N$. Since the probability that an arbitrarily selected m -bit binary sequence matches a specific m -bit pattern is $1/2^m$, P_c can be expressed as:

$$P_c = \frac{1}{C_m^N} \left(\frac{1}{2} \right)^m \quad (4.10)$$

A lower P_c implies a stronger authorship proof. Given a scan chain length N , the longer the signature length m , the stronger the authenticity of the watermark.

The false positive rate can also be determined statistically by applying a set of randomly generated test vectors (excluding the vector X') from an arbitrary distribution to the watermarked design in the test mode. Let N_T be the total number of test vectors applied and $N_C(\tau)$ be the total number of output responses that were

detected with at least τ fraction of matched watermark bits. Then, the false positive rate can be determined as follows:

$$P_\lambda(\tau) = \frac{N_c(\tau)}{N_T} \quad (4.11)$$

where $0 \leq \tau \leq 1$. To constitute a false positive, $\tau = 1$ since all bits extracted by the detector need to be matched exactly with the watermark bits. As τ reduces, P_λ increases and a threshold of discrimination can be determined empirically that with certain degree of confidence, the authenticity of the design can be assured by detecting only a fraction of the watermark bits.

In general, no steganographic scheme is immune to different types of attacks. Therefore, it is important to analyze how well a proposed scheme stands up to different types of attacks. In the following analysis, Alice is the IP owner and Bob the attacker. Subscripts “ A ” and “ B ” are used to distinguish the variables corresponding to Alice and Bob, respectively.

4.6.1 Ghost Searching

Without touching the watermarked design from Alice, Bob can generate his watermark W_B by signing his own message with his private key K_B . He can also generate Y'_B with an arbitrary X'_B and then select some bits from Y'_B to make up W_B to forge the ownership. Bob’s ownership claim can be easily repudiated by Alice in court by demonstrating that the presence of her watermark in the IP is a consequence of some deliberate actions. Alice can show that the localities of the m watermark bits of W_A are related to specific flip-flops in the design that are uniquely selected by a random sequence generated by her secret key K_A . Unless Bob can inverse the cryptographic one-way function, like SHA-1, he cannot prove that the m watermark bits extracted by him are in any way related to the design and the detection method by a provable cryptographic means.

4.6.2 Denial Attacks

Bob may deny Alice's ownership by arguing that the validation test and response vectors, X'_A and Y'_A are obtained by trial-and-error to match the watermark W_A generated by Alice's key K_A . Bob cannot sustain his denial if Alice can convince the court that her design has been watermarked. Alice only has to show that the watermark and the watermarked positions in the scan chain are uniquely generated by her with a cryptographic one-way function using her own secret key, and the false positive rate for $\tau = 1$ is very low (ideally zero) for a sufficiently large number of random tests.

4.6.3 Removal Attacks

The price to pay for on-chip authentication in all scan chain watermarking is that the scan chain logic can be traced by the test enabled signal. Therefore, SFT watermarking does not prevent the scan chain logic from being detected by hiding it in the functional part, but it does increase the reverse engineering effort required to successfully rip out the authorship proof without degrading the value (quality) of the fraudulent IP. One advantage of the proposed scheme over conventional SFF-based watermarking is that the design can be watermarked at a high level of design abstraction so that the downstream watermarked design is better protected. The posterior optimization processes make the complexity of analyzing the circuit at the lower abstraction level prohibitively high for the adversary to forge the watermarked IP without degrading the circuit cost or performance.

4.6.3.1 Combinational Logic Re-synthesis

In order to erase the watermark, Bob may try to use various logic optimization tools to re-synthesize the combinational logic of the circuit. From the combinational logic optimization perspective, the re-synthesis operation will preserve the inputs/outputs behaviors of the flip-flops in the design and the logic relationship between any two flip-flops in the design will not be affected. Therefore, the SFT watermarking is robust against attacks by combinational logic re-synthesis.

4.6.3.2 Scan Chain Exchange

In multiple scan chain design, each scan chain is configured to test a kernel or disjoint portion of logic that can be tested independently so that the test for each kernel can be carried out in parallel to reduce the test time [NAR95]. If Alice's watermarked IP contains multiple scan chains, Bob may cut the watermarked chain and connect it with another chain to erase the watermark. For example, in Figure 4.8, scan chains 1 and 2 are composed of sequentially stitched scan flip-flops $\{i, j, k\}$ and $\{x, y, z\}$, respectively. Suppose one of the scan chains, scan chain 1 is watermarked. Bob may exchange the inputs of the flip-flop j and flip-flop y so that the two scan chains are configured as $\{i, y, z\}$ and $\{x, j, k\}$ by adding a multiplexor controlled by the test enable signal and redirecting the wirings to preserve the functionality. Thus the watermark on scan chain 1 cannot be verified under the specific input sequence. By sharing functional paths between flip-flops as scan path, SfT watermarking causes the architectural synthesizer to allocate adjacent flip-flops in the same scan cluster near each other to minimize the interconnections. Thus, the area saved after physical synthesis is even more significant than the area saved by logic synthesis [NOR96]. There is little or no room for placement and routing manipulation on SfT-based watermarked design than on conventional SFF-based watermarked design. To further constrain adverse routing manipulation, the physical information can also be considered concurrently when the watermark is embedded into the scan chain ordering [HIR98]. Besides, the fault coverage of the two modified kernels can no longer be guaranteed and non-existing faults are likely to be detected in the modified kernels by the given test vectors. With no knowledge of Alice's IP design, significant effort is required to rectify the false negatives or regenerate the new test patterns for the specified fault coverage. The performance penalty will limit the value and usefulness of the fraudulent design.

Figure 4.8 A scenario for scan chain exchange and flip-flop exchange.

4.6.3.3 Flip-Flop Exchange

In an attempt to prevent Alice from detecting the correct W_A , Bob may try to exchange some flip-flops in the scan chain with the hope to corrupt the watermark bit sequence. An efficient routing tool can be used to perform this operation with reasonable effort. Merely exchanging two flip-flops in the scan chain has no effect on the scan output, unless the scan function can also be changed without jeopardizing the normal operations of the design. Rerouting the output (or input) of one flip-flop with another to change their scan-dependency cannot guarantee correct functionality of the design.

In SFT, each flip-flop in the scan chain does not constitute a scan element by itself and there is no clear demarcation between the functional and test logics associated with a flip-flop after logic optimization. Except for the case of cost 0 where the relation between two flip-flops is trivial, scan dependency cannot be altered by simply swapping or rerouting flip-flop connections without affecting the behaviors of the design. To succeed, Bob needs to identify the relationship between the flip-flop pairs prior to the possible exchange. If the HDL code is available, this task is comparatively easier. HDL code will not be provided by nature of the scope of protection for this category of watermarking scheme. Thus, Bob needs to analyze the watermarked circuit to extract enough information to change the scan-dependency of any pair of flip-flops without altering the normal operation or degrading the quality of the design. For example, if Bob wants to change the scan order of flip-flops j and y in Figure 4.8, he must identify the relations of flip-flop j to its predecessor, flip-flop i , and its successor, flip-flop k , as well as those of flip-flop y to flip-flops x and z . Bob needs to

winnow out the test logics from f_1, f_2, f_3 and f_4 , which are the input functions of flip-flops j, k, y and z , respectively, before he can establish a new meaningful scan relation between flip-flops j and y . To completely erase and alter the watermarked scan chain, Bob must remove the subcircuits related to the scan functions, regenerate his own scan functions to replace the original watermarked scan chain and repeat the entire synthesis and optimization tasks of the design. As the test logic has been well blended with the functional logic of the watermarked design after synthesis and optimization, segregating them requires substantive reverse engineering effort. Therefore, the difficulty of the above tasks is comparable to the repetition of the original logic design, synthesis and optimization processes.

Assume that with reasonable reverse engineering effort, Bob is able to alter a fraction of the combinational circuits such that the positions of α ($\alpha \ll N$) flip-flops in the scan chain have changed without affecting the design functionality or quality. In what follows, the probability that Bob will successfully remove at least i watermarked bits in this attempt will be derived.

Let d_k denote the number of derangements of k ($k > 1$) flip-flops, where a derangement [DOS02] is defined as a complete rearrangement of objects such that no object occupies its original position. d_k can be obtained by recursion if an arbitrary flip-flop r_i is exchanged with another flip-flop, r_j initially, where $i \neq j$. If both r_i and r_j are retained after the exchange, there are d_{k-2} derangements of $(k-2)$ flip-flops. If r_i is allowed to further exchange with other flip-flops after the initial exchange, there are d_{k-1} derangements of $(k-1)$ flip-flops. Therefore,

$$d_k = (k-1)(d_{k-1} + d_{k-2}) \quad (4.12)$$

where $d_1 = 0$ and $d_2 = 1$.

Let N_0, N_1, m_0 and m_1 denote the total number of “0” and “1” bits in Y'_A , and the total number of “0” and “1” watermark bits, respectively. $N = N_0 + N_1$ and $m = m_0 + m_1$. Let S be the set of α flip-flops selected by Bob for derangement and S_0 and S_1 be its subsets of flip-flops that hold the “0” and “1” bits of Y'_A , respectively. A derangement

of α flip-flops can only alter an even number of bits in Y'_A since the bit values can only be modified by exchanging flip-flop pairs that hold different bit values. If $|S_0| = \alpha_0$, $|S_1| = \alpha_1 = \alpha - \alpha_0$, the number of ways to alter precisely β bit values in Y'_A can be derived in two steps. First, $\beta/2$ flip-flops from S_ϕ , $\phi = 0$ or 1 , are swapped with the same number of flip-flops from $S_{\bar{\phi}}$ to form two new sets, $\hat{S}_0$ and $\hat{S}_1$. Then, different derangements can be made from the unaltered $(\alpha_\phi - \beta/2)$ flip-flops in $\hat{S}_\phi$ and different permutations of $\beta/2$ exchanged flip-flops from $S_{\bar{\phi}}$. Since there are $P_{\beta/2}^{\beta/2}$ ways to permute $\beta/2$ flip-flops, the number of conditional derangements in the latter step for either set, $\hat{S}_\phi$ ($\phi = 0$ or 1) has the following recursion:

$$d(\alpha_\phi | \beta/2) = P_{\beta/2}^{\beta/2} \sum_{j=0}^{\beta/2} [C_j^{\beta/2} \cdot C_j^{\alpha_\phi - \beta/2} \cdot d(\alpha_\phi - \beta/2 | j)] \quad (4.13)$$

where $C_r^n = 0$ if $r > n$, $d(i | 0) = d_i$, $d(i | i) = P_i^i$ and $d(i | j) = 0$ if $j > i$.

Let E_1 be the event that exactly β bits in Y'_A are modified by an α -derangement. Then,

$$P(E_1) = \frac{C_{\beta/2}^{\alpha_0} \cdot C_{\beta/2}^{\alpha_1} \cdot d(\alpha_0 | \beta/2) \cdot d(\alpha_1 | \beta/2)}{d_\alpha} \quad (4.14)$$

The probability of E_1 can be expressed as a function of α_0 only since $\alpha_1 = \alpha - \alpha_0$. The bits altered in the event E_1 may or may not be the watermark bits. Let E_2 be the event that there are α_m watermarked flip-flops in S . Then,

$$P(E_2) = \frac{C_{\alpha_{m0}}^{m_0} \cdot C_{\alpha_{m1}}^{m_1} \cdot C_{\alpha_0 - \alpha_{m0}}^{N_0 - m_0} \cdot C_{\alpha_1 - \alpha_{m1}}^{N_1 - m_1}}{C_\alpha^N} \quad (4.15)$$

where α_{m0} and α_{m1} are the number of watermarked flip-flops that hold the “0” and “1” bits in S , respectively. Since $\alpha_{m1} = \alpha_m - \alpha_{m0}$, $P(E_2)$ can be evaluated if α_0 and α_{m0} are known.

Let E_3 be the event that γ watermark bits are modified by an α -derangement. The conditional probability of E_3 given that E_1 and E_2 have occurred is:

$$P(E_3 | E_1 \cap E_2) = \frac{\sum_{\gamma_0=0}^{\min(\gamma, \alpha_{m0})} C_{\gamma_0}^{\alpha_{m0}} \cdot C_{\beta/2-\gamma_0}^{\alpha_0-\alpha_{m0}} \cdot C_{\gamma_1}^{\alpha_{m1}} \cdot C_{\beta/2-\gamma_1}^{\alpha_1-\alpha_{m1}}}{C_{\beta/2}^{\alpha_0} \cdot C_{\beta/2}^{\alpha_1}} \quad (4.16)$$

where γ_0 and γ_1 are the number of “0” and “1” watermark bits, respectively, and $\gamma_1 = \gamma - \gamma_0$.

For a fixed value of α_0 , E_1 and E_2 are independent and $P(E_1 \cap E_2) = P(E_1)P(E_2)$. To alter γ watermark bits, there must be at least γ watermarked flip-flops and at least γ bits must be modified by an α -derangement. The maximum number of watermarked flip-flops in S , α_m is $\min(\alpha, m)$ and the maximum number of bits, β that can be altered in the event E_1 is $2 \times \min(\alpha_0, \alpha_1)$. By enumerating all possible combinations of parameters, α_0 , α_m , α_{m0} and β , the probability of modifying exactly γ watermark bits can be determined as follows:

$$P(E_3) = \sum_{\alpha_0=0}^{\min(\alpha, N_0)} \sum_{\alpha_m=\gamma}^{\min(\alpha, m)} \sum_{\alpha_{m0}=0}^{\min(\alpha_m, m_0)} \left\{ P(E_1) \cdot \sum_{\beta=\gamma}^{2 \min(\alpha_0, \alpha_1)} [P(E_2) \cdot P(E_3 | E_1 \cap E_2)] \right\} \quad (4.17)$$

Since at most $\min(\beta, \alpha_m)$ watermark bits can be altered, the probability of successfully removing at least i watermark bits by Bob is given by:

$$P_r(\gamma \geq i) = \sum_{\alpha_0=0}^{\min(\alpha, N_0)} \sum_{\alpha_m=i}^{\min(\alpha, m)} \sum_{\alpha_{m0}=0}^{\min(\alpha_m, m_0)} \left\{ P(E_2) \cdot \sum_{\beta=i}^{2 \min(\alpha_0, \alpha_1)} \left[P(E_1) \cdot \sum_{\gamma=i}^{\min(\beta, \alpha_m)} P(E_3 | E_1 \cap E_2) \right] \right\} \quad (4.18)$$

4.6.4 Unauthorized Addition

In order for Bob to embed his own watermark into Alice’s watermarked design, he has to identify and remove all the watermarked traces in the scan chain before he can embed his own. Even if Bob has the necessary tools to mount a successful removal attack, he will still need to repeat the watermarking process with a design not authored by him. This task is at least as difficult as the complete repetition of the test design and optimization process. Realistically, it is more possible for Bob to randomly locate

m watermarked positions in the scan chain without any attempt to remove Alice's watermark but just exchange the order of some flip-flops so that the output response, Y_B' to an input validation, X_B' selected by him contains his watermark, W_B . To succeed, he needs to modify substantive subcircuits and this is a difficult task as discussed earlier. In addition, if the value of $P_r(\gamma \geq i)$ is sufficiently low, after Bob's watermark addition, Alice can still correctly retrieve a significant amount of her watermark bits from Bob's watermarked design but the reverse is not possible for Bob.

4.7 Experimental Results and Discussions

The experimentation is performed with the help of several tools. The SIS⁹ tool is used for the identification of flip-flop relation and the DfT tool, DFTAdvisor¹⁰ and the ATPG tool, FastScan¹¹ from Mentor Graphics are used to generate the preliminary vectors, Y from X on the optimized scan design without the watermark. Watermarks of various lengths were embedded into sequential circuits obtained from ISCAS89, ISCAS99 and LGSynth93 benchmark suites. Both the originally optimized scan design and the watermarked design are synthesized using the algebraic script from SIS and technology mapped to the Mississippi State University standard cell library¹². All experiments were run on a 750-MHz Sun UltraSPARC-III with Solaris operating system and 2 GB of memory.

Tables 4.1 and 4.2 show the synthesis results of watermarking the ISCAS89, ISCAS99 and LGSynth93¹³ benchmark circuits. The first 32, 64 and 128 bits of the

⁹ SIS is an interactive tool for synthesis and optimization for sequential circuits. The SIS synthesis system supports a design methodology that searches a larger solution space than in previous algorithm [SEN92].

¹⁰ DFTAdvisor is the complete tool for preparing designs for scan test. It turns a gate-level netlist into a design that is ready for scan testing and test pattern compression. It is a perfect compliment for FastScan ATPG tools. It has a powerful feature set that facilitates hierarchical techniques and reuse.

¹¹ FastScan is a powerful, yet simple, tool. With its wide range of fault models, comprehensive design rules checks, extensive clocking support, and innovative algorithms for performance-oriented pattern compaction. FastScan is widely used in industry and is regarded as the most powerful ATPG tool.

¹² The MSU standard cell library contains an extensive library of standard cells.

¹³ These three benchmark suites include many typical sequential designs in different sizes. They provide a fair way for the evaluation and comparison of different optimization techniques. Also, quite a few sequential designs with long scan chains are included. They are useful to demonstrate the applicability of the proposed scheme to designs with reasonably long scan chain.

SHA-1 hash values of the ownership information serve as the applied watermarks. Different watermark lengths, m were selected for each design according to the scan chain length, N (i.e., $m < N/2$). A_{org} and D_{org} are the area in μm^2 and the delay in ns of the originally optimized design, respectively. A_{wm} and D_{wm} denote the area and delay of the watermarked design, respectively. ΔA and ΔD are the percentage increments from A_{org} to A_{wm} and that from D_{org} to D_{wm} , respectively. A positive percentage implies degradation. It is possible that the reordered scan chain performs better than the originally optimized scan chain without the watermark. The negative percentage in Table 4.1 implies the performance improvement due to watermarking. For most designs, the area and delay overheads are less than 3%. The average area has actually improved by 0.58% and the average delay overhead by 0.38% for the circuits with $N > 200$. For the nine larger circuits that have more than 500 flip-flops in the scan chain, the average area and delay have improved by 0.09% and 0.01%, respectively. As a whole, SFT watermarking incurs negligible overheads on both logic and timing for designs with reasonably long scan chain.

Table 4.1 Performance overheads on ISCAS89, ISCAS99 benchmarks.

Circuit	N	A _{org}	D _{org}	m	A _{wm}	D _{wm}	ΔA(%)	ΔD(%)
S3330	132	33544	47	32	33504	46.4	-0.12	-1.28
				64	33496	47	-0.14	0.00
S3384	183	47512	79.6	32	47736	83.2	0.47	4.52
				64	46160	83.6	-2.85	5.03
S9234	228	59424	64.4	32	59528	65.6	0.18	1.86
				64	59520	65.2	0.16	1.24
S6669	239	75096	134.6	32	75136	134.6	0.05	0.00
				64	75312	134.2	0.29	-0.30
S15850	597	130744	120.4	64	130928	123	0.14	2.16
				128	130880	123.6	0.10	2.66
S13207	669	126120	136.6	64	126024	140.2	-0.08	2.64
				128	123456	131	-2.11	-4.10
S38584	1426	344392	404.2	64	345872	400.6	0.43	-0.89
				128	342520	401.6	-0.54	-0.64
S38417	1636	373312	353	64	372288	354	-0.27	0.28
				128	373696	356.4	0.10	0.96
S35932	1728	365504	349.4	64	366528	348.2	0.28	-0.34
				128	366920	347	0.39	-0.69
B14	245	139648	142.4	32	139664	142.4	0.01	0.00
				64	139176	142.4	-0.34	0.00
B15_1	449	219208	148.0	64	219152	147.6	-0.03	-0.27
				128	218208	148.0	-0.46	0.00
B21	490	298720	156.6	64	298464	155.2	-0.09	-0.89
				128	298720	155.2	0.00	-0.89
B22	735	430456	210.6	64	429832	210.2	-0.14	-0.19
				128	430168	209.6	-0.07	-0.47
B17	1415	689888	405	64	690296	404.6	0.06	-0.10
				128	689840	403	-0.01	0.49

Table 4.2 Performance overheads on LGSynth93 benchmarks.

Circuit	N	A _{org}	D _{org}	m	A _{wm}	D _{wm}	$\Delta A(\%)$	$\Delta D(\%)$
sort	136	51376	151.2	32	51144	151.2	-0.45	0.00
				64	51152	151.4	-0.44	0.13
fir	142	91480	74.6	32	91504	74.6	0.03	0.00
				64	91616	74.6	0.15	0.00
elliptic	194	39200	56.6	32	39120	57.0	-0.20	0.71
				64	39080	68.4	-0.31	20.84
bigkey	224	87408	99.4	32	87416	99.4	0.01	0.00
				64	87408	99.4	0.01	0.00
dsip	224	66040	61.2	32	66432	61.2	0.59	0.00
				64	67192	63.8	1.74	4.25
psdes	230	60744	64.4	32	61080	64.6	0.55	0.31
				64	61248	65.2	0.83	1.24
r4000_32	249	86576	141	32	86584	141	0.01	0.00
				64	86640	140.8	0.07	-0.14
diffeq	305	77424	84.6	64	77048	82.2	-0.49	-2.84
				128	77032	82.4	-0.51	-2.60
mle	323	158824	100.6	64	138896	100.6	-12.55	0.00
				128	139064	100.6	-12.44	0.00
tseng	385	86224	115.2	64	85616	112.6	-0.71	-2.26
				128	85680	136.6	-0.63	18.58
valu	495	130760	137	64	130760	136.6	0.00	-0.29
				128	130776	137	0.01	0.00
pmac	590	129072	134.8	64	129032	134	-0.03	-0.59
				128	129248	134.4	0.14	-0.30
frisc	886	213216	230.6	64	213288	231	0.03	0.17
				128	213096	227.6	-0.06	-1.30

Currently, only one watermarking method on DfT [KIR98b] was proposed but there was no experimental result. Hence, the proposed method is compared with a recent dynamic watermarking scheme [OLI01] in Table 4.3. As introduced in Section 2.4.2, the STG in [OLI01] is implicitly manipulated so that the sequence of states traversed by an input sequence specified by the watermark exhibits a unique property that is rare in a non-modified STG. This STG modification can be obtained by a relatively simple manipulation of the circuit that realizes the original FSM. As the same synthesis tool and technology library were used, the area and delay results are excerpted from [OLI01] for the same benchmark circuits. Here the area is measured in terms of the number of literals to be consistent with [OLI01]. For each benchmark, the lower overhead results between the two methods are highlighted in bold print. In general, the proposed method incurs lower area overhead than [OLI01] but the delay

overhead is slightly higher. It should be noted that [OLI01] does not allow the ownership to be publicly authenticated in the field.

Table 4.3 Comparison of dynamic watermarking methods.

Circuit	<i>m</i>	ΔA (%)		ΔD (%)	
		proposed	[OLI01]	proposed	[OLI01]
S3330	64	-1.41	16.4	0.00	4.26
S3384	64	-2.97	7.34	5.03	-1.46
S9234	64	0.32	6.17	1.24	-1.50
S6669	64	0.54	6.83	-0.30	6.25
S15850	64	0.23	2.06	2.16	-2.28
	128	0.92	0.86	2.66	-3.70
S13207	64	-0.28	5.15	2.64	-3.38
	128	-3.24	3.40	-4.10	-3.38
S38584	64	0.53	0.75	-0.89	4.76
	128	-0.48	0.18	-0.64	5.60
S38417	64	-0.37	-5.17	0.28	-2.99
	128	0.47	-5.59	0.96	-2.99
S35932	64	1.25	-5.02	-0.34	2.35
	128	1.47	-5.35	-0.69	1.95

During the placement and routing, the flip-flops that are constrained to be placed near each other in the layout for short functional interconnects should also be placed near each other to produce short scan interconnects. Thus, the watermarked scan chains ordered by SfT are expected to have near-optimal interconnections. It has been shown in [COX94, NOR96] that SfT can achieve better area reduction before physical synthesis than the SFF-based scan design. In fact, the area savings after routing is even greater than that before routing [NOR96]. The area savings due to logic sharing and the reduction in interconnect lengths by SfT is more than sufficient to offset the penalty due to the watermark constraint, if any, on placement and routing operations. SfT-based watermarking is more promising when it is applied at RTL. Architectural synthesis and logic optimization will strive to reuse logic on existing functional paths whenever possible to eliminate the negative impact of scan on circuit performance so that there is little need for re-optimization after placement and routing [HUA01]. This also indirectly implies that it is more difficult to randomly modify the scan chain logic of an already watermarked design without violating the design specifications.

The strength and robustness of watermark is evaluated in Table 4.4. It is observed that as the scan chain length increases, the probability of coincidence, P_c decreases rapidly. When $N > 600$ and $m = 128$, P_c reduced to less than 1.0×10^{-170} . The false positive rate, P_λ is also determined empirically by 200 randomly generated input vectors. The ATPG tool is used to obtain the output response to each random test vector. None of the output responses was detected with a perfectly matched watermark, i.e., $P_\lambda(\tau = 1) = 0$, for each circuit. Besides, test patterns for fault detection were also generated by the FastScan for each watermarked design. Again, no matched watermark was found in their output responses. It is thought to be reasonable that a sufficiently low probability (below 1% as suggested by [OLI01]) is adequate to prove the authorship and make the denial attacks unsustainable. Hence, the watermark correlation is reduced from 100% to 75% match. It was found that for $\tau = 0.75$, $P_\lambda < 1\%$ for all the watermarked designs for both the randomly generated patterns and the tool-generated test patterns. When τ is reduced to 0.625, 16 out of 54 watermarked designs have $P_\lambda \geq 1\%$ for randomly generated patterns.

Table 4.4 Watermark strength and probability of removal.

Circuit	N	N_0	N_1	m	P_c	P_λ (%)		$P_\lambda(\gamma \geq m/4)$
						$\tau = 0.75$	$\tau = 0.625$	
S3330	132	66	66	32	5.1E-41	0.00	0.00	2.6E-02
				64	1.5E-58	0.00	0.00	5.1E-01
S3384	183	92	91	32	4.4E-46	0.00	7.50	3.4E-03
				64	3.2E-70	0.00	0.00	7.1E-02
S9234	228	110	118	32	2.1E-49	0.00	0.00	7.4E-04
				64	1.5E-77	0.00	0.00	9.3E-03
S6669	239	133	106	32	4.2E-50	0.00	5.50	6.7E-04
				64	4.6E-79	0.00	0.80	6.4E-03
S15850	597	305	292	64	5.0E-107	0.00	1.00	5.6E-08
				128	1.3E-172	0.00	0.00	8.1E-07
S13207	669	335	334	64	2.3E-110	0.00	0.00	1.1E-08
				128	1.1E-179	0.00	0.00	4.7E-08
S38584	1426	728	724	64	1.2E-132	0.00	0.50	1.0E-13
				128	6.7E-226	0.00	0.00	3.2E-17
S38417	1636	815	821	64	5.0E-136	0.00	0.00	1.7E-14
				128	8.1E-233	0.00	0.00	9.5E-19
S35932	1728	849	879	64	1.4E-137	0.00	0.00	7.0E-15
				128	5.5E-236	0.00	0.00	1.7E-19
B14	245	135	110	32	1.8E-50	0.00	0.00	5.6E-04
				64	7.3E-80	0.00	0.00	4.9E-03
B15_1	449	219	230	64	1.39E-98	0.00	0.00	2.7E-06

				128	2.0E-154	0.00	0.00	3.1E-04
B21	490	255	235	64	3.4E-101	0.00	5.00	8.9E-07
				128	4.3E-160	0.00	0.50	7.2E-05
B22	735	378	357	64	4.2E-113	0.00	1.00	2.8E-09
				128	1.9E-185	0.00	0.00	5.2E-09
B17	1415	696	719	64	6.6E-132	0.00	2.50	1.5E-13
				128	2.1E-224	0.00	0.00	6.1E-17
sort	136	71	65	32	1.7E-41	0.00	7.50	2.3E-02
				64	1.2E-59	0.00	0.50	4.6E-01
fir	142	72	70	32	3.6E-42	0.00	1.10	1.7E-02
				64	2.9E-61	0.00	0.00	3.7E-01
elliptic	194	85	109	32	5.7E-47	0.6	6.90	1.9E-03
				64	3.3E-72	0.00	2.90	3.9E-02
bigkey	224	116	108	32	3.9E-49	0.00	0.00	9.3E-04
				64	5.8E-77	0.00	0.00	1.2E-02
dsip	224	108	116	32	3.9E-49	0.00	0.00	8.3E-04
				64	5.8E-77	0.00	0.00	1.1E-02
psdes	230	122	108	32	1.6E-49	0.00	2.50	8.1E-04
				64	8.0E-78	0.00	0.00	9.1E-03
r4000_32	249	118	131	32	1.0E-50	0.00	0.00	3.9E-04
				64	2.2E-80	0.00	0.00	3.7E-03
diffeq	305	141	164	64	8.6E-87	0.00	1.50	3.6E-04
				128	5.1E-128	0.00	0.00	1.1E-01
mle	323	140	183	64	1.4E-88	0.00	3.50	1.7E-04
				128	4.2E-132	0.00	0.00	4.4E-02
tseng	385	183	202	64	6.0E-94	0.00	5.10	2.0E-05
				128	3.2E-144	0.00	0.50	4.5E-03
valu	495	225	270	64	1.7E-101	0.00	1.50	6.8E-07
				128	9.6E-161	0.00	0.00	2.9E-05
pmac	590	312	278	64	1.1E-106	0.00	6.40	6.8E-08
				128	7.1E-172	0.00	0.90	1.3E-06
frisc	886	470	416	64	1.6E-118	0.00	0.00	1.9E-10
				128	9.4E-197	0.00	0.00	4.7E-11

Based on the above results, it is reasonable to assume that when more than three quarters of the watermark bits are matched, the authorship proof is still veracious. In other words, if more than 25 percent of bits are corrupted, the detected watermark is invalid. For the evaluation of the watermark robustness against removal attack, it is conservatively presumed that $\alpha = m$ flip-flops in the scan chain can be deranged with reasonable effort and without significantly degrading the performance. The probability, $P_r(\gamma \geq m/4)$ of successfully removing at least one quarter of the watermarked bits by a random derangement of m flip-flops is shown in Table 4.4. It is found that the designs with larger ratio of N/α are more resilient to removal attack. For the designs with

more than 1500 scan flip-flops, the probability of modifying more than one quarter of the watermark bits (for $m = 128$) is lower than 3.2×10^{-17} . The results indicate that for moderate and long scan chain length, the proposed watermarking scheme is robust against the removal attack.

4.8 Summary

IP infringement poses a significant threat to design reuse, yet the protection and deterrent of high-tech IP privacy by non-law enforcement methods have not been widely adopted by the VLSI and EDA industry. One possible cause is the disparity in the interest of IP protection and the priority of IP integration. The problem can be addressed by reducing the overheads of IP protection and introducing a more open protection mechanism to effectively detect and track the IP usage. In this chapter, an easy-to-trace IP watermarking scheme was proposed to enable IP providers to protect their IPs against unauthorized use, and to detect and trace the reproduction and redistribution of IPs. It is the first ever approach that uses SfT to protect firm and hard IPs. The watermark is embedded based on the scan chain ordering problem. Hence, it can be detected directly in the field even after the IP cores have been integrated and packaged into chip. Unlike the SFF-based DfT watermarking, the method requires no specialized cell library and is applicable to the protection of IP cores in programmable logic devices. The logic synthesis and optimization performed upon watermarking the scan chain completely blend the test logic with the normal logic functions, making it expensive to succeed in modifying the test circuit to create a useful fraudulent IP. The empirical evidence has been presented that the watermarked designs incur very low area and timing overheads and possess very low rate of false positives. The probability of watermark removal based on flip-flop derangement has also been derived. The results indicate that the removal attacks are likely to succeed only on relatively short scan chains. The probability of successful random obliteration descends rapidly with the increased scan chain length. Also, the scan chain ordering does not affect the original fault coverage. The original sets of test input vectors and output responses just need to be reordered according to the permuted scan chain order to complete the test.

Additionally, an identity-based signature scheme has been also proposed to enable multiple IP cores watermarked by different authors in an SoC to be publicly authenticated in the field by the IP buyers and users. The application of this cryptographically-proven protocol on the proposed method makes tracking of fraudulence activities of IP licensees feasible, hence effectively deters illegal IP distribution.

As scan cells can be easily identified from the design by the tool, the watermark imposed on the DfT stage can be removed once the test structure is deleted successfully. To a certain extent, the SfT method proposed in this chapter has reduced the vulnerability of scan chain watermarking. Alternatively, it is also possible to augment the conventional DfT watermarking with a different watermarking scheme at other higher design abstraction levels to overcome its weaknesses. In the next chapter, the integration of FSM watermarking with DfT watermarking will be explored to make the watermark detection easier and its removal more difficult.

Chapter 5

A New Hybrid Watermarking Scheme

5.1 Introduction

Conceptually, when the signature is embedded at a higher level of design abstraction, it is more tedious to detect the presence of watermark in the downstream design but it is also harder for the attacker to erase the watermark in the downstream design by redundancy removal or other forms of logic manipulation. It is therefore sensible to independently watermark a design at multiple abstraction levels in a design flow. Multi-level watermarking increases the difficulty of a single successful attack on one particular level of abstraction to remove the watermark embedded at a different level, so that a breach of one level of watermark will alert the IP owner to trace the forgery using the watermarks embedded at other levels.

In this chapter, a new hybrid IP watermarking scheme is explored. The watermarking processes at two different levels are coherently associated by the reachability information of sequential circuit. At behavioral synthesis level, the watermark is embedded in the STG (state transition graph) of the FSM (finite-state machine) to enable the authorship to be verified directly by running the watermarked FSM with a specific code sequence. In the backend, scan cells replace the sequential elements at the DfT stage to provide controllability and observability of the circuit nodes from the primary input and output pins outside the chip, which makes authentication of the encapsulated IP in the field possible. The scan chain can be further watermarked to provide a second level of protection with a fragile watermarking feature. The complementary protection modes at different abstraction levels will make the authorship proof harder to erase and the IP forgery easier to trace.

The remaining sections of this chapter are organized as follows. In Section 5.2, the watermarking on the FSM is proposed and an example is given. The watermarking process on the scan chain ordering is elaborated in Section 5.3. The watermarking strength is evaluated and its resilience to various typical attacks is analyzed in Section 5.4. The experimental results are presented and discussed in the next section. Finally, the chapter is summarized in Section 5.5.

5.2 Finite State Machine Watermarking

At behavior level, STG representation makes watermarking FSMs in industrial designs promising as efficient sequential logic synthesis tools and optimization methods are available to lower the cost of embedding and detection of watermark. FSM watermarking has the advantage that the IP author signature can be lucidly recovered by applying a verification code sequence. As the STG is in general exponentially larger than the circuit description itself, such scheme also has high resilience against tampering at lower abstraction levels.

5.2.1 Preliminaries

A finite state machine (FSM) is a sequential circuit characterized by a set of internal states and transitions between them. A formal and concise definition of an FSM is given in [PEN99] as follows:

Definition 5.1: An FSM is a tuple $M = (\Sigma, \Delta, Q, s_0, \delta, \lambda)$, where Σ and Δ are the finite, non-empty sets of input and output alphabets, respectively. Q is a finite, non-empty set of states and $s_0 \in Q$ represents a unique reset state. $\delta(s, X): Q \times \Sigma \rightarrow Q \cup \{\emptyset\}$ is the state transition function and $\lambda(s, X): Q \times \Sigma \rightarrow \Delta \cup \{\tau\}$ is the output function, where $\emptyset$ denotes an unspecified state and τ denotes an unspecified output.

For $s_i, s_j \in Q$, s_j is said to be the next state of s_i if $\exists X \in \Sigma$ s.t. $s_j = \delta(s_i, X)$. The application of X on s_i also produces an output, $Y = \lambda(s_i, X) \in \Delta$. For an FSM with n input and k output variables, each input alphabet, $X = x_1 x_2 \dots x_n$ is a string of n bits and each output alphabet, $Y = y_1 y_2 \dots y_k$ is a string of k bits. Each bit, $x_i, y_i \in \{0, 1, -\}$, where “0” and “1” are the binary constants, and “-” denotes a “don’t care” value. For the simplicity of the notation, an upper case letter is used to denote an input or output

alphabet in Σ and Δ , a lower case letter is used to denote an input or output variable whose value is in $\{0, 1, -\}$, and y_{ij} to address the j -th bit of the i -th alphabet, Y_i .

An FSM is usually designed with the state transition graph (STG).

Definition 5.2: An STG, $STG(M) = G(V, E)$ is a labeled directed graph of M . Each symbolic state, $s \in Q$ is represented by a node in V . A state transition, t , from a source node, $S(t)$, to a destination node, $D(t)$, is represented by a directed edge, $e_{ij} \in E$ connecting $S(t)$ to $D(t)$. Each edge is tagged with an input/output label, $I(t)/O(t)$, to encapsulate the relations $D(t) = \delta(S(t), I(t))$ and $O(t) = \lambda(S(t), I(t))$.

Definition 5.3: A state transition, t , can be represented by a quadruple $(S(t), D(t), I(t), O(t))$. The input combinations that are absent from all transitions of a source state in an STG are called the *free (or unspecified) input combinations* of that state, and a transition that can be created from the free input combinations is called an unspecified transition.

Unlike [OLI01], as the number of states in an FSM is a dominant factor of the implementation complexity, only the properties of the edge set are modified to synthesize the watermarked design in order to preserve the nodes in $STG(M)$ [ABD05b]. The FSM watermarking scheme using output mapping algorithm [ABD05b] takes advantage of the original transitions in the STG to lower the overhead of watermarking. The embedding process is fast as no special effort is made to search the states of STG. These merits [ABD05b] are inherited by the proposed method while some improvements are made to overcome the limitations of method in [ABD05b].

In light of dynamic watermarking, the watermark detection process involves the extraction of an output sequence, $\hat{Y} = \{\hat{Y}_1, \hat{Y}_2, \dots, \hat{Y}_N\}$, $\hat{Y}_i \in \Delta$, from the watermarked design, $\hat{M}$ by applying a specific input sequence, $\hat{X} = \{\hat{X}_1, \hat{X}_2, \dots, \hat{X}_N\}$, $\hat{X}_i \in \Sigma$, on a state, $\hat{s} \in Q$ such that $\hat{Y} = \lambda(\hat{s}, \hat{X}) = \lambda(\delta(\delta(\dots \delta(\hat{s}, \hat{X}_1) \dots), \hat{X}_{N-1}), \hat{X}_N)$. The

watermark synthesis process requires that the outputs of $\hat{M}$ be compatible with the outputs of M for every input symbol, $X \in \Sigma$, and the output mappings of $\hat{M}$ for every input symbol, $\hat{X}_i \in \Sigma, \forall i = [1, N]$ be dictated by a signature that identifies the ownership of the design. The signature is cryptographically generated from a secret key so that $\hat{Y} = \lambda(\hat{s}, \hat{X})$ becomes a unique property of $\hat{M}$.

In [TOR00, ABD05b], the length, N , of $\hat{X}$ and $\hat{Y}$, is equal to m/k , where m is the watermark length and k is the number of output variables of the FSM. When k is large, it becomes very difficult to make use of existing transitions to reduce the overhead of watermarking due to the low probability of output coincidence. Moreover, as all output bits are watermarked in consecutive transitions after the starting state on which $\hat{X}$ is applied, the watermarked transitions are not well obfuscated, causing the watermarked FSM to be vulnerable. Therefore, N is set greater than m/k so that not all bits in $\hat{Y}$ are watermarked. The locality of the watermark is randomized by a cryptographic one-way function such that any number (from 1 to k) of bits at any output bit from any transition of the STG is equally probable to be watermarked. To increase the watermark strength and minimize the next state decoder logic of the watermarked design, the method also capitalizes on the extra headroom created by the pseudo input variables and the free input combinations of the FSM. With these preliminaries, the proposed FSM watermarking algorithm will be elaborated.

✱

5.2.2 Generation of Watermark and Random Sequence

A meaningful text string, MSG is first encoded into a binary string and then encrypted by a provable cryptographic algorithm with the private key, K_e , of the IP owner. If the length of the encrypted message is too long, a message digest (MD) algorithm, SHA-1 [MEN96] can be used to reduce its length. The resultant binary bit vector of length m is the watermark, $W = \{w_i\}_{i=1}^m$ and $w_i \in \{0, 1\}$.

A keyed one-way PNG is used to generate a sequence, $B = \{b_i\}_{i=1}^m$ of m unique integers between 1 and $N \times k$, i.e., $b_i \in [1, N \times k] \forall i = 1, 2, \dots, m$ and $b_i \neq b_j \forall i \neq j$. The

length, N of the sequence, $\hat{\mathbf{X}}$ is determined empirically. The purpose of B is to randomly disperse the m watermark bits into $\hat{\mathbf{Y}}$. If $\exists(i, j) \forall i \in [1, N]$ and $j \in [1, k]$ such that $(i-1)k + j = b_l$, then $\hat{y}_{i,j} = w_l$ where $\hat{y}_{i,j}$ is the j -th bit of $\hat{Y}_i \in \hat{\mathbf{Y}}$. The secure hash algorithm, SHA-1 [MEN96] is used as an MD as well as in a keyed one-way PNG for the generation of these two random sequences, W and B . As it is computationally infeasible to find a collision of this hash function, the possibility that the same group of numbers is generated by coincidence is extremely low without the knowledge of the secret key.

5.2.3 Watermarking the STG

The watermark, W is inserted into $STG(M)$ by modifying some of its edges without changing the operational behavior of M to find a sequence of N consecutive transitions, $\hat{t}_i = (\hat{s}_i, \hat{s}_{i+1}, \hat{X}_i, \hat{Y}_i)$, $i = 1, 2, \dots, N$, such that each watermark bit, $w_l \in W$, $l \in [1, m]$, will be randomly mapped to one bit in the sequence, $\hat{\mathbf{Y}} = \hat{Y}_1 \hat{Y}_2 \dots \hat{Y}_N = \hat{y}_{1,1} \dots \hat{y}_{1,k} \hat{y}_{2,1} \dots \hat{y}_{2,k} \dots \hat{y}_{N,1} \dots \hat{y}_{N,k}$. The mapping from W to $\hat{\mathbf{Y}}$ is injective but not surjective. The value of each bit, $\hat{y}_{i,j}$, in $\hat{\mathbf{Y}}$ can be determined as follows: if $(i-1)k + j = b_l$, then $\hat{y}_{i,j} = w_l$, else $\hat{y}_{i,j} = \text{"-"}$.

Given an output, $\hat{Y}_i$, and a source state, $\hat{s}_i$, the destination state, $\hat{s}_{i+1}$ of the watermarked transition, $\hat{t}_i$, will be determined by an output compatibility check. Two bits, $x, y \in \{0, 1, -\}$, are compatible if they are of equal value or one of them has a *don't care* value, i.e., $x \cap y \neq \emptyset$. This intersection of two ternary variables is defined in Table 5.1. Likewise, two alphabets, X and Y are compatible, denoted by $X \equiv Y$, if none of the elements in $X \cap Y = \{x_i \cap y_i\}$ has a null value.

Table 5.1 Intersection of two ternary variables.

$\cap$	0	1	–
0	0	$\emptyset$	0
1	$\emptyset$	1	1
–	0	1	–

Starting with $i = 1$, an arbitrary state, $\hat{s}_i \in Q$, is selected. Let $T(\hat{s}_i)$ be the set of transitions emanating from a state, $\hat{s}_i$. A set of transitions, $C(\hat{s}_i)$, that is output compatible with $\hat{Y}_i$ is sought, i.e., $C(\hat{s}_i) = \{t_i \in T(\hat{s}_i) \mid O(t_i) \equiv \hat{Y}_i\}$. To avoid entering into a deadlock, transitions terminate at a deadlock state (i.e., state with no fanout) are excluded from $C(\hat{s}_i)$. Four distinct scenarios are considered for the determination of $\hat{t}_i$.

Cases 1: There is only one output compatible transition, $|C(\hat{s}_i)| = 1$, then $\hat{t}_i = C(\hat{s}_i)$ and $\hat{s}_{i+1} = D(\hat{t}_i)$.

Case 2: If more than one output compatible transition are found, i.e., $|C(\hat{s}_i)| > 1$, then a transition from $C(\hat{s}_i)$ whose next state has the highest number of free input combinations will be selected as $\hat{t}_i$. Its output will be modified to $O(\hat{t}_i) = O(\hat{t}_i) \cap \hat{Y}_i$ and $\hat{s}_{i+1} = D(\hat{t}_i)$.

Case 3: If $|C(\hat{s}_i)| = 0$, then the free input combinations of $\hat{s}_i$ will be considered. Let $F(\hat{s}_i) = \{X \in \Sigma \mid \delta(\hat{s}_i, X) = \emptyset\}$ be the set of free input combinations of $\hat{s}_i$. For $F(\hat{s}_i) \neq \emptyset$, let $D(\hat{s}_i) = \{\hat{s}_j \in Q \mid \hat{s}_j = D(\hat{t}_i) \forall \hat{t}_i \in T(\hat{s}_i)\}$ be the set of all destination states of $\hat{s}_i$. $\hat{s}_{i+1}$ is set to the state with the highest number of free input combinations in $D(\hat{s}_i)$ (excluding the deadlock states) unless $D(\hat{s}_i) = \emptyset$. When $D(\hat{s}_i) = \emptyset$, $\hat{s}_{i+1}$ is set to the state with the highest number of free input combinations in $STG(M)$. If there exists an edge connecting $\hat{s}_i$ to $\hat{s}_{i+1}$ in $STG(M)$, a new input/output pair, $I(\hat{t}_i)/O(\hat{t}_i)$ is added for the transition, $\hat{t}_i$. Otherwise, a new edge directed from $\hat{s}_i$ to $\hat{s}_{i+1}$ labeled with $I(\hat{t}_i)/O(\hat{t}_i)$ will be created in $STG(M)$ for $\hat{t}_i$. $O(\hat{t}_i) = \hat{Y}_i$ and the determination of $I(\hat{t}_i)$

will be explained later.

Case 4: If $|C(\hat{s}_i)| = 0$ and $F(\hat{s}_i) = \emptyset$, then a pseudo input variable, x_{n+1} , needs to be introduced in M and the number of input variables, n , is increment by 1. x_{n+1} is set to an unspecified logic value “*” for all existing transitions. A new edge directed from $\hat{s}_i$ to $\hat{s}_{i+1}$ labeled with $I(\hat{t}_i)/O(\hat{t}_i)$ will be created for $\hat{t}_i$. $\hat{s}_{i+1}$ is set to the state with the highest number of free inputs in $D(\hat{s}_i)$ or in $STG(M)$ if $D(\hat{s}_i) = \emptyset$, and $O(\hat{t}_i) = \hat{Y}_i$. Both symbols “*” and “-” can assume either a logic “0” or a logic “1” value but there is a subtle difference. “-” is meant for the currently used input combinations whereas “*” can be associated with either the used or free input combinations. A “*” can be construed as a reserved free input literal as its logic state (“0” or “1”) will only be defined at the time when some input combinations subsumed by it are freed to become $I(\hat{t}_i)$.

When there is no existing transition with compatible output, as in Cases 3 and 4, the input alphabet, $I(\hat{t}_i)$ for $O(\hat{t}_i) = \lambda(\hat{s}_i, I(\hat{t}_i)) = \hat{Y}_i$ needs to be determined. $I(\hat{t}_i)$ is set to one of the free input combinations of $\hat{s}_i$ if no “*” appears in all the used input combinations of $\hat{s}_i$. Otherwise, an alphabet, $X \subseteq I(t_u)$, $t_u \in T(\hat{s}_i)$, that contains at least one “*” from the set of used input combinations of $\hat{s}_i$ will be split into two. Initially, $I(\hat{t}_i) = X$. A “*” bit in X is selected and assigned a fixed but randomly generated binary constant, $a \in \{0, 1\}$ while the corresponding “*” bit in $I(\hat{t}_i)$ is assigned its complement, $\bar{a}$. Meantime, all the “-” bits in $I(\hat{t}_i)$ are replaced by the “*” bits. For example, if $X = “1-”$, and $a = 0$, then it will be split into $X = “1-0”$ and $I(\hat{t}_i) = “1*1”$. As the number of transitions with “*” bits in the pseudo input variable space outnumbers those in the original input variable space enormously, to simplify the next state and output decoder design, it is lucrative to preserve “*” in the pseudo input variable space whenever free input combinations from the original variable space can be used to produce $I(\hat{t}_i)$. In the search for the next state, $\hat{s}_{i+1}$, an input alphabet with j exclusive “*” bits is considered as a cover of 2^{j-1} free input combinations. When two states possess the same highest number of free input combinations, preference will be

given to the state that covers the highest number of output combinations in its fan out transitions.

The above watermarking process is repeated for $i = 2$ to N until $\hat{i}_N$ is determined. The residual “*” in the input alphabets of all edges will be replaced with “-” and the resultant $STG(M)$ is the watermarked $STG(\hat{M})$ and $\hat{X} = I(\hat{i}_1)I(\hat{i}_2)\cdots I(\hat{i}_N)$. If the overhead of watermarked design is not satisfactory, the entire process can be repeated with an adjusted value of N . The overall watermark-insertion flow is shown in Figure 5.1.

The pseudo input variable is introduced in a fundamentally different way from [ABD05b]. For each pseudo input variable added, at least 2^{n-1} potential free input combinations are created in every state transition, where n here refers to the total number of input variables including the pseudo variables. These free input combinations have been consumed in [ABD05b] by fixing the value of each pseudo input variable to be “0” consistently for all existing transitions and “1” consistently for the watermarked transition immediately upon its creation. This has not only increased the complexity of the decoders, but also made the watermarked transition discernible from the pseudo inputs. The introduction of the reserved free literal allows the assignments of “*” in the input alphabets of all transitions to be deferred until some input combinations subsumed by it are needed to watermark a transition. The transformation of “-” to “*” in the $I(\hat{i}_i)$ when a random assignment is made on “*” serves two important purposes. First, it judiciously preserves the *don't care* inputs in the transitions to optimize the design of the next state and output decoders. Second, it allows the same edge to be revisited for watermarking to maximally exploit the free input combinations. This will minimize the required number of pseudo input variables, especially when a long watermark is to be embedded for a strong authorship proof.

Figure 5.1 Flowchart of FSM watermarking.

The number of transitions, N has no bearing on the probability of coincidence but it has an impact on the cost of watermarking. If N is small, the probability of finding compatible outputs from existing transitions is low and more design overhead will be incurred. On the other hand, if N is large, fewer new transitions and pseudo inputs need to be added which will lower the cost of watermarking, but the code sequence required to detect the watermark is long. To avoid introducing an excessive number of unspecified transitions due to the addition of pseudo input variables, N needs to be sufficiently larger than m/k . When $N \approx m$, each output alphabet in $\hat{\mathbf{Y}}$ contains one watermark bit on average and the resultant watermarked design generally possesses acceptably low overhead. As the proposed embedding algorithm can run very quickly even for large FSM, the watermarking process can be repeated for different N to select an optimal watermarked design with reasonable verification code length. This is different from [ABD05b] in which the watermarked design is accepted passively in

one pass regardless of the quality of the solution. The following procedure is suggested to legitimately limit the number of trials. Let A_{wm_i} denote the area of the watermarked FSM with $N = N_i$ at the i -th trial. $N_i = N_{i-1} \pm \delta_i$ and $N_1 \approx m$. N_i is incremented (or decremented) by δ_i depends on the extent to which $A_{wm_{i-1}}$ is increased (or reduced) over the previous trial. The standard deviation, σ_i , of A_{wm} is defined as:

$$\sigma_i = \sqrt{\frac{1}{i} \sum_{j=1}^i (A_{wm_j} - \overline{A_{wm}})^2} \quad (5.1)$$

where $\overline{A_{wm}} = \frac{1}{i} \sum_{j=1}^i A_{wm_j}$ is the mean area of the trial watermarked FSMs. The trial terminates when $\sigma_i / A \leq \varepsilon$, i.e., when the design overhead is acceptable, or when $N \geq N_{max}$, where ε is a small preset value, A is the area of the FSM before watermarking and N_{max} is some preset limit on the verification code length. The least overhead watermarked design from among the trials is selected.

5.2.4 Detection of Watermark

To verify the authorship, one needs to run the watermarked FSM with the input sequence $\hat{\mathbf{X}} = \{\hat{X}_1, \hat{X}_2, \dots, \hat{X}_N\}$ applied on state $\hat{s}_1$. If the operation halts before N transitions, the watermark cannot be detected. Otherwise, an output sequence, $\tilde{\mathbf{Y}}$ of $N \times k$ bits is obtained. The bits indexed by the set, B , of m random numbers, are selected from $\tilde{\mathbf{Y}}$ to form an ordered sequence, $\tilde{W}$. The authorship is proved if $\tilde{W}$ perfectly matches or is highly correlated with the watermark, W of the IP owner.

5.2.5 An Example of the Proposed FSM Watermarking

The STG of a simple FSM to be watermarked is shown in Figure 5.2(a). It has five states, represented mnemonically as $Q = \{s_1, s_2, s_3, s_4, s_5\}$. Assume that the encrypted watermark, $W = "110110"$. The number of output labels to be mapped, N should be greater than $6/2 = 3$ as $m = 6$ and $k = 2$. Let $N = 7$. Suppose the set of six random numbers between 1 and 14 ($k \times N$) generated by the PNG with the IP owner's secret

key is, $B = \{9, 4, 2, 7, 12, 3\}$.

From Section 5.2.2, the $\hat{y}_{i,j}$ with indices satisfying the equality $(i-1)k+j = b_l$ will be mapped to the watermarked bits. Since $2(1-1) + 1 = 1 \notin B$, $\hat{y}_{1,1} = \text{"-"};$ Since $2(1-1) + 2 = 2 = b_3$, $\hat{y}_{1,2} = w_3 = \text{"0"};$ $3 = b_6 \Rightarrow \hat{y}_{2,1} = w_6 = \text{"0"};$ $4 = b_2 \Rightarrow \hat{y}_{2,2} = w_2 = \text{"1"};$ $5 \notin B \Rightarrow \hat{y}_{3,1} = \text{"-"};$ $6 \notin B \Rightarrow \hat{y}_{3,2} = \text{"-"};$ $7 = b_4 \Rightarrow \hat{y}_{4,1} = w_4 = \text{"1"};$ $8 \notin B \Rightarrow \hat{y}_{4,2} = \text{"-"};$ $9 = b_1 \Rightarrow \hat{y}_{5,1} = w_1 = \text{"1"};$ $10 \notin B \Rightarrow \hat{y}_{5,2} = \text{"-"};$ $11 \notin B \Rightarrow \hat{y}_{6,1} = \text{"-"};$ $12 = b_5 \Rightarrow \hat{y}_{6,2} = w_5 = \text{"1"};$ $13 \notin B \Rightarrow \hat{y}_{7,1} = \text{"-"};$ and $14 \notin B \Rightarrow \hat{y}_{7,2} = \text{"-"}.$ Hence, $\hat{Y} = \text{"-0 01 -1 -1 -1 -"}.$

(a) Original FSM.

(b) Use of existing transition.

(c) Introduction of pseudo input variable and new transition.

(d) The watermarked FSM.

(e) Excitation of watermarked transitions.

Figure 5.2 Example of watermarking on FSM.

An arbitrary starting state, $\hat{s}_1 = s_1$ is selected to commence the watermarking process. For $\hat{Y}_1 = \text{"-0"}$, $C(\hat{s}_1) = \{s_1, s_2, s_4\}$ and none of them has any free input combination. $\hat{s}_2$ is set to s_1 arbitrarily and $\hat{Y}_1 = \text{"00"}$. $\hat{t}_1$ is marked by a heavy edge in Figure 5.2(b). For $\hat{Y}_2 = \text{"01"}$, there is no compatible output from $T(\hat{s}_1)$ and $C(\hat{s}_1) = \emptyset$. Since s_3 has the most free input combinations among $D(\hat{s}_2)$, a new transition from s_1 to s_3 is added. As $F(\hat{s}_2) = \emptyset$, a pseudo input variable, x_3 , is introduced. It assumes a value of $\text{"*"} on the inputs of all existing transitions. Suppose the input of the transition, $t = (s_1, s_3, \text{"10*"}, \text{"11"})$ is split into $I(t) = \text{"101"}$ and $I(\hat{t}_2) = \text{"100"}$. The new transition, $\hat{t}_2 = (s_1, s_3, 100, 01)$ is added into the $STG(M)$ as indicated by a dotted edge in Figure 5.2(c). For $\hat{Y}_3 = \text{"—"}$, $C(\hat{s}_3) = \{s_2, s_4\}$. Both states have an equal number of free input combinations but s_4 is preferred over s_2 as s_4 covers more output combinations ("01" and "11") in its fanout transitions than that ("01") of s_2 . Therefore, $\hat{t}_3 = (s_3, s_4, \text{"11*"}, \text{"10"})$. The process continues until all seven transitions are identified. Then all residual $\text{"*"} in the final STG are changed to "—" . The watermarked STG is shown in Figure 5.2(d) where Cases 1 and 2 transitions are marked by heavy edges and the added transitions of Cases 3 and 4 are marked by dotted edges. Figure 5.2(e) shows the complete watermarked sequence of inputs and outputs and the transitional states. The overhead of the synthesized watermarked FSM is checked. If it is not satisfactory, N is modified and the watermarking process is repeated.$$

To verify the existence of the watermark, W , an input sequence $\hat{X} = (\text{"01-"}, \text{"100"}, \text{"11-"}, \text{"0-0"}, \text{"1—"}, \text{"00-"}, \text{"00-"}, \text{"—"}) \in \{0,1\}$, is applied on state s_1 of the FSM. A binary stream, $\tilde{W}$, is retrieved from bit positions 9, 4, 2, 7, 12, 3 of the output sequence, $\hat{Y}$. If $\tilde{W} = W = \text{"110110"}$, the authorship is proved.

5.3 Watermarking on Scan Chain Ordering

Although the ownership can be authenticated directly by running the watermarked FSM with $\hat{X}$, it does not permit the IP authorship to be field authenticated by the IP buyers after the watermarked FSM has been integrated into an integrated circuit and

packaged. Since only the test signals can be traced after the chip is packaged, the authorship proof of the watermarked FSM can be verified off chip by making it a part of the test kernel. The scan path provides controlled accesses to all internal states and combinational circuits of the watermarked IP. A sequence of test vectors can be applied serially through the scan-in, S_{in} pin to bring $\hat{M}$ to the designated state, $\hat{s}_1$ in the test mode, followed by N designated test vectors that incorporate $\hat{X}$. The output responses, $\tilde{Y}$ can then be collected serially from a scan-out, S_{out} pin externally to verify the authenticity of $\hat{M}$.

Since scan chain is used as a medium to aid authorship verification of the IP encapsulated in the test kernel, for security, it should also be independently watermarked to boost the confidence in positive watermark identification. While the detection of the authorship of the design at a higher abstraction level is facilitated by the scan chain design, the chain selection tool developer is also clearly identified by the signature embedded in the solution of the backend DfT process. Failure to detect the chain selection signature alerts malicious tampering or removal of the test structure in attempt to misappropriate the protected IP. By watermarking the scan chain, the aggressor needs additional effort to also successfully tamper or redesign the test structure to provide the fault coverage of the pirated IP. Therefore, it is imperative that the protocol and information provided for the detection of either signature should not in any way weaken the security of its augmented scheme.

In a full-scan design, the scan path provides controlled access to the combinational circuit of the IP core to carry out the stimulus-response tests. The stimuli (test vectors) are serially applied through the scan-in, S_{in} pin and the responses (output vectors) are collected serially from the scan-out, S_{out} pin. Assume a scan chain of length N . A permutation π is defined as a one-to-one mapping of a set of scan cells, $R = \{r_i\}$ to a set of positions, $P = \{p_j\}$ such that the j -th bit of the test vector is loaded into the i -th scan cell once a complete test vector has been shifted into the scan chain, where $i, j = 1, 2, \dots, N$. The proposed DfT watermarking is designed based on the power-driven scan cell ordering algorithm [BONH02]. The aim of this scan cell ordering algorithm is to find an optimal $\pi: R \rightarrow P$ to minimize the power consumption of scan test. By

allocating a scan flip-flop, $r_j \in R$ to the i -th position of the scan chain, the i -th bit of the test vector will be loaded into r_j once a complete test vector has been shifted into the scan chain. Although the test vectors are independently generated and compact to provide the desired fault coverage of the core under test, their bit streams can be ordered to minimize the switching power consumption during scan test. Finding an optimal bijection, $\pi: R \rightarrow P$, where R is the set of scan flip-flops in a scan chain, for minimal test power is an NP-complete problem [DAB98]. Hence, additional watermark constraints can be imposed on a heuristic solver to generate a watermarked solution such that a unique signature will be retrieved from the scan output when a specific test vector is injected into S_{in} , and the probability that the same signature is recovered from a non-watermarked scan chain by coincidence is infinitesimally small.

Unlike the SfT watermarking method proposed in the previous chapter, the procedure presented in this section is applied at the DfT stage after the watermarked FSM has been synthesized and optimized. The scan design to be watermarked has a scan chain of length L and an ordered set, $R = \{r_i\}_{i=1}^L$ consists of the flip-flops in the watermarked FSM and its associated circuits on the same chip. The idea can be extended to designs that contain multiple cores and multiple clusters of scan cells.

5.3.1 Generation of Watermark Constraints

An arbitrary binary input vector, $\chi = \{\chi_i\}_{i=1}^L$, $\chi_i \in \{0,1\}$ is shifted into the initial scan chain to obtain an output vector, $\mathbf{Z} = \{z_i\}_{i=1}^L$, $z_i \in \{0,1\}$. The flip-flops that hold these “0” and “1” bits of $\mathbf{Z}$ are divided into two sets, R_0 and R_1 , respectively. Let L_0 and L_1 be the numbers of “0” and “1” bits in $\mathbf{Z}$, respectively. An L -bit binary vector, $\mathbf{U}$, is formed by concatenating L_0 bits of “0” and L_1 bits of “1”, i.e., $\mathbf{U} = \{u_i\}_{i=1}^L$, $u_i = 0$ for $1 \leq i \leq L_0$ and $u_i = 1$ for $(L_0+1) \leq i \leq L$.

The watermarked constraint consists of a random sequence of L unique integers between 1 and L , i.e., $\omega = \{\omega_i\}_{i=1}^L$, $\omega_i \in \mathbb{Z}^+$, $\omega_i = [1, L]$ and $\omega_i \neq \omega_j \forall i \neq j$. It is generated by the PNG referred in Section 5.2.2. To avoid revealing the watermark

locality, ω is further deranged using a PKC [MEN96] to $\hat{\omega} = \text{PKC}(K_e, \omega)$ with the private key, K_e of the scan chain developer. The vector, $\mathbf{U}$ is then permuted by $\hat{\omega}$ to obtain $\hat{\mathbf{U}} = \{\hat{u}_i\}_{i=1}^L$, where $\hat{u}_i = u_j \forall i, j = 1, 2, \dots, L$ and $j = \hat{\omega}_i$.

5.3.2 Construction of Weighted Connected Graph

The scan-in (scan-out) power for a given test vector has been shown to exhibit a strong correlation with the number of transitions in the test (response) vectors and the intrinsic positions of these transitions relative to the order in which the bits are shifted into and out of the scan chain [BONH02]. In [BONH02], the number of weighted transition in a test or response vector is defined as the summation of the intrinsic positions of its transitions. The scan power dissipated by a complete test sequence is estimated by summing the weighted transitions in all the test and response vectors, and L rippling transitions for every bit difference between the first bit of the test vector and the last bit of the previous response vector. A weighted connected graph, $G(V, E)$ is constructed to model this estimated scan power, where each node represents a distinct scan flip-flop, $r_i \in R$. An edge $E(r_i, r_j)$ is drawn between every pair of nodes, r_i and $r_j \forall r_i, r_j \in R$ and $r_i \neq r_j$. It has a weight equal to the total number of bit differences between r_i and r_j for the complete test sequence. As the weight of $E(r_i, r_j)$ is related to the number of transitions generated by making r_i and r_j adjacent in the scan chain, minimization of the total weighted transitions can be recast to a problem of finding a minimum cost Hamiltonian cycle in G , where the cost of a cycle is obtained by summing the total edge weights of the cycle. An optimal solution is found by using the same NN greedy algorithm [BONH02] as in Chapter 4 with the additional constraint imposed by $\hat{\mathbf{U}}$ on the derangement of the scan chain, R to the watermarked scan chain, $\hat{R}$.

5.3.3 Watermark Insertion

Let $\hat{r}_i = r_j$ denote the allocation of a flip-flop, $r_j \in R$ to the i -th position of the watermarked scan chain. The watermark insertion procedure is summarized by the pseudo code in Figure 5.3. To minimize the spurious transitions between two end

flip-flops, the first watermarked flip-flop, $\hat{r}_1$ will be allocated from a flip-flop in R_ϕ with the smallest maximum edge value, where $\phi = \hat{u}_k \in \{0, 1\}$ and $k = \omega_1$, as indicated in lines 9 and 10. The maximum edge value of a flip-flop, r_i is determined by $E_{\max}(r_i) = \max_{j \neq i} \{E(r_i, r_j)\}$ from G . Then, the watermarked flip-flop, $\hat{r}_1$ is removed from R_ϕ as indicated in line 11. For subsequent watermarked flip-flop positions, $i = 2, 3, \dots, L$, the watermarked flip-flop, $\hat{r}_i$ is selected from R_ϕ ($\phi = \hat{u}_k$ and $k = \omega_i$), that has the least edge cost to the watermarked flip-flop, $\hat{r}_{i-1}$ in the graph G , as indicated in lines 13 and 14. In line 15, the selected flip-flop $\hat{r}_i$ is then removed from the set R_ϕ . This process is repeated until every scan flip-flop in R has been allocated to a unique position in the scan chain.

```

1: watermark_insertion_DFT ( $G, L, \chi, R, K_e$ ) {
2:    $\omega = \{\omega_i\}_{i=1}^L = \text{PNG}(K_e, L)$ ,  $\omega_i = [1, L]$ ;
3:   Obtain  $Z = \{z_i\}_{i=1}^L$  by applying  $\chi = \{\chi_i\}_{i=1}^L$  onto  $R$ ;
4:    $R_0 = \{r_i \in R \mid z_i = 0 \quad \forall 1 \leq i \leq L\}$ ,  $L_0 = |R_0|$ ;
5:    $R_1 = \{r_i \in R \mid z_i = 1 \quad \forall 1 \leq i \leq L\}$ ,  $L_1 = |R_1|$ ;
6:    $U = \{u_i \in U \mid u_i = 0 \quad \forall 1 \leq i \leq L_0 \text{ and } u_i = 1 \quad \forall (L_0 + 1) \leq i \leq L\}$ ;
7:    $\hat{\omega} = \text{PKC}(K_e, \omega) = \{\hat{\omega}_i\}_{i=1}^L$ ,  $\hat{\omega}_i = \omega_j$ ,  $1 \leq j \leq L$ ;
8:    $\hat{U} = \{\hat{u}_i \in \hat{U} \mid \hat{u}_i = u_j \quad \forall i = 1, 2, \dots, L, j = \hat{\omega}_i\}_{i=1}^L$ ;
9:    $\phi = \hat{u}_k$  and  $k = \omega_1$ ;
10:   $\hat{r}_1 = \min(E_{\max}(r_i)) \quad \forall r_i \in R_\phi$ ;
11:   $R_\phi = R_\phi - \hat{r}_1$ ;
12:  for ( $i = 2$  to  $N$ ) {
13:     $\phi = \hat{u}_k$  and  $k = \omega_i$ ;
14:     $\hat{r}_i = \min(E(\hat{r}_{i-1}, r_j)) \quad \forall r_j \in R_\phi$ ;
15:     $R_\phi = R_\phi - \hat{r}_i$ ;
16:  }
17:  return  $\hat{R}$ ;
18: }
```

Figure 5.3 Watermark insertion by power-driven scan chain ordering algorithm.

5.3.4 Verification of Authorship

Let $\hat{R} = \pi(R) = \{\hat{r}_i\}_{i=1}^L$ be an ordered set of flip-flops in the watermarked scan chain. For proper test, the test and response vectors are reordered accordingly by the mapping π . To avoid revealing π during watermark detection, only $\hat{\chi} = \pi(\chi)$ and $\hat{\omega}$ are provided to the verifier for watermark detection. By injecting $\hat{\chi}$ onto the watermarked scan chain in the test mode, a response vector, $\tilde{Z}$ is obtained. Let $\tilde{L}_0$ and $\tilde{L}_1$ be the numbers of “0” and “1” bits of $\tilde{Z}$, respectively. An L -bit binary vector with $\tilde{L}_0$ “0”s followed by $\tilde{L}_1$ “1”s is created and reordered according to $\hat{\omega}$ to obtain a random binary sequence, $\tilde{U}$. $\hat{\omega}$ is decrypted by the verifier using the public key, K_d of the scan chain developer to recover the random integer sequence, $\omega = \text{PKC}(K_d, \hat{\omega})$. ω is used to permute $\tilde{Z} = \{\tilde{z}_i\}_{i=1}^L$ to obtain a binary sequence $V = \{v_i\}_{i=1}^L$ such that the ω_i -th ($\omega_i \in \omega$) element of V is equal to the i -th element of $\tilde{Z}$, i.e., $v_k = \tilde{z}_i$, where $k = \omega_i \forall i = [1, L]$. If V matches $\tilde{U}$, the authorship of the scan chain is verified as ω can only be generated with the secret key of the IP owner.

Since the input, output and state variable registers of the watermarked FSM, $\hat{M}$ in the scan chain have been deranged by the bijection π , some derangement are necessary on $\hat{\mathbf{X}}$ and B to verify $\hat{M}$. The binary bits in $\hat{X}_i$ are reordered according to π for each $i = 1, 2, \dots, N$. The preamble sequences to bring $\hat{M}$ to $\hat{s}_1$ are also reordered accordingly. For each $b_l = (i-1)k + j \in B$, $l = 1, 2, \dots, m$, its value will be replaced by $b_l = (i-1)k + \pi(j)$, where $\pi(j)$ denotes the watermarked scan flip-flop position that holds the FSM output symbol $\hat{y}_{i,j} \in \hat{\mathbf{Y}}$. By applying the deranged preamble sequences and $\hat{\mathbf{X}}$ as consecutive test vectors during the test mode, $\hat{M}$ can be authenticated by the same procedure as in Section 5.2.4.

5.3.5 A Simple Example

Consider the scan chain in Figure 5.4 as an example for watermarking. The original scan chain order generated by the DfT tool is $R = “r_1 r_2 r_3 r_4 r_5”$. X_i and Z_i on the left

hand side denote the i -th test and response vectors, respectively.

Suppose $\mathbf{Z} = "01110"$ is obtained when $\chi = "00110"$ is loaded into the scan chain. From $\mathbf{Z}$ and $\mathbf{R}$, $R_0 = \{r_1, r_5\}$, $R_1 = \{r_2, r_3, r_4\}$ and $\mathbf{U} = "00111"$. Assume that $\omega = \{2, 3, 5, 4, 1\}$ and $\hat{\omega} = \text{PKC}(K_e, \omega) = \{4, 3, 1, 2, 5\}$. According to $\mathbf{U}$ and $\hat{\omega}$, $\hat{\mathbf{U}} = "11001"$.

Figure 5.4 Example for scan chain watermarking.

A weighted graph is constructed in Figure 5.5(a) based on the complete test sequence on the left hand side of Figure 5.4. The vertices corresponding to the flip-flops in R_0 and R_1 are marked with dot lines and continuous lines, respectively. Since $\omega_1 = 2$ and $\hat{u}_2 = 1$, the flip-flop allocated to $\hat{r}_1$ must come from R_1 . As r_2, r_3 and r_4 all have the same maximum edge value of 10, r_2 is arbitrarily allocated to $\hat{r}_1$ and removed from R_1 . As $\omega_2 = 3$ and $\hat{u}_3 = 0$, r_1 and r_5 from R_0 will compete for $\hat{r}_2$. From Figure 5.5(b), $E(r_1, r_2) = 5$ is the least cost edge. So $\hat{r}_2 = r_1$ and r_1 is removed from R_0 . Similarly, $\hat{r}_3$ is selected from $R_1 = \{r_3, r_4\}$ since $\omega_3 = 5$ and $\hat{u}_5 = 1$. $\hat{r}_3 = r_3$ because $E(r_1, r_3) < E(r_1, r_4)$. The process is repeated until all flip-flops are allocated. The ordering of the watermarked scan chain is $\hat{R} = \pi(R) = r_2 r_1 r_3 r_5 r_4$, as shown in Figure 5.4. When the vector $\hat{\chi} = \pi(\chi) = "00101"$ is loaded into $\hat{R}$ in the test mode, $\tilde{Z} = "10101"$ is obtained if the watermarked scan chain has not been tampered. Based on $\tilde{Z}$ and $\hat{\omega}$, the string "00111" is derived and permuted to obtain $\hat{\mathbf{U}} = "11001"$. $\hat{\omega}$ is then

decrypted with the public key K_d of the scan chain developer to obtain $\omega = \{2, 3, 5, 4, 1\}$. The scan output $\tilde{Z}$ is then permuted according to ω to obtain $V = "11001" = \tilde{U}$, hence the watermark existence is proved.

Figure 5.5 Nearest neighbor search for watermark insertion.

5.4 Watermark Resilience Analysis

The watermarked scan chain provides auxiliary fragile watermarking function to alert for the misappropriation of the IP under test and increases the effort level required to successfully forge a testable IP without being detected. The robustness of the authorship proof lies mainly on the watermarked FSM. Hence, the emphasis of attack resilience analysis will be placed on the FSM watermarking.

5.4.1 Authorship Credibility

The credibility of the authorship proof can be evaluated by the probability that an unintended watermark is detected in a design [TOR00]. Suppose that an arbitrary input sequence exits to excite N' ($N' \geq N$) consecutive transitions through the reachable states of an FSM with k output variables. The output sequence of length N' (each output alphabet has k binary bits) will be one of $2^{k \times N'}$ possible solutions. The odds that the output sequence contains the identical watermark bits at the positions specified by the author's signature are:

$$P_c = \frac{2^{k \times N' - m}}{2^{k \times N'}} = \frac{1}{2^m} \quad (5.2)$$

A longer watermark has a lower probability of coincidence. As m increases, more new

transitions may have to be added. The beauty of the proposed method is the input sequence length, N can increase to mitigate the overhead increment without compromising the authorship credibility.

For the scan chain watermarking, the probability that a non-watermark scan chain outputs the same permutation of “0” and “1” bits under the specific input vector $\hat{\chi}$ is given by:

$$P_c = \frac{P_{L_0}^{L_0} \cdot P_{L_1}^{L_1}}{P_L^L} = \frac{L_0! L_1!}{L!} \quad (5.3)$$

The false positive rate, which is the probability that the watermark is detected in the output sequence under a different random input sequence, can be estimated statistically. If there are $N_C(\tau)$ output sequences detected with at least τ fraction of matched watermark bits when N_T random input sequences are applied, then the false positive rate is determined as:

$$P_\lambda(\tau) = \frac{N_C(\tau)}{N_T} \quad (5.4)$$

where $0 \leq \tau \leq 1$. As discussed in the Chapter 4, $P_\lambda(\tau = 1)$ denotes the false positive as all bits extracted from the specific positions by the detector need to be matched exactly with the watermark bits. P_λ increases when τ reduces and a threshold of discrimination can be determined empirically. The threshold for τ , which is usually smaller than 1, means with certain degree of confidence, the authenticity of the design can be assured given only a fraction of the watermark bits detected. A suitable error correction scheme can also be considered based on P_λ to correct the partially corrupted output subsequence due to tampering.

P_c and P_λ are important to repudiate the denial of authorship. To show that the output sequences excited by the verification input cannot be obtained by trial-and-error to match the watermark, the claimant needs only to demonstrate that the watermark and the watermarked positions in the output sequence are uniquely generated with a cryptographic one-way function using a secret key in his/her possession, provided that P_c is very low and P_λ is low enough for a sufficiently large number of random tests.

5.4.2 Resilience Analysis

The following conceivable attacks on watermarking of sequential circuit designs are analyzed with Alice as the IP owner and Bob as the attacker who attempts to tamper an illegally acquired copy of Alice's watermarked IP.

5.4.2.1 Combinational Logic Re-synthesis

Bob may use various logic optimization tools [SEN92] to re-synthesize the combinational logic of the watermarked FSM. Such combinational logic re-synthesis operation maintains the inputs/outputs behaviors of the flip-flops in the design and has no effect on the STG structure. Therefore, the watermark embedded on the STG is robust against attack by combinational logic re-synthesis.

5.4.2.2 Circuit Retiming

Bob may apply retiming transformation [RAN98] to move the latches across the combinational logic blocks of Alice's watermarked FSM without changing the design functionality. Retiming can change the STG structure. Such transformation can be divided into three cases for analysis. (1) Splitting one state into two one-step equivalent states. (2) Merging two one-step equivalent states into one state. (3) Switching between two states that are one-step equivalent. Two states s_i and s_j are said to be one-step equivalent if and only if the two states have the same outputs and the same next state under the same input excitation.

The consequence of splitting, merging or switching transformation on the outputs retrieved in the watermark detection process can be analyzed with a simple STG before and after retiming in Figure 5.6. Assume that states s_{31} and s_{32} are two one-step equivalent states and transitions (s_1, s_{31}, x_t, y_t) and $(s_{31}, s_5, x_{t+1}, y_{t+1})$ are traversed in the watermarking process. Upon retiming, states s_{31} and s_{32} are merged into state s_3 . When the sequence $\hat{X}$ is applied onto the retimed FSM, transitions (s_1, s_3, x_t, y_t) and $(s_3, s_5, x_{t+1}, y_{t+1})$ are traversed, the same outputs as Alice's watermarked FSM are generated from these two steps. Similarly, splitting or switching operations on the watermarked FSM will not prevent the detection of Alice's watermark. Alice's

watermark will not be removed as a state can only be substituted by the state with the same behaviors in retiming transformation.

Figure 5.6 Retiming of the FSM.

5.4.2.3 State Recoding (or Assignment)

Bob may recode the states of Alice's watermarked FSM to remove her watermark. State assignment changes the mnemonic representations of the states in Q . It has no effect on the functional specification of the FSM [CHE95]. As the watermark is embedded in the state transitions rather than the states, Alice's watermark will survive the state recoding attack.

5.4.2.4 Combinational and Sequential Redundancy Removal

When a redundant fault is identified in a sequential circuit, the part of logic can be deleted to simulate the effect of the fault. Bob can remove the combinational logic that is not necessary for the correct behavior of the circuit. This attack has similar effect as the combinational re-synthesis attack as far as the sequential behavior is concerned. So it will not affect the embedded watermark.

Elimination of sequential redundancy may change δ and λ while maintaining the I/O behaviors of the circuit. The sequential redundancies can be categorized into sequentially non-excitable (SNE) and non-distinguishable (ND) faults [CHO93]. An SNE fault is a fault that cannot be excited from any reachable state [CHO93]. As an SNE fault does not affect the reachable part of the STG, removal of SNE faults maintains the integrity of the reachability information. In the proposed watermarking

scheme, all states traversed by $\hat{X}$ are reachable as long as the starting state, $\hat{s}_1$ is selected as a reachable state after the reset state, s_0 . This can be easily guaranteed. As all IOs on the edges of these reachable states are not changed, Alice's watermark can still be detected upon the removal of SNE faults.

Although an ND fault does not affect the I/O behavior, it may change the reachable part of the STG. An ND fault can be identified by verifying the equivalence between the watermarked circuit and the circuit obtained by forcing one node in the circuit to a constant value [CHO93]. If they are equivalent, then a stuck-at fault at that node is non-detectable and some redundancy can be removed. The FSM watermark may be partially erased if the ND faults are detected around the circuit corresponding to the added transitions in the watermarking process. However, this attack is expensive since it requires for each node, a computation of equivalence between two possibly large sequential circuits. This equivalence is obtained by computing the product machine and its set of reachable states. Even with the use of implicit STG traversal techniques, the applicability of this type of sequential redundancy removal is restricted to small circuits. An ND fault can be excited, but none of the excitation vectors can be extended to a test as its effect can never be observed from any primary output. Hence, the auxiliary scan chain watermark will not be affected, even if the ND faults may corrupt the FSM watermark.

5.4.2.5 State Reduction

Bob may perform a state reduction on Alice's watermarked FSM based on the identification of sets of compatible states (compatibles) [CHO93, KYU94]. A set of states is compatible if and only if for each input sequence, there is a corresponding output sequence which can be produced by each state in the compatible. All outputs in the transitions are preserved in the reduced FSM even if the states have been substituted by their compatibles. As the watermark is embedded in the transitions instead of the states, the proposed FSM watermarking will survive the state reduction operation. However, a watermark embedded on the states of the STG, as in the scheme of [OLI01], is vulnerable to the state reduction operation.

5.4.2.6 Transition Elimination

Bob may try to eliminate some transitions in $\hat{I}$. In the proposed watermarking scheme, the existing transitions and the added transitions are indistinguishably utilized for watermarking. There are few added transitions and they are randomly interleaved in the watermarked transition sequence, $\hat{I}$. There is no easy means to eliminate these transitions from the circuit netlist without modifying the correct behaviors of the FSM. Besides, the extraction of the STG from the circuit netlist is an NP-complete problem. Thus, there is no incentive for Bob to do so as the time and effort required for a successful attack is almost as good as redesigning the IP function from scratch.

5.4.2.7 Removal of Circuitries with Pseudo Inputs

The pseudo inputs, if any, are documented as part of the test or primary inputs in the distributed watermarked IP. Due to their random logic assignments, and the high number of *don't cares* they introduced, they are well camouflaged after the logic optimization process. Even if Bob knows about the addition of some pseudo inputs, removal of the circuitries connected to these pseudo inputs will cause malfunction to the watermarked FSM. The conflicts arise because the unspecified transitions created by the pseudo inputs can have different outputs or destination states under the same input combinations as the existing transitions upon the removal of the pseudo inputs. For example, in Figure 5.2(c), when the pseudo input variable is eliminated by the removal of some subcircuits, there will be two transitions from state s_1 with I/O = 10/11 and 10/01, respectively to state s_3 . This is obviously an output conflict, hence such attack is not sustainable.

5.4.2.8 Ghost Searching

Without tampering Alice's design, Bob may claim his ownership of Alice's FSM by specifying some bits in the output sequence generated by his own selected input sequence to make up his watermark. However, it is computationally infeasible for Bob to reverse the PNG to prove that the positions of these extracted bits are cryptographically related to his signature. Alternatively, he can generate a group of integers with his key using a one-way function and then select the bits from these

positions to extract his watermark. Again, it will be computationally infeasible for him to show that the watermark is cryptographically associated with a meaningful ownership message. It is also computationally impractical for Bob to enumerate different sequences of input combinations to match his own watermark to the extracted output sequence of Alice's FSM in his chosen bit positions. The number of trials grows exponentially with the size of the FSM. Depending on Bob's selected bit positions, there is no guarantee that such an input sequence can be found even after trying all possible input sequences.

5.4.2.9 Addition of Watermark

Bob may embed his own watermark into Alice's watermarked design to claim his ownership, if he has the necessary tools and knowledge of the watermarking process. He would have to first recover the STG from the netlist, which is computationally impractical for large designs [OLI99]. Additionally, Bob needs to repeat the entire watermarking and optimization process to ensure that the overhead is acceptable. This problem can be solved by involving a secure third party [ABD05c], e.g., a legal firm or a watermarking governing body. In this case, Alice will generate a time-stamped authenticated signature, and keep it at an authorized legal firm. This firm will keep a record of such signatures and the date it was generated, which can be used in front of court to show the exact time the watermark was generated and embedded in any future dispute. The overall IP watermarking framework is depicted in Figure 5.7.

Figure 5.7 Watermarking with third party keeping a time-stamped signature.

5.5 Experimental Results

The experimentation is performed on the circuits, which are described in KISS2¹⁴ format, from the IWLS93 benchmark set and some FSM designs from the ISCAS89 benchmark set¹⁵. The FSM watermarking scheme is implemented using the C++ language. Watermarks of 64-bit and 128-bit were embedded into each FSM design. Using the SIS [SEN92] tool, state minimization and state assignment are carried out on the original and watermarked designs. The optimized FSM designs are synthesized using the algebraic script from SIS and technology mapped to the Mississippi State University standard cell library. For the watermarking at the DfT stage, the DfT tool, DFTadvisor and the ATPG tool, FastScan by Mentor Graphics are used to generate the required output vector and test patterns for each design. All experiments were run on a 750-MHz Sun UltraSPARC-III with Solaris operating system and 2 GB of memory.

¹⁴ KISS2 format is a standard FSM format that is used by many tools. It is accepted for the input and output in sequential synthesis tool SIS and can also be embedded in other common format file, such as BLIF specification [SEN92]. It describes the STG explicitly and can be easily dealt with using C++ language.

¹⁵ IWLS93 benchmark set [MCE93] includes many sequential designs in different sizes and they have been used in many other references on FSM watermarking [ABD05b, TOR00]. Designs in ISCAS'89 benchmark are used in [OLI01]. For the purpose of comparison, some designs from ISCAS89 are experimented.

Table 5.2 Statistics for ISCAS89 benchmark designs.

Circuit	$ Q $	n	k	A	D	m	n_p	N	n_a	A_{wm}	D_{wm}	$\Delta A(\%)$	$\Delta D(\%)$
s27	6	4	1	824	7	64	1	600	2	856	7.4	3.88	5.71
						128	1	300	2	856	7.4	3.88	5.71
s208	18	11	2	1912	13	64	1	100	4	1984	11.8	3.77	-9.23
						128	1	100	4	1992	11.8	4.18	-9.23
s420	18	19	2	2112	13	64	1	400	1	1928	11.6	-8.71	-10.77
						128	1	200	2	2056	11.8	-2.65	-9.23
s499	22	1	22	2480	13.6	64	2	100	3	3328	13	34.19	-4.41
						128	2	200	4	3608	16.4	45.48	20.59
s386	13	7	7	2512	13	64	1	100	5	3032	14.0	20.70	7.69
						128	1	600	5	3080	14.4	22.61	10.77
s832	25	18	19	5144	19.8	64	1	40	4	5656	20.2	9.95	2.02
						128	1	200	6	6080	19.0	18.2	-4.04
s510	47	19	7	5528	18.4	64	1	100	4	5888	17.8	6.51	-3.26
						128	1	200	7	6272	17.8	13.46	-3.26
s820	25	18	19	6112	20.6	64	1	200	3	5504	19.6	-9.95	-4.85
						128	1	300	4	6344	20.0	3.80	-2.91
s1488	48	8	19	10544	23	64	1	200	2	10864	24.2	3.03	5.22
						128	1	300	4	11168	23.4	5.92	1.74
s1494	48	8	19	10904	24.8	64	1	300	4	10856	24.2	-0.44	-2.42
						128	1	300	5	11808	24.4	8.29	-1.61

In Tables 5.2 and 5.3, the columns “ $|Q|$ ”, “ n ” and “ k ” are the numbers of states, input variables and output variables of each FSM design, respectively. “A” and “D” are the area and delay of the originally optimized design, respectively, as reported by SIS. The column “ m ” is the watermark length. “ n_p ” is the number of pseudo input variables introduced in the watermarked design. “ N ” denotes the length of verification code sequence. “ n_a ” denotes the number of new transitions added onto the watermarked STG. “ A_{wm} ” and “ D_{wm} ” denote the area and delay of the watermarked design, respectively. ΔA and ΔD are the percentage area and delay overheads, respectively.

Each design is watermarked with the first 64 and 128 bits of the SHA-1 hash values of the ownership information. Different lengths of verification code sequence, N have been experimented with $N_1 = 80$ and $N_{max} = 600$. $\delta_i = 20$ when $N_i < 100$ and $\delta_i = 100$ when $N_i \geq 100$. Typically, σ_i/A converges to $\epsilon = 0.05$ in less than five trials. The value of “ N ” indicated is the one that produces the least area overhead watermarked FSM design. “ n_p ” is the number of pseudo input variables introduced in the watermarked design. As shown by the data in column “ n_p ”, for the two designs, “s499” and “dk27”

with only one input variable, at most two pseudo input variables are introduced in the watermarking process. For the three incompletely specified FSMs, “kirkman”, “ex1” and “sand”, either none or only one pseudo input variable is needed. All other designs need at most one pseudo input variable. “ n_a ” denotes the number of new transitions added onto the watermarked STG.

For the design overhead denoted by ΔA and ΔD , a negative percentage implies that watermarking has actually improved the performance. In general, more new transitions have been added onto the designs with 128-bit watermark than with 64-bit watermark. The performance overheads reduce as the size of the FSM increases. For the six larger designs (twelve watermarked designs), the average area is increased by 4.23% but the average timing is actually improved by 0.52%. It is conjecture that the performance overheads are negligible for the watermarked FSMs of actual industrial designs, which can have many more states and input and output variables than those simulated.

Table 5.3 Statistics for IWLS93 benchmark designs.

Circuit	$ Q $	n	k	A	D	m	n_p	N	n_a	A_{wm}	D_{wm}	$\Delta A(\%)$	$\Delta D(\%)$
lion	4	2	1	440	5.6	64	1	500	2	600	6.2	36.36	10.71
						128	1	600	2	600	6.2	36.36	10.71
mc	4	3	5	608	7.8	64	1	400	2	832	7.8	36.84	0.00
						128	1	500	5	1032	8.6	69.74	10.26
dk27	7	1	2	624	6.2	64	1	500	4	832	7.6	33.3	22.58
						128	1	500	9	920	7.4	47.44	19.35
bbara	10	4	2	1112	10.4	64	1	600	3	1208	9.2	8.63	-11.54
						128	1	600	5	1304	10.2	17.27	-1.92
dk15	8	3	5	1440	10	64	1	500	1	1448	10.4	0.56	4.00
						128	1	500	1	1568	11.2	8.89	12.00
ex4	14	6	9	1584	10.8	64	0	100	11	2072	11.2	30.81	3.70
						128	0	600	9	1944	11.0	22.73	1.85
opus	10	5	6	1768	10.8	64	1	400	4	1944	12.0	9.95	11.1
						128	1	400	9	2152	12.8	21.7	18.5
sse	16	7	7	2560	12.2	64	1	400	3	2976	13.6	16.3	11.48
						128	1	500	6	2896	14.2	13.13	16.4
kirkman	16	12	6	3632	16.8	64	1	600	9	4096	18.6	12.78	10.71
						128	1	60	15	4384	17.0	20.70	1.19
ex1	20	9	19	4360	16	64	0	300	5	5176	18.6	18.72	16.25
						128	0	500	10	5776	19.2	32.48	20
s1	20	8	6	5112	16.6	64	1	200	4	4728	16.8	-7.51	1.20
						128	1	500	7	5592	20.2	9.39	21.69
tbk	32	6	3	5352	19.6	64	1	400	2	5224	20.2	-2.39	3.06
						128	1	200	6	5352	19.4	0.00	-0.01
styr	30	9	10	8480	22.2	64	1	60	5	9032	22.2	6.51	0.00
						128	1	200	5	8656	19.6	2.08	-11.7
sand	32	11	9	9008	23	64	0	400	4	9184	21.4	1.95	-6.96
						128	0	400	5	9392	24.8	4.26	7.83
planet	48	7	19	9784	21	64	1	100	5	9920	21.2	1.39	0.95
						128	1	100	6	10968	20.4	12.1	-2.86
ram_test	72	16	24	9864	24	64	1	600	3	9152	20	-7.22	-16.67
						128	1	600	4	9152	21.4	-7.22	-10.83
scf	121	27	56	12640	23.8	64	1	400	13	14080	26	11.39	9.24
						128	1	400	15	14824	26.2	17.28	10.08

According to (5.2), the probabilities of coincidence, $P_c = 5.42 \times 10^{-20}$ and 2.49×10^{-39} for $m = 64$ and 128, respectively. The false positive rate, P_λ is determined empirically by applying 1000 randomly generated input code sequences of length N onto each watermarked FSM at the watermarked starting state. None of the output sequence was detected with a perfectly matched watermark for each watermarked FSM, i.e., $P_\lambda(\tau = 1) = 0$ for all watermarked designs. It is thought to be reasonable that a sufficiently low probability is adequate to prove the authorship and make the denial attacks

unsustainable. Hence, the watermark correlation is reduced from 100% to 75% match. It was found that for $\tau = 0.75$, $P_\lambda = 0$ for all watermarked designs. When τ is reduced to 0.7, 15 out of 54 watermarked designs have $P_\lambda > 0$. Based on these results, it is reasonable to assume that when more than three quarters of watermark bits are matched, the authorship proof is still veracious.

The SIS tool and the same technology library are used to synthesize the designs watermarked by the method in [ABD05b]. Their areas and delays are compared with those of the proposed FSM watermarking method, as shown in Table 5.4. ΔA and ΔD are the percentage reductions of area and delay, respectively, of the proposed scheme over those of [ABD05b]. It is evident that most designs watermarked by the proposed method have lower area and timing overheads.

The proposed watermarking method is compared with Oliveira's [OLI01] FSM watermarking scheme in Table 5.5. As the same synthesis tool and technology library were used, the area and delay results are excerpted from [OLI01] for those circuits provided with both BLIF and KISS2 formats in the benchmark suite and have comparable literal counts in their original designs before watermarking. In Table 5.5, the area is measured in terms of the number of the literals to be consistent with [OLI01]. All designs watermarked by the proposed method have consistently lower area and timing overhead than [OLI01]. ΔA and ΔD are the percentage reductions of area and delay, respectively, of the proposed scheme over those of [OLI01]. It should also be noted that the watermarking method of [OLI01] does not survive the state reduction operation.

Tabele 5.4 Comparison with FSM watermarking method in [ABD05b].

Circuit	m	Area			Delay		
		[ABD05b]	proposed	ΔA (%)	[ABD05b]	proposed	ΔD (%)
s27	64	1064	856	19.6	7.8	7.4	5.1
	128	1064	856	19.6	7.8	7.4	5.1
s208	64	3680	1984	46.1	15.8	11.8	25.3
	128	5248	1992	62.0	19.8	11.8	40.4
s420	64	4376	1928	55.9	18.4	11.6	37.0
	128	4704	2056	56.3	17.20	11.8	31.4
s386	64	4312	3032	29.7	16.0	14.0	12.5
	128	4592	3080	32.9	16.2	14.4	11.1
s832	64	6256	5656	9.6	21.0	20.2	3.8
	128	6904	6080	11.9	21.6	19.0	12.0
s510	64	7608	5888	22.6	21.2	17.8	16.0
	128	7816	6272	19.8	19.4	17.8	8.2
s820	64	6208	5504	11.3	20.6	19.6	4.9
	128	7352	6344	47.0	21.4	20.0	6.5
s1488	64	12032	10864	9.7	23.2	24.2	-4.3
	128	14120	11168	20.9	25.6	23.4	8.6
s1494	64	12488	10856	13.1	29.0	24.2	16.6
	128	12816	11808	7.9	25.2	24.4	3.2
lion	64	608	600	1.3	7.2	6.2	13.9
	128	608	600	1.3	7.2	6.2	13.9
dk27	64	1280	832	35	9.6	7.6	20.8
	128	1424	920	35.4	11.2	7.4	33.9
bbara	64	2512	1208	51.9	13.2	9.2	30.3
	128	2512	1304	48.1	13.2	10.2	22.7
dk15	64	2104	1448	31.2	11.6	10.4	10.3
	128	2504	1568	37.4	13.6	11.2	17.6
ex4	64	2104	2072	1.5	11.4	11.2	1.8
	128	3008	1944	35.4	13.0	11.0	15.4
sse	64	4024	2976	26.0	15.8	13.6	13.9
	128	4496	2896	35.6	15.2	14.2	6.6
ex1	64	5776	5176	10.4	17.8	18.6	-4.5
	128	6056	5776	4.6	18.4	19.2	-4.3
s1	64	5712	4728	17.2	17.8	16.8	5.6
	128	7376	5592	24.2	19.6	20.2	-3.1
sand	64	9840	9184	6.7	23.2	21.4	7.8
	128	10952	9392	14.2	21.6	24.8	-14.8
planet	64	11608	9920	14.5	22.6	21.2	6.2
	128	11968	10968	8.4	25.4	20.4	19.7
styr	64	10056	9032	10.2	21.8	22.2	1.83
	128	10096	8656	14.3	26.2	19.6	25.2
scf	64	13752	14080	-2.4	24.2	26	-7.4
	128	14584	14824	-1.6	29.2	26.2	10.3

The FSM watermarking method is also compared with the FSM watermarking method [TOR00] in Table 5.6. For consistency, the designs are watermarked with the same length of watermark as [TOR00] and synthesized using the same MSU (Mississippi State University) script from SIS. The watermark length “ m ” used by [TOR00] is design dependent and can be determined by the product of the number of output variables, $|\Delta|$ of the watermarked design and the minimum number of watermarked transitions, $n_{\min}$ needed to satisfy the required P_c (P_u in [TOR00]). ΔA is the percentage area reduction of the proposed scheme over that of [TOR00]. It is evident that the proposed method incurs lower area overhead than [TOR00] for the same constraint on the watermark robustness.

Table 5.5 Comparison with FSM watermarking method in [OLI01].

Circuit	m	Area			Delay		
		[OLI01]	proposed	ΔA (%)	[OLI01]	proposed	ΔD (%)
s27	64	297	51	82.8	21.8	7.4	66.1
	128	541	51	90.6	25.4	7.4	70.9
s208	64	308	164	46.8	15.6	11.8	24.4
	128	441	170	61.5	19.8	11.8	40.4
s420	64	380	151	60.3	25.2	11.6	54.0
	128	489	162	66.9	25.6	11.8	53.9
s499	64	879	272	69.1	34.6	13	62.4
	128	1230	292	76.3	39.6	16.4	58.6
s386	64	444	248	44.1	17.8	14.0	21.3
	128	644	258	59.9	23.0	14.4	37.4
s832	64	680	486	28.5	22.2	20.2	9.0
	128	804	518	35.6	24.4	19.0	22.1
s510	64	581	512	11.9	20.6	17.8	13.6
	128	688	545	20.8	21.0	17.8	15.2
s820	64	669	463	30.8	26.4	19.6	25.8
	128	814	539	33.8	24.0	20.0	16.7
s1488	64	1318	968	26.6	33.4	24.2	27.5
	128	1495	981	34.4	31.0	23.4	24.5
s1494	64	1329	945	28.9	32.8	24.2	26.2
	128	1547	1050	32.1	31.4	24.4	22.3

Table 5.6 Comparison with FSM watermarking method in [TOR00].

Circuit	m	Area		
		[TOR00]	proposed	ΔA (%)
s27	36	1.53k	0.62k	59.5
bbara	30	2.01k	1.05k	47.8
dk14	35	1.84k	1.74k	5.4
styr	40	10.69k	7.54k	29.5
bbsse	70	2.62k	2.46k	6.1
cse	35	4.08k	3.62k	11.3
sse	21	2.43k	2.34k	3.7
ex1	76	5.55k	4.38k	21.1
ex1	38	5.40k	4.06k	24.8
scf	112	21.02k	14.4k	31.5

Eight larger watermarked FSMs were selected for further watermarking at the DfT process and the results are shown in Table 5.7. In Table 5.7, “ L ” denotes the scan chain length. “ WT_{org} ” and “ WT_{wm} ” denote the numbers of weighted transitions of the optimized scan design of the FSM before and after DfT watermarking, respectively. “ ΔWT ” = $(WT_{wm} - WT_{org}) \times 100 / WT_{org}$ represents the percentage test power overhead of the watermarked design. A negative percentage implies a reduction in the test power consumption. Although the highest overhead is 6%, a 7.24% reduction in test power is observed on average. “ L_0 ” and “ L_1 ” are the number of “0” and “1” bits, respectively in the response vector obtained under the verification vector. “ P_{c1} ” and “ P_{c2} ” are the probabilities of coincidence of the FSM and DfT watermarking, respectively. The probability, “ P_c ” that a non-watermarked design carries both the FSM and scan design authorships simultaneously, is computed as the product of P_{c1} and P_{c2} . As the size of the design increases, the overheads incurred on the FSM and DfT watermarked designs become negligible and the P_c of the hybrid watermarking scheme declines rapidly.

Table 5.7 Scan power overheads and watermark strength of the hybrid watermarking.

Circuit	L	WT_{org}	m	WT_{wm}	ΔWT	L_0	L_1	P_{c1}	P_{c2}	P_c
s510	33	33.1k	64	33.2k	0.11	15	18	5.4E-20	9.6E-10	5.2E-29
			128	25.5k	-23.0	20	13	2.9E-39	1.7E-09	5.1E-48
planet	33	39.8k	64	30.5k	-4.53	19	14	5.4E-20	1.2E-09	6.6E-29
			128	39.4k	-23.4	19	14	2.9E-39	1.2E-09	3.6E-48
s1488	34	46.0k	64	37.8k	-17.8	21	13	5.4E-20	1.1E-09	5.9E-29
			128	43.9k	-4.53	16	18	2.9E-39	4.5E-10	1.3E-48
s1494	34	44.3k	64	44.6k	0.68	21	13	5.4E-20	1.1E-09	5.9E-29
			128	41.9k	-5.50	19	15	2.9E-39	5.4E-10	1.6E-48
s820	43	66.6k	64	57.1k	-14.3	24	19	5.4E-20	1.3E-12	6.8E-32
			128	60.8k	-8.71	18	25	2.9E-39	1.6E-12	4.8E-51
s832	43	88.9k	64	76.6k	-13.9	25	18	5.4E-20	1.6E-12	8.9E-32
			128	94.2k	5.93	17	26	2.9E-39	2.4E-12	7.0E-51
ram_test	48	80.4k	64	80.6k	0.16	21	27	5.4E-20	4.5E-14	2.4E-33
			128	74.4k	-7.55	30	18	2.9E-39	1.4E-13	4.0E-52
scf	91	40.5k	64	42.9k	6.00	40	51	5.4E-20	9.4E-27	5.1E-46
			128	38.3k	-5.59	40	51	2.9E-39	9.4E-27	2.8E-65

5.6 Summary

This chapter presents a new hybrid watermarking scheme that offers a high degree of tamper resistance and provides an easy and truly noninvasive copy detection scheme. The proposed scheme features two new techniques of embedding unique ownership information into sequential circuit design in the frontend and backend processes of VLSI tool flow. The robust FSM watermark is highly resilient to all conceivable watermark removal attacks, and is fortified by an auxiliary fragile DfT watermarking. The four main goals of IP protection [VSI00], namely (1) protection of IPs against unauthorized use, (2) protection of all types of design data used to produce and deliver IPs, (3) detection of IP usage, and (4) traceability of IP usage are fulfilled by integrating the robust and fragile watermarking features of both methods via the reachability information of sequential designs. The watermarked scan design enables the watermark on the FSM to be detected directly and publicly in the field even after the IP is integrated into an SoC and packaged without divulging the design data. The DfT watermarking adds more barriers to the breaching of FSM watermark. The barrier put up by the watermarked scan chain also serves as an intruder-alert to enable the tracking of unauthorized IP distribution. The more robust FSM watermark remains intact even if the scan chain developer's signature is corrupted by the aggressor. The experimental results show that each watermarking mode incurs acceptably low

performance overheads and possesses very low possibility of coincidence and false positive rate. Hence, the combined authorship proof of the hybrid watermarking scheme is even stronger and undeniable.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

IP-based design methodology has become a mainstream strategy for accelerating new product development in the SoC era. The reuse-based design environment makes IP sharing more vulnerable than ever to piracy and illegal distribution. The escalating annual revenue loss in IC business due to IP infringement poses a dire threat to the closing of design productivity gap by reusable IPs. The importance of this problem is marked by the identification of IPP as one of the six key enabling technologies by the VSI Alliance in late 1997. The primary goals of IPP include the protection of IPs against unauthorized use, the protection of design data used to generate and deliver the IPs, and the detection and tracing of the use of IPs.

Detection-based IPP such as watermarking is more efficient and economical than the deterrent approaches like copyright and patent. The work developed in this thesis is dedicated to constraint-based watermarking techniques. The contributions can be summarized into four major domains: logic synthesis watermarking for combinational logic networks, synthesis-for-testability watermarking for field authentication and fraudulence tracking of multiple authorships and IP cores, and hybrid watermarking for protection of sequential circuit designs.

In the proposed watermarking scheme at the logic synthesis level, an originally optimized master design is used to extract the excess bandwidth relative to the original timing specifications. In order to maximally exploit the excess bandwidth to improve the watermarked solution quality, slack sustainability is introduced to determine the headroom of independent closed cone for incremental technology

mapping. The signature is used to select the qualified closed cones in random and depending on their binary values, the designated templates in the selected cones are either retained or extricated upon remapping. A direct signature correlation verification is also made possible by this method, which has successfully avoided the exposure of the grammar used for the generation of watermark constraints. Compared with other peer watermarking techniques at logic synthesis level, the proposed approach incurred the least performance overhead under the same credibility of authorship proof.

The vulnerability due to the reverse engineering required for watermark detection has been alleviated in the above watermarking scheme. Nonetheless, some reverse engineering is still needed to extract the local networks corresponding to the watermark hosts. To overcome the inherent weaknesses of watermark detection in static watermarking, dynamic watermarking technique on the scan chain ordering problem has been explored. The watermark is embedded before synthesis and optimization process by means of synthesis-for-testability. Using this method, the testability features are merged with the synthesis process itself so that the test design becomes an integral part of the functionality of the watermarked IP. This has made attempts to obfuscate the embedded watermark more costly and fragile than when the attacks are mounted on existing DfT-based watermarking methods. The empirical evidence has been presented to show that the watermarked designs incur very low area and timing overheads, and possess very low rate of false positives. The probability of successful random obliteration of watermark is very low.

An identity-based signature scheme has been proposed for the first time to enable multiple IP cores watermarked by different authors in an SoC to be publicly authenticated by the IP buyers and the end users in the field. The application of this cryptographically-proven protocol in the SfT watermarking method makes tracking of fraudulence activities of IP licensees feasible, hence effectively deters illegal IP distribution.

In the SfT watermarking, the test control circuitries can still be identified with the help of advanced tool. To make the watermark harder to remove while still easy to

track, two primary facets of sequential circuit design flow have been explored to develop a hybrid dynamic watermarking scheme. At the higher behavioral synthesis level, watermark is embedded in the STG of the FSM to enable the authorship to be verified directly by running the watermarked FSM with a specific code sequence. At the backend DfT stage, the scan chain that incorporates the watermarked FSM as one of the test kernels is further watermarked with the scan chain developer's signature to provide a second level of protection with a fragile watermarking feature. The dual protection modes at two different abstraction levels make the authorship proof harder to erase and the IP forgery easier to trace. The watermarking scheme has shown to have acceptably low overhead while exhibiting a strong authorship proof.

In summary, the main objective to develop new watermarking schemes for VLSI IP protection with strong authorship proof that are easy to verify and more resilient to attacks has been achieved in this thesis. Three watermarking approaches targeted at different design levels and an identity-based authentication protocol have been proposed. The watermarking processes are transparent to the design tools. They incur negligible performance overhead on the original designs while providing strong authorship proof. The watermarks are easy to verify and resilient to many known attacks. Overall, the results of the original contributions presented in this thesis are better in most aspects of measurements compared with other schemes proposed at the same design abstraction levels.

6.2 Suggestions for Future Work

Based on the work presented in this thesis, several relevant topics and potential direction for further exploration are proposed.

6.2.1 Combination of Static Watermarking with Dynamic Watermarking

The proposed watermarking scheme at the logic synthesis level has provided a relatively direct watermark detection method. However, as a static watermarking, it still relies on reverse engineering to detect embedded watermark and disables field authentication of authorship after the IP core is integrated or the IC is packaged. Dynamic watermarking during DfT stage makes the detection of watermark after

packaging possible, however, it is vulnerable to the removal attacks as the test circuitry can be identified by the tools and then removed. It is promising to protect an IP using the proposed static watermarking and dynamic watermarking simultaneously to combine the merits of both watermarking techniques to make the protected IP strong to resist removal attacks and easily to be authenticated publicly.

6.2.2 Constraining Design Overheads of Watermarking

Insertion of extra design constraints for watermarking introduces design overhead. Such overhead should be kept low to make the watermarking scheme applicable. The overheads due to many existing watermarking techniques can be kept acceptably low while providing strong authorship proof. However, the number of watermarking constraints usually decides the design overhead and the random selection of the watermarking constraints makes the design overhead hard to predict. The design overhead cannot be exploited or absolutely bounded in current watermarking techniques, including those proposed in the thesis. A potential research work is to develop a model that can allow the design overhead of the proposed methods to be reasonably bounded to facilitate the appropriate selection of watermarking constraints. This will avoid a situation where the success or failure of watermarking an IP can only be passively accepted when watermarking is completed.

6.2.3 General Watermarking Scheme on FSM

For any sequential circuit of reasonable complexity, extraction of the STG is practically infeasible, since the STG itself is too large to be represented explicitly. The proposed FSM watermarking and some other schemes rely on the transitions in the STG, which makes the extraction of the STG mandatory. Such watermarking may not be applicable to large FSMs. As watermarking on FSM can protect the design functionality at a higher design level, more research shall be focused on the general FSM watermarking scheme that can be easily carried out on large FSMs without the need of the STG.

6.2.4 Soft IP Protection

Soft IPs allow customer-dependent optimization and are more valuable. However, releasing IP in synthesizable behavioral-level HDL is highly vulnerable to re-synthesis attack. Watermarking soft IPs to provide strong authorship proof and survivability from re-synthesis might be made feasible with documentation and verification assumptions. The abstract interpretation method proposed in [COU04] to watermark software is highly vague. Nonetheless, it could be of significance as the abstract interpretation is an NP-hard problem. The technique can be considered on HDL codes to increase the embedding capacity and reduce the design overhead by integrating it with other watermarking techniques on firm IPs and hard IPs. The combination of hard IP and soft IP at different levels to provide a quality bound on the watermarked solution is a promising direction.

6.2.5 Birthmarking Method for CAD Tools and Algorithms

In principle, CAD tools and algorithms can be watermarked by treating the design tools and algorithms as IPs. In practice, this idea hardly works as the quality of solutions generated by the protected tools and algorithms cannot be assured and it can only be applied to new tool and algorithm. Existing CAD tools and algorithms possess certain unique characteristics which are usually imprinted in the solutions they generated unintentionally. Forensic engineering technique detects potential CAD tool and algorithm piracy by identifying problem-dependent traces of the solutions generated by strategically different tools and algorithms. An alternative idea that can be explored in this research is to intentionally force a trace of the tool and algorithm in the solutions generated by them. This birthmark is to be created locally in the design solution to avoid the design quality from being degraded.

6.2.6 Watermarking Industrial Size Designs

To evaluate and illustrate the new methodologies, extensive experimentations are to be carried out on real VLSI IPs to demonstrate the robustness and performance metrics. Most VLSI design watermarking techniques today have been demonstrated on small designs. Watermarking a real industrial size design is a challenge, especially for the authentication of multi-core Network-on-a-Chip (NoC) where each IP block

has a different ownership. The positive side of this is the amount of data passed over the NoC is large which enables a subliminal channel to be defined for the generation of multiple watermarks for different authorship authentication.

6.2.7 Watermarking Analog Integrated Circuits

In analog circuit designs, reduction of development cycle time is also a challenging target. The increased requirements on performance and functionality cause the design effort in the creation of analog circuits to become extraordinarily high. Making use of IP-reuse can help to reduce the development cycle time tremendously. Some possible reuse methods have been proposed. The protection of the copyrights of the reusable analog IPs is now a concern. It is a promising research direction to look into the challenges of developing watermarking schemes for the copyright protection of analog integrated circuits.

Author's Publication

Journal Papers

- [1] **Aijiao Cui**, Chip-Hong Chang, and Sofiène 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, September 2008, pp. 1565-1570.
- [2] Chip-Hong Chang and **Aijiao Cui**, "Synthesis-for-Testability watermarking for field authentication of VLSI intellectual property," accepted by *IEEE Transactions on Circuits and Systems I: Regular Papers*.
- [3] **Aijiao Cui**, Chip-Hong Chang, Sofiène Tahar, and Amr T. Abdel-Hamid, "A hybrid watermarking scheme for IP protection of sequential circuit designs," submitted to *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* for review as a regular paper.

Conference Papers

- [1] **Aijiao Cui**, and Chip-Hong Chang, "Stego-signature at logic synthesis level for digital design IP protection," in *Proceedings IEEE International Symposium on Circuits and Systems*, Kos, Greece, May 2006, pp. 4611-4614.
- [2] **Aijiao Cui**, and Chip-Hong Chang, "Kernel Extraction for Watermarking Combinational Logic Networks," in *Proceedings IEEE Asia Pacific Conference on Circuits and Systems*, Singapore, December 2006, pp. 1023-1026.
- [3] **Aijiao Cui**, and Chip-Hong Chang, "Watermarking for IP protection through template substitution at logic synthesis level," in *Proceedings IEEE International Symposium on Circuits and Systems*, New Orleans, Louisiana, USA, May 2007, pp. 3687-3690.
- [4] **Aijiao Cui**, and Chip-Hong Chang, "Intellectual property authentication by watermarking scan chain in design-for-testability flow," in *Proceedings IEEE International Symposium on Circuits and Systems*, Seattle, Washington, USA, May 2008, pp. 2645-2648.

- [5] **Aijiao Cui**, and Chip-Hong Chang, "An improved publicly detectable watermarking scheme based on scan chain ordering," in *Proceedings IEEE International Symposium on Circuits and Systems*, Taipei, Taiwan, May 2009, pp. 29-32.

Bibliography

- [AAR97] E. H. L. Aarts, and J. K. Lenstra, *Local Search in Combinatorial Optimization*. London: John Wiley and Sons, 1997.
- [ABD05a] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "A survey on IP watermarking techniques," *Design Automation for Embedded Systems*, vol. 10, Springer Verlag, 2005, pp. 1-17.
- [ABD05b] A. T. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "A public-key watermarking technique for IP designs," in *Proceedings Design, Automation and Test in Europe*, vol. 1, Munich, Germany, March 2005, pp. 330-335.
- [ABD05c] A. T. Abdel-Hamid, "Watermarking techniques for Intellectual Property protection in SoC designs," a thesis in the department of Electrical and Computer Engineering for the Degree of Doctor of Philosophy at Concordia University, Canada, September 2005.
- [ABR90] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. New York: W. H. Freeman & Co., 1990.
- [BEN96] Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Systems Journal*, vol. 35, no. 3&4, 1996, pp. 313-336.
- [BONE96] L. Boney, A. H. Tewfik, and K. N. Hamdy, "Digital watermarks for audio signals," *Proceedings IEEE International Conference Multimedia Computing and System* Hiroshima, Japan, June 1996, pp. 473-480.
- [BONH02] Y. Bonhomme, P. Girard, C. Landrault, and S. Pravossoudovitch, "Power driven chaining of flip-flops in scan architectures," in *Proceedings IEEE International Test Conference*, Washington, USA, 2002, pp. 796-803.
- [BRAS96] J. Brassil, and L. O'Gorman, "Watermarking document images with bounding box expansion," *Information Hiding*, Lecture Notes in Computer Science, vol. 1174, Springer Verlag, 1996, pp. 227-235.
- [BRAU96] G. W. Braudaway, K. A. Magerlein, and F. Mintzer, "Protecting publicly available images with a visible image watermark," *Proceedings SPIE-International Society for Optical Engineering*, vol. 2659, February 1996, pp. 126-133.

-
- [BUR89] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," in *Proceedings Royal Society, Series A*, vol. 426, 1989, pp. 233-271.
 - [BURG99] P. Burggraaf, "1999ITRS: 2000 begins with a revised industry "roadmap"," http://www.solid-state.com/display_article/55597/5/none/none/Feat/1999-ITRS:-2000-begins-with-a-revised-industry-%22Roadmap%22, 1999.
 - [CAS07] E. Castillo, U. Meyer-Baese, A. Garcia, L. Parrilla, and A. Lloris, "IPP@HDL: efficient Intellectual Property protection scheme for IP Cores," *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 15, no. 5, May 2007, pp. 578-591.
 - [CHAK93] S. T. Chakradhar, V. D. Agrawal, and S. G. Rothweiler, "A transitive closure algorithm for test generation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, pp. 1015-28, July 1993.
 - [CHAN05] C. H. Chang, Z. Ye, and M. Zhang, "Fuzzy-ART based adaptive digital watermarking scheme," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 1, January 2005, pp. 65-81.
 - [CHAR98] E. Charbon, "Hierarchical watermarking in IC design," in *Proceedings IEEE Custom Integrated Circuits Conference*, Santa Clara, California, USA, May 1998, pp. 295-298.
 - [CHE95] D. Chen, M. Sarrafzadeh, and G. K. H. Yeap, "State encoding of finite state machines for low power design," in *Proceedings IEEE International Symposium on Circuits and Systems*, Seattle, Washington, USA, April 1995, pp. 2309 - 2312.
 - [CHO93] H. Cho, G. D. Hachtel, and F. Somenzi, "Redundancy identification / removal and test generation for sequential circuits using implicit state enumeration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 7, July 1993, pp. 935-945.
 - [CIR91] M. A. Cirit, "Characterizing a VLSI standard cell library," in *Proceedings IEEE Custom Integrated Circuits Conference*, San Diego, California, USA, May 1991, pp. 25.7/1 - 25.7/4.
 - [CON00] J. Cong, and H. Huang, "Depth optimal incremental mapping for field programmable gate arrays," *Proceedings ACM/IEEE Design*

- Automation Conference*, Los Angeles, California, USA, June 2000, pp. 290-293.
- [COO71] S. A. Cook, "The complexity of theorem proving procedures," in *Proceedings 3rd Annual ACM Symposium on Theory of Computing*, pp. 151-158, 1971.
- [COU04] P. Cousot, and R. Cousot, "An abstract interpretation-based framework for software watermarking," in *Proceedings 31th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Venice, Italy, January 2004, pp. 173-185.
- [COX94] H. Cox, "On synthesizing circuits with implicit testability constraints," in *Proceedings International Test Conference*, Washington D.C., USA, Oct. 1994, pp. 989-998.
- [COXI08] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich and T. Kalker, *Digital Watermarking and Steganography*. (Second Edition) Morgan Kaufmann, 2008.
- [DAB98] V. Dabholkar, S. Chakravarty, I. Pomeranz, and S.M. Reddy, "Techniques for minimizing power dissipation in scan and combinational circuits during test application," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 12, December 1998, pp. 1325-1333.
- [DEV89] S. Devadas, "Optimal layout via Boolean satisfiability," in *IEEE International Conference Computer-Aided Design*, November 1989, pp. 294-297.
- [DOS02] J. A. Dossey, A. D. Otto, L. E. Spence, and C. V. Eynden, *Discrete Mathematics*. Boston: Addison Wesley, 2002.
- [DRE01] N. J. Drew, and M. M. Dillinger, "Evolution toward reconfigurable user equipment," *IEEE Communications Magazine*, February 2001, pp. 158-164.
- [FAN03] Y. C. Fan, and H. W. Tsao, "Watermarking for intellectual property protection," *IEE Electronics Letters*, vol. 39, no.18, September 2003, pp. 1316-1318.
- [FSU] <http://microscopy.fsu.edu/creatures/technical/packaging.html>

-
- [HCH99] H. Chang, L. Todd, A. McNelly, G. Martin, M. Hunt, and Larry Cooke, *Surviving the SOC revolution - A guide to platform-based design*. Springer, 1999.
- [HEM54] E. Hembrooke, "Identification of sound and like signals," US Patent 3,004,104, filed 1954, Issued 1961.
- [HIR98] M. Hirech, J. Beausang, and X. Gu, "A new approach to scan chain reordering using physical design information," in *Proceedings International Test Conference*, Washington D.C., USA, October 1998, pp. 348-355.
- [HON98] I. Hong, and M. Potkonjak, "Techniques for intellectual property protection of DSP designs," in *Proceedings IEEE International Conference Acoustics, Speech, and Signal Processing*, Seattle, Washington, USA, vol.5, May 1998, pp. 3133-3136.
- [HUA01] Y. Huang, C. C. Tsai, N. Mukherjee, O. Samman, D. Devries, W. T. Cheng, and S. M. Reddy, "On RTL scan design," in *Proceedings IEEE International Test Conference*, Baltimore, Maryland, USA, October 2001, pp.728-737.
- [IBS] <http://ibs.etgroup.net>.
- [INTEL'08] <http://www.intel.com/technology/architecture-silicon>, 2008.
- [JAI03] A. K. Jain, L. Yuan, P. R. Pari, and G. Qu, "Zero overhead watermarking technique for FPGA designs," *Proceedings ACM Great Lakes Symposium on VLSI*, Washington, D.C., USA, April 2003, pp. 147-152.
- [JOU98] J.-Y. Jou, and D.-S. Chou, "Sensitisable-path-oriented Clustered Voltage Scaling Technique for Low Power," *IEE Proceedings Computers and Digital Techniques*, vol. 145, no. 4, July 1998, pp. 301-307.
- [KAH01] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. 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, October 2001, pp. 1236-1252.
- [KIR98a] D. Kirovski, Y. Y. Hwang, M. Potkonjak, and J. Cong, "Intellectual property protection by watermarking combinational logic synthesis

- solutions,” *Proceedings IEEE/ACM International Conference on CAD*, San Jose, California, November 1998, pp.194-198.
- [KIR98b] D. Kirovski, and M. Potkonjak, “Intellectual property protection using watermarking partial scan chains for sequential logic test generation,” in *Proceedings IEEE High Level Design, Verification, and Test*, La Jolla, California, USA, November 1998.
- [KIR03] D. Kirovski, and M. Potkonjak, “Local watermarks: methodology and application to behavioral synthesis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 9, September 2003, pp. 1277-1283.
- [KIR06] 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, December 2006, pp. 2687-2696.
- [KO04] H. F. Ko, and N. Nicolici, “Functional scan chain design at RTL for skewed-load delay fault testing,” in *Proceedings IEEE Asian Test Symposium*, Kenting, Taiwan, November 2004, pp. 454-459.
- [KOM88] N. Komatsu, and H. Tominaga, “Authentication system using concealed images in telematics,” *Memoirs of the School of Science and Engineering*, Waseda University, Japan, No. 52, 1988, pp. 45-60.
- [KOU03] F. Koushanfar, I. Hong, and M. Potkonjak, “Behavioral synthesis techniques for intellectual property protection,” Technical Report, Computer Science Department, University of California, Los Angeles, CA, USA, 2003.
- [KOU05] F. Koushanfar, I. Hong, and M. Potkonjak, “Behavioral synthesis techniques for intellectual property protection,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, no. 3, July 2005, pp. 523-545.
- [KYU94] R. J. Kyung, 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, February 1994, pp. 167-177.
- [LAC98a] J. Lach, W. Mangione-Smith, and M. Potkonjak, “Fingerprinting digital

- circuits on programmable hardware,” *Information Hiding*, Lecture Notes in Computer Science, vol. 1525, Springer Verlag, 1998, pp. 16-31.
- [LAC98b] J. Lach, W. Mangione-Smith, and M. Potkonjak, “FPGA fingerprinting techniques for protecting intellectual property,” in *Proceedings IEEE Custom Integrated Circuits Conference*, Santa Clara, California, May 1998, pp. 299-302.
- [LE02] T. V. Le, and Y. Desmedt, “Cryptanalysis of UCLA watermarking schemes for intellectual property protection,” *Information Hiding*, Lecture Notes In Computer Science, vol. 2578, Springer Verlag, 2002, pp. 213–225.
- [LI04] C. -T. Li, “Digital fragile watermarking scheme for authentication of JPEG images,” *Proceedings IEE Vision, Image and Signal Processing*, vol. 151, no. 6, December 2004, pp. 460-466.
- [LINC98] C. C. Lin, M. Marek-Sadowska, K. T. Cheng, and M. T. C. Lee, “Test-point insertion: scan paths through functional logic,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 9, September 1998, pp. 838-851.
- [LIN83] S. Lin, and D. J. Costello, *Error control coding*. Prentice Hall, 1983.
- [MAC01] J. MacBeth, and P. Lysaght, “Dynamically reconfigurable cores,” Lecture Notes in Computer Science, vol. 2147, Field-Programmable Logic and Applications, Springer, 2001, pp. 462-472.
- [MCE93] K. McElvain, “LGSynth93 benchmark set: version 4.0,” http://www.cbl.ncsu.edu/pub/Benchmark_dirs/LGSynth93/doc/iwls93.ps, May 1993.
- [MEG00] S. Meguerdichian, and M. Potkonjak, “Watermarking while preserving the critical path,” *Proceedings ACM/IEEE Design Automation Conference*, Los Angeles, California, June 2000, pp. 108-111.
- [MEN96] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.
- [MIC94] G. De Micheli, *Synthesis and Optimization of Digital Circuits*. New York: McGraw-Hill, 1994.

-
- [MOH99] S. P. Mohanty, "Digital watermarking: a tutorial review," Technical Report, Department of Electrical Engineering, Indian Institute of Science, Bangalore, India, 1999.
 - [MOI05] M. Moiz Khan, and Spyros Tragoudas, "Rewiring for watermarking digital circuit netlists," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, July 2005, pp. 1132-1137.
 - [MOO65] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, vol. 38, no. 8, April 1965, pp. 114-117.
 - [NAR95] S. Narayanan, and M. A. Breuer, "Asynchronous multiple scan chains," in *Proceedings IEEE VLSI test symposium*, Princeton, NJ, USA, April 1995, pp. 270-276.
 - [NOR96] R. B. Norwood, and E. J. McCluskey, "Synthesis-for-scan and scan chain ordering," in *Proceedings of 14th VLSI Test Symposium*, Princeton, New Jersey, USA, April 1996, pp. 87-92.
 - [OLI99] A. L. Oliveira, "Robust techniques for watermarking sequential circuit designs," in *Proceedings IEEE/ACM Design Automation Conference*, New Orleans, Louisiana, USA, June 1999, pp. 837-842.
 - [OLI01] 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, September 2001, pp. 1101-1117.
 - [PEN99] J. M. Pena, and A. L. Oliveira, "A new algorithm for exact reduction of incompletely specified finite state machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 11, November 1999, pp. 1619 – 1632.
 - [QU98] G. Qu, and M. Potkonjak, "Analysis of watermarking techniques for graph coloring problem," in *Proceedings IEEE/ACM International Conference on Computer Aided Design*, November 1998, pp. 190-193.
 - [RAN98] R. K. Ranjan, V. Singhal, F. Somenzi, and R. K. Brayton, "On the optimization power of retiming and resynthesis transformations," in *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, November 1998, San Jose, California, USA, pp. 402-407.

-
- [RAS99] A. Rashid, J. Asher, W. H. Mangione-Smith, and M. Potkonjak, "Hierarchical watermarking for protection of DSP filter cores," in *Proceedings IEEE Custom Integrated Circuits Conference*, New York, USA, May 1999, pp. 39-42.
 - [SAB85] K. Sabnani, and A. Dahbura, "A procedure for generating protocol tests," in *Proceedings 9th Data Communications Symposium*, Whistler Mountain, B. C., Canada, September 1985.
 - [SEN92] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Sequential circuit design using synthesis and optimization," in *Proceedings IEEE International Conference on Computer Design: VLSI in Computers and Processors*, Cambridge, MA, USA, October 1992, pp. 328 – 333.
 - [SYN04] Synopsys, Inc. Design Compiler 1 Workshop. Synopsys Customer Education Services, 2004.
 - [TOR00] I. Torunoglu, and E. Charbon, "Watermarking-based copyright protection of sequential functions," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 3, February 2000, pp. 434-440.
 - [VSI97] VSI Alliance, Fall Worldwide Member Meeting: A Year of Achievement. Santa Clara, California, USA, October 1997.
 - [VSI00] VSI Alliance, "Intellectual property protection white paper: schemes, alternatives and discussion version 1.0," September 2000.
 - [WAG96] K. D. Wagner, and S. Dey, "High-level synthesis for testability: a survey and perspective," in *Proceedings of 33rd Annual Conference on Design Automation*, Las Vegas, Nevada, June 1996, pp. 131-136.
 - [WIPO] <http://www.wipo.int/about-ip/en/>
 - [WON01] J. L. Wong, D. Kirovski, and M. Potkonjak, "Computational forensic techniques for intellectual property protection," in *Information Hiding*, Lecture Notes in Computer Science, vol. 2578, Springer Verlag, 2001, pp. 66-80.
 - [YUA06] L. Yuan, G. Qu, L. Ghouti, and A. Bouridane, "VLSI design IP protection: solutions, new challenges, and opportunities," *Proceedings*

Bibliography

IEEE NASA/ESA Conference on Adaptive Hardware and Systems, June 2006, pp. 469-476