HARDWARE (IP) SECURITY FOR DSP AND MULTIMEDIA APPLICATIONS

Ph.D. Thesis

By

MAHENDRA RATHOR

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

SEPTEMBER 2021

HARDWARE (IP) SECURITY FOR DSP AND MULTIMEDIA APPLICATIONS

A THESIS

Submitted in partial fulfillment of the
requirements for the award of the degree
of
DOCTOR OF PHILOSOPHY

by
MAHENDRA RATHOR

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
SEPTEMBER 2021

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **HARDWARE (IP) SECURITY FOR DSP AND MULTIMEDIA APPLICATIONS** in the partial fulfillment of the requirements for the award of the degree of **DOCTOR OF PHILOSOPHY** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING, INDIAN INSTITUTE OF TECHNOLOGY INDORE**, is an authentic record of my own work carried out during the time period from DECEMBER, 2018 to SEPTEMBER 2021 under the supervision of Dr. ANIRBAN SENGUPTA, Associate Professor, Indian Institute of Technology, Indore.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Signature of the student with date
(MAHENDRA RATHOR)

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

Signature of Thesis Supervisor with date
(ANIRBAN SENGUPTA)

MAHENDRA RATHOR has successfully given his/her Ph.D. Oral Examination held on

Signature of Chairperson (OEB)
Date:

Signature of External Examiner
Date:

Signature(s) of Thesis Supervisor(s)
Date:

Signature of PSPC Member #1
Date:

Signature of PSPC Member #2
Date:

Signature of PSPC Member #3
Date:

Signature of Convener, DPGC
Date:

Signature of Head of Discipline
Date:

ACKNOWLEDGEMENTS

First and foremost, I wish to express my sincere gratitude to my supervisor Dr. Anirban Sengupta for providing me the opportunity to do work under their supervision. I wish to thank them for their persistence and faith in me, without their relentless effort and guidance I would have not been to understand the importance of research and the sacrifice it requires to reach a certain level. I have learned a lot from their invaluable advice, kindness, profoundness and experience. Under their supervision, I transformed into a person with more values.

I also owe a mention to Dr. Abhishek Srivastava, Dr. Manish Goyal and Dr. Puneet Gupta for their valuable feedbacks on my research work throughout these years.

Further, I wish to express my deepest gratitude to my parents, for their strong belief in me and for their continuous support all the way. I also wish to express my gratitude to my dear wife, Lakshmi, for being the driving force of my career and for her moral and emotional support throughout the PhD work. Thank you for being with me every single moment to keep me always prepared to work for the past three years.

I wish to thank all the faculty members and colleagues for their continuous support. The time spent at IIT Indore will always remain the most valuable memory of my life.

At last, I wish to thank IIT Indore, DST and CSIR to help financially and providing me an opportunity to present my research at international grounds.

DEDICATED TO MY FATHER LATE SHRI MOHANLAL RATHOR

ABSTRACT

A core based design paradigm has become popular in the semiconductor business market for the last few decades. The underlying reasons are the increasing time to market pressure, design complexity and cost of system-on-chip (SoC) designs. However, the contrast impact of core based design paradigm is the susceptibility of the intellectual property (IP) cores towards the hardware threats of IP piracy and hardware Trojan insertion. An adversary in the untrusted SoC design house may infringe or misuse the IP core for personal benefits. Moreover, the reliance of chip manufacturing on a distinct offshore foundry also enhances the risks of IP piracy and potential Trojan insertion.

The digital signal processing (DSP) and multimedia applications are thriving in the modern consumer electronics (CE) market. The stringent performance and low power demand have enforced the realization of DSP and multimedia applications through their hardware accelerator or application specific IPs. However, owing to the proliferating usage of DSP and multimedia IPs in the SoCs, their security concerns cannot be undervalued. Hence, an IP core designer needs to employ protection measures against the piracy and potential Trojan insertion attacks to ensure trust in hardware. For highly complex designs such as DSP and multimedia cores, a high level synthesis (HLS) framework is amenable to employing security mechanisms. Towards the security of IP cores, this thesis contributes the following novel methodologies: (a) IP core steganography approaches to secure DSP cores against piracy, (b) a hologram based obfuscation approach to thwart the potential Trojan insertion attack, (c) double line of defense approaches based on structural obfuscation and IP vendor’s secret mark to counter both Trojan insertion and IP piracy attacks, (d) secured hardware accelerator design approach for various image processing filter applications and a DFT processor.

LIST OF PUBLICATIONS

PEER-REVIEWED JOURNALS (12):

1. A. Sengupta and M. Rathor, "Protecting DSP Kernels using Robust Hologram based obfuscation", ***IEEE Transactions on Consumer Electronics***, vol. 65, no. 1, Feb 2019, pp. 99-108, **Impact Factor: 2.947**.

2. A. Sengupta and M. Rathor, "IP Core Steganography for Protecting DSP Kernels used in CE Systems," ***IEEE Transactions on Consumer Electronics***, vol. 65, no. 4, Nov. 2019, pp. 506 – 515, **Impact Factor: 2.947**.

3. M. Rathor and A. Sengupta, "IP Core Steganography Using Switch Based Key-Driven Hash-Chaining and Encoding for Securing DSP Kernels Used in CE Systems," ***IEEE Transactions on Consumer Electronics***, vol. 66, no. 3, pp. 251-260, Aug. 2020, **Impact Factor: 2.947**.

4. A. Sengupta and M. Rathor, "Enhanced Security of DSP Circuits Using Multi-Key Based Structural Obfuscation and Physical-Level Watermarking for Consumer Electronics Systems", ***IEEE Transactions on Consumer Electronics***, vol. 66, no. 2, May 2020, pp. 163-172, **Impact Factor: 2.947**.

5. A. Sengupta and M. Rathor, "Obfuscated Hardware Accelerators for Image Processing Filters-Application Specific and Functionally Reconfigurable Processors," ***IEEE Transactions on Consumer Electronics***, vol. 66, no. 4, pp. 386-395, Nov. 2020, **Impact Factor: 2.947**.

6. A. Sengupta and M. Rathor, "Security of Functionally Obfuscated DSP core against Removal Attack using SHA-512 based Key Encryption Hardware", ***IEEE Access Journal***, vol. 7, pp. 4598-4610, 2019, **Impact Factor: 3.367**.

7. M. Rathor and A. Sengupta, "Robust Logic locking for Securing Reusable DSP Cores," ***IEEE Access***, vol. 7, pp. 120052-120064, 2019, **Impact Factor: 3.367**.

8. A. Sengupta and M. Rathor, "Structural Obfuscation and Crypto-Steganography based Secured JPEG Compression Hardware for Medical Imaging Systems", ***IEEE Access***, vol. 8, pp. 6543-6565, 2020, **Impact Factor: 3.367**.

9. M. Rathor and A. Sengupta, "Low-Cost Robust Anti-Removal Logic for Protecting Functionally Obfuscated DSP core against Removal Attack", ***IET Electronics Letters***, vol. 55, no. 7, pp. 374-376, 2019, **Impact Factor: 1.1316**.

10. A. Sengupta and M. Rathor, "Crypto based Dual phase Hardware Steganography for Securing IP cores," ***IEEE Letters of the Computer Society,*** vol. 2, no. 4, pp. 32-35, 2019.

11. M. Rathor and A. Sengupta, "Design Flow of Secured N-Point DFT Application Specific Processor Using Obfuscation and Steganography," ***IEEE Letters of the Computer Society,*** vol. 3, no. 1, pp. 13-16, 2020.

12. A. Sengupta, M. Rathor, S Patil, and NG Harishchandra, "Securing Hardware Accelerators Using Multi-Key Based Structural Obfuscation," ***IEEE Letters of the Computer Society,*** vol. 3, no. 1, pp. 21-24, 2020.

BOOK CHAPTERS (6):

13. A. Sengupta and M. Rathor, "Designing Secured N-point DFT Hardware Accelerator using Obfuscation and Steganography", ***IET Book: Secured Hardware Accelerators for DSP and Image processing applications***, 2021, Print: 978-1-83953-306-8, eBook: 978-1-83953-307-5.

14. A. Sengupta and M. Rathor, "Structural transformation and obfuscation frameworks for Data-intensive IPs", ***IET Book: Secured Hardware Accelerators for DSP and Image processing applications***, 2021, Print: 978-1-83953-306-8, eBook: 978-1-83953-307-5.

15. A. Sengupta and M. Rathor, "Security of Functionally Obfuscated DSP cores", ***IET Book: Frontiers in Securing Hardware IP Cores: Forensic detective control and obfuscation techniques***, 2020, ISBN: 978-1-83953-031-9/978-1-83953-032-6.

16. A. Sengupta and M. Rathor, "Hologram based Structural Obfuscation for DSP Cores", ***IET Book: Frontiers in Securing Hardware IP Cores: Forensic detective control and obfuscation techniques***, 2020, ISBN: 978-1-83953-031-9/978-1-83953-032-6.

17. A. Sengupta and M. Rathor, "Hardware (IP) Watermarking during Behavioural Synthesis", ***Springer Book: Behavioral Synthesis for Hardware Security***, 2020, Invited Chapter (Eds.Srinivas Katkoori, Sheikh Ariful Islam).

18. A. Sengupta and M. Rathor, "Design Space Exploration of DSP hardware using Bacterial Foraging and Particle Swarm Optimization Algorithm for Power/Area-Delay Tradeoff", ***IET Book: Low Power Nanoscale IC Design*, **Invited Book Chapter, 2020, Print ISBN: 978-981-15-7936-3.

PEER- REVIEWED CONFERENCE PUBLICATIONS (4):

19. M. Rathor, P. Sarkar, VK Mishra and A. Sengupta, "Securing IP Cores in CE Systems using Key-driven Hash-chaining based Steganography", ***Proceedings of 10th IEEE International Conference on Consumer Electronics- Berlin (ICCE Berlin)***, Germany, 2020, pp. 1-4.

20. M. Rathor and A. Sengupta, "Obfuscating DSP Hardware Accelerators in CE Systems Using Pseudo Operations Mixing", ***Proceedings of 4th IEEE International Conference on Zooming Innovation in Consumer Electronics 2020 (ZINC 2020)***, Serbia, 2020, pp. 218-221.

21. M. Rathor and A. Sengupta, "Enhanced Functional Obfuscation of DSP core using Flip-Flops and Combinational logic", ***Proceedings of 9th IEEE International Conference on Consumer Electronics (ICCE)- Berlin***, Berlin, 2019, pp. 1-5.

22. A. Sengupta and M. Rathor, "Improved Delay Estimation Model for Loop Based DSP Cores", ***Proceedings of 37th IEEE International Conference on Consumer Electronics (ICCE)***, Las Vegas, USA, Jan 2019, pp. 1-4.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Vi	i^{th} storage variable
$E_{Vi,Vj}$	Entropy of (Vi, Vj) storage variable pair
$ME_{Vi,Vj}$	Maximum entropy of (Vi, Vj) storage variable pair
T^E	Threshold entropy
Pc	Probability of coincidence
G	Number of registers in the design
f	Number of stego-constraints embedded
F_b^a	b^{th} instance of functional unit of vendor type a
M_b^a	b^{th} instance of multiplier of vendor type a
A_b^a	b^{th} instance of adder of vendor type a
DFG_{sab}	Scheduled, allocated and binded DFG
$N(Z_i)$	Number of instances of functional unit Z of type i
$\mathrm{key}_{\mathrm{m}}^{\mathrm{s}}$	Maximum possible size of the stego-key
U	Total possible encodings for a DSP application
$\mathrm{A}_{\mathrm{m}}^{\mathrm{sk}}$	Attacker's max effort in terms of deducing the stego-key
$\mathrm{A}_{\mathrm{m}}^{\mathrm{eb}}$	Attacker's effort in terms of finding the encoded bits
A_T^S	Attacker's total effort in determining the stego-constraints
$C_f(Zi)$	Cost of the design scheduled using the resource constraints Zi
L_T	Design delay/latency
A_T	Design area
A_{max}	Maximum area
L_{max}	Maximum delay
β_1, β_2	Weights assigned to latency and area in calculating the cost
S_n	Strength of obfuscation
G_c^f	Number of gates affected due to applying obfuscation
G_c^T	Number of gates in the respective un-obfuscated design
ΔG_c^{fb}	Difference in gate count between obfuscated design and un-obfuscated version
G_c^{ip}	Number of gates altered in terms of input connectivity post obfuscation
$ODFG_{sab}$	Obfuscated scheduled, allocated and binded DFG
u	Number of instances of FU type U_r, where r is the total types of FUs

x	Number of Muxes of size X_v, where v indicates various sizes of Mux in the design
d	Number of Demuxes of size D_e, where e indicates various sizes of Demux in the design
α, β, γ	Signature variables for physical level watermarking
T_S	Tamper tolerance
Q	Number of distinct variables in the chosen signature for physical level watermarking
Z	Size of the signature in physical level watermarking
$[A]_{(I\times J)}$	A pixel matrix of image of size I×J
X_{ij}	A pixel value of the input image where i and j vary from 0 to I-1 and 0 to J-1 respectively
$[F]_{(n\times m)}$	Kernel matrix of size n×m
f_{pq}	Kernel values
O_w	Output pixel value
[O]	Output matrix
F^B	kernel matrix of a 3×3 blur filter
F^S	kernel matrix of a 3×3 sharpening filter
F^{VE}	kernel matrix of a 3×3 vertical embossment filter
F^{HE}	kernel matrix of a 3×3 horizontal embossment filter
F^{ED}	kernel matrix of a 3×3 edge detection filter
T_{ILB}	Total ILBs in a functionally obfuscated design
T_{DLC}	Total DLCs in a functionally obfuscated DSP core
ω	Designer specified tuning variable for DLCs insertion
P_v	Probability of obtaining valid key in exhaustive trials
K_b	Total number of key-bits in DLC based logic locking
En^s	Encryption strength
N_{op}^{AES}	Number of encrypted output bits generated from AES128
N_{kbits}^{total}	Total number of DLC key bits in a design

ACRONYMS

HLS	High Level Synthesis
VLSI	Very Large Scale Integration
IP	Intellectual Property
DSP	Digital Signal Processor
CE	Consumer Electronics
IC	Integrated Circuits
SoC	System on Chip
RTL	Register Transfer Level
VHDL	Very High Speed Integrated Circuit Hardware Description Language
GDS	Graphic Database System
ALU	Arithmetic Logic Unit
DFG	Data Flow Graph
CDFG	Control Data Flow Graph
SDFG	Scheduled Data Flow Graph
FSM	Finite State Machine
RE	Reverse Engineering
CS	Control Step
CIG	Colored Interval Graph
UF	Unrolling Factor
3PIP	3rd party Intellectual Property
ASIC	Application Specific Integrated Circuit
PSO	Particle Swarm Optimization
DSE	Design Space Exploration
FU	Functional Unit
HLT	High Level Transformation
opn	Operation
LU	Loop Unrolling
THT	Tree Height Transformation
ROE	Redundant Operation Elimination
LICM	Loop Invariant Code Motion
SOK	Structural Obfuscation Key

RFC	Round Function Computation
FF	Flip-Flop
DCT	Discrete Cosine Transform
IDCT	Inverse Discrete Cosine Transform
DWT	Discrete Wavelet Transform
DIT	Decimation In Time
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
DFT	Discrete Fourier Transform
JPEG	Joint Photographic Expert Group
MPEG	Moving Picture Expert Group
AES	Advanced Encryption Standard
SHA	Secure Hash Algorithm
MDS	Maximum Distance Separable
ARL	Anti-Removal Logic
ILB	IP core Locking Block
DLC	DSP Locking Cell
Mux	Multiplexer
Demux	Demultiplexer
CLLC	Combinational Logic Locking Cell
SLLC	Sequential Logic Locking Cell

Chapter 1

Introduction

We are fortunate enough for living in an era where internet speed is touching 5G, 8D audio effects are mesmerizing us, high definitional (HD) videos and graphics are fascinating our generation; thanks to electronics. The ubiquitous applications of electronics have led it in becoming an integral part of consumer's life [1]. At the heart of the consumer electronics (CE) systems such as laptops, tablets, and smart phones etc., there functions a system-on-chip (SoC). A SoC is designed using various modules such as functional blocks, memory units and memory controller and different peripherals for wireless and wired communication etc. Instead of designing a SoC from scratch, its various modules are purchased from third party vendors or designers. The pre-designed and pre-verified modules of a SoC, procured from distinct design houses (fabless), also referred to as third party intellectual property (3PIP) cores. And, this kind of system design paradigm is called as core based design paradigm. The core based design paradigm results in benefits in terms of reduced cost, alleviated design complexity and lower time to market pressure. This is because, designing a SoC from scratch is not only cumbersome due to higher design complexity, but also involves higher investment in terms of time and resources or cost. Further, in the design supply chain, the SoC design file is sent to a distinct offshore manufacturing house (also referred to as a foundry) for the chip fabrication. Relying on distinct foundry for the chip fabrication is economic, because building and maintaining a foundry require billions of dollar investment. Thereby, different entities (third party IP vendors, system integrator, and foundry) get involvement in the IC design chain. This helps in sustaining the IC design process at lesser cost, lower design complexity and lower time requirement [2]-[5].

In the SoC designs of consumer electronic systems, digital signal processing (DSP) and multimedia cores play a pivotal role owing to their utility in various applications such as de-noising, image compression-decompression and video encoding-decoding etc. Because of the vital role of DSP and multimedia applications in modern electronic systems, their usage has been proliferating for the past few decades [6]. Since the DSP and multimedia applications are intended to be executed in the systems where stringent performance and power requirement are imperative, therefore these applications are designed as application specific processors or hardware accelerators to be integrated in the SoCs [1], [2], [7].

Besides consumer applications, the DSP and multimedia processors also claim their utility in some critical applications such as military and healthcare. For example, in military and defense applications, DSP cores integrated in the SoCs facilitate live video streaming from unmanned aerial vehicles (UAVs). Further, in the healthcare and medical systems, joint photographic experts group (JPEG) processors are used for compression-decompression of medical images to enable low storage requirement and low-bandwidth transmission.

However, the involvement of distinct entities (or houses) in the design chain raises the issue of untrustworthiness. This is because an adversary or attacker in an untrustworthy house may realize his/her malicious intents of IP infringement or misuse [3], [4]. This invites different hardware security threats which can be broadly of two types: (i) IP piracy [1], [2] (ii) hardware Trojan insertion [1], [2]. As the DSP and multimedia based IP cores possess significant role in consumer, military and healthcare applications, therefore their security perspective cannot be overlooked. A fake or infringed IP used in military systems or medical devices can even endanger the human’s life. Hence, security of IP cores is of paramount importance to ensure secure, safe and reliable functioning of electronic systems. Additionally, securing IP cores is also indispensable from the vendor’s perspective in order to save him/her from huge revenue and reputation loss due to IP piracy [8].

This chapter briefly discusses the background on the various key aspects that the proposed IP core (or hardware) security techniques are developed around. The first section puts the background on IP cores. Further, the second section provides an overview of DSP and multimedia applications and corresponding algorithmic representations. The third section discusses the various threats to IP cores. The fourth section provides a background on high level synthesis

(HLS) process and discusses its role in hardware (IP) security. Finally, the fifth section presents the thesis organization.

1.1. Semiconductor IP Core and its Various Forms

In the electronics domain, a reusable logic block which is an intellectual property of the designer or owner is termed as intellectual property (IP) core. An IP core is designed and sold into the market in its various forms such as (i) soft IP core (ii) firm IP core (iii) hard IP core [9],[10]. A synthesizable register transfer level (RTL) code in a hardware description language (such as VHDL or Verilog) or schematic design is referred to as a soft IP core. With the soft IP cores, a chip designer has the flexibility to modify the design parameters as per the requirement. The performance of a soft IP core can vary considerably as it gets transformed into lower levels of design abstractions. Further, a firm IP core is another type which is a technology dependent gate level netlist of the design. It is lesser modifiable than a soft IP core. Further, a hard IP core is a GDS-II design database or layout file of the design. It cannot be substantially modified by the designers as it has lower-level physical description that is specific to a particular process node. The core based design paradigm is based on the reusable IP cores, hence playing a pivotal role in alleviating the design complexity and helping in satisfying time-to-market requirement [10], [11]. A typical SoC is composed of various kinds of IP cores such as a general purpose processor core for addressing general purpose functionality, DSP cores to perform application specific functionality, memory controller IP cores to manage the data exchanged between the memories and other modules and interface IP core such as UART, SPI, I2C to facilitate serial communication and so on.

1.2. DSP and Multimedia Applications and Corresponding Algorithmic Representation

The digital signal processing (DSP) algorithms such as finite impulse response (FIR) filter, infinite impulse response (IIR) filter, discrete cosine transform (DCT), inverse discrete cosine transform (IDCT), fast Fourier transform (FFT) and discrete wavelet transform (DWT) etc. have wide applications in modern electronic systems. Additionally, processors used in multimedia applications such as joint photographic experts group (JPEG) compression-decompression (codec), moving picture experts group (MPEV) etc. are also dominating in today's consumer electronic systems. The DSP and multimedia cores are employed to perform various functionalities such as filtering, image compression-decompression, image enhancement, audio/ video encoding-decoding etc. [1], [2], [8].

In order to generate an application specific processor of a DSP or multimedia application, its high level or algorithmic or behavioral description is taken as input to the synthesis process. The algorithmic description can be of various forms such as a C/C++ code or a mathematical representation (an equation representing input-output relationship) etc. For example, an algorithmic description in the form of a mathematic representation of a 4-point DCT application is given as follows [2]:

$$X[0]=c1*x[0]+ c2* x[1] + c3* x[2] + c4* x[3] \tag{1.1}$$

Where, x[0] to x[3] are input variables, c1 to c4 are input coefficients and X[0] is an output.

1.3. Threats to IP Cores

As discussed earlier, the various offshore entities such as a 3PIP vendor, a system integrator and a contract foundry participate in the current semiconductor business model, leading to the globalization of the design supply chain. However, in this semiconductor business model, the design data transfer is asymmetric. More explicitly, an IP design is transferred or sold to a system integrator however there is no transfer of intellectual property the other way around. Similarly, the design file is sent to a contract foundry for chip fabrication where the design data is transparent to the foundry. In these cases, the design IP is transferred only in one direction [3]. Hence, this globalization of the design supply chain and asymmetric nature of this business model render the participating entities untrustworthy, resulting in the emergence of various hardware threats in the semiconductor market. Fig. 1.1 depicts the potential threat scenario in the untrustworthy houses (SoC design house and foundry) where the IP cores designs are vulnerable to hardware threats of IP piracy and hardware Trojan insertion [8], [11].

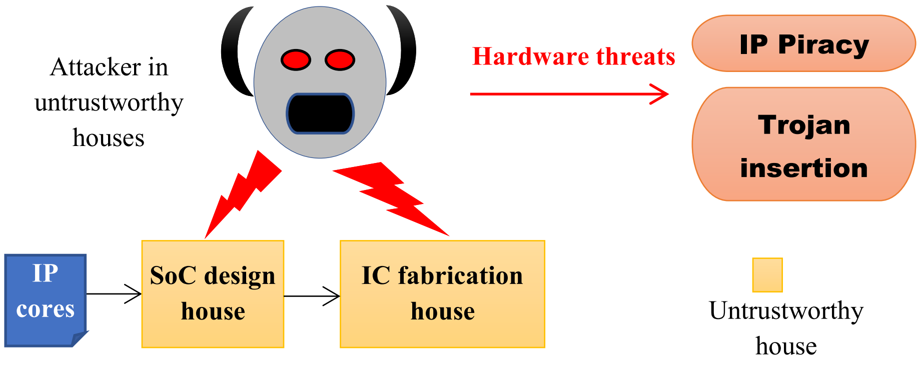


Fig. 1.1. Potential hardware security threats in untrustworthy houses

In 2010, the first case against a counterfeit-chip broker was prosecuted by the United States. Thousands of counterfeit chips, many of which were intended to be used in military products, were sold by the company VisionTech [14]. Further, the data provided by IHS (Information Handling Services, Englewood, CO, USA) displays that reports of counterfeit parts have quadrupled since 2009. Legitimate electronics enterprises lose out on about $100 billion of global revenue per year because of counterfeiting. Around 1% of semiconductor sales are projected to be those of fake components [15]. Additionally, in 2007, Syrian radar botched to warn of an incoming air strike [16]. A backdoor (hardware Trojan) built into the system's chips was rumored to be responsible. This raises questions as to just how much the global supply chain for ICs can be relied [16]. A brief discussion on the IP piracy and hardware Trojan insertion threats is as follows:

1.3.1 IP Core Piracy

IP piracy [3], [4], [12], 13] may take various forms such as IP counterfeiting, IP cloning and false claim of IP ownership which may cause adverse effects in the form of huge financial loss for the genuine IP vendor/designer and harm of IP vendor's brand and reputation. Fig. 1.2 depicts the potential IP piracy threat scenario.

IP counterfeiting: In this threat, a potential adversary sells a low quality or fake IP into the market under the brand name of an authentic or genuine IP vendor. The adversary can have various objectives of doing so, such as: (i) the adversary may have malicious intents of sabotaging authentic vendor's reputation by enabling the integration of low quality counterfeit IPs as genuine

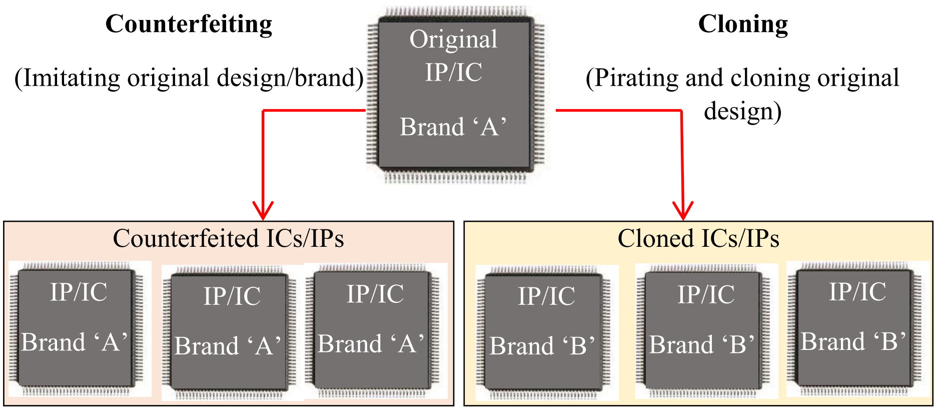


Fig. 1.2. Counterfeiting and cloning attacks, where brand 'A' indicates the brand name of a genuine IP vendor and brand 'B' indicates the brand name used by an adversary

ones into the SoCs (ii) the adversary may be intending earning illegal income under the brand name of an authentic vendor (iii) the adversary may supply fake IPs to the offshore system integrators because of some national interests. The counterfeit IPs may not be rigorously tested in terms of their functionality, safety and reliability. Additionally, they may also contain some secret malicious logic (hardware Trojan) to adversely affect the system operation, performance and security. Therefore, detection of counterfeit IPs and ensuring integration of only authentic IPs in SoCs is vital for safe, reliable and secure functioning of electronic systems.

IP cloning and false claim of IP ownership: In this threat, an adversary (rouge system integrator) instantiates the third party vendor's IP more number of times in the SoC designs than specified in the agreement of IP use. Moreover, the adversary (rouge system integrator or foundry) may steal the vendor's IP and sell it illegally into the market as his/her own. Thus an adversary may clone the true vendor's IP to earn the illegal income. Moreover, he/she may claim the ownership of stolen/cloned IP fraudulently. The IP cloning and false claim of IP ownership threat harms the genuine IP designer in terms of both revenue loss and ownership loss. The potential adversaries could be an untrustworthy SoC designer and an untrustworthy foundry. Therefore a true IP designer needs to secure his IPs against the cloning to save from the illegal revenue loss and the fraudulent claim of IP ownership.

1.3.2. Hardware Trojan Insertion

Hardware Trojan [17], [18] is a malicious/intended modification in the design to cause different kinds of adverse effects such as compromising secret information from the system, wrong functional output, excessive power consumption and degraded performance etc. The hardware Trojan insertion is a back door entry of malicious logics/components into the design by an adversary in an untrustworthy design house or foundry. The hardware Trojan is intentionally designed to be stealthy in nature so that it cannot be readily detected during the validation process. Further, it is designed to be triggered only upon certain rare events at particular nodes in the design. The globalization of IC design chain has given the opportunity of inserting hardware Trojans to rouge elements. Hence, in this globalized design chain, the IP core designer needs to employ some preventive control against the possible Trojan insertion in an untrustworthy house by an adversary, to ensure the trust in hardware [2], [8].

1.4. Background on High Level Synthesis and its Importance in IP Core Security

The synthesis is a design process whose objective is to find a structure that implements the behavior required, for a given input specification with a set of constraints and goals to be fulfilled. Here, the mapping from inputs to outputs (i.e. the way components of the system interact) defines the behavior and the set of interconnected components that composes the system defines the structure. The synthesis can be performed at different levels of design abstraction as a design can be represented at different levels of detail. The high level synthesis (HLS) [19]-[22] also referred to as behavioral or architectural synthesis is a high level design process to convert an algorithmic description of a system into a structure implemented in the form of datapath and controller. Thus obtained structure is also called as register transfer level (RTL) design which is composed of high level components such as functional units (FUs), registers, interconnect units such as multiplexers (Muxes) and demultiplexers (Demuxes) etc.

Let’s discuss the internal details of the HLS process in brief [8], [19]-[22]. The HLS process first takes the algorithmic representation such as C/C++ code or a mathematical function as an input and generates a corresponding intermediate representation called data flow graph (DFG) or a control data flow graph (CDFG). For example, the algorithmic description of a 4-point DCT application has been given in eq. (1) and its corresponding DFG representation is shown in Fig. 1.3(a). Further, the DFG can be subjected to various kinds of high level transformations such as loop unrolling (LU), redundant operation elimination (ROE), tree height transformation (THT) and loop invariant code motion (LICM) etc., depending on the nature of the application or the applicability of the transformation. Thus obtained DFG is subjected to different phases of the HLS process viz. scheduling, allocation and binding. In the scheduling phase, different operations of the application (nodes of the DFG) are assigned to a particular time stamp or control step (CS) to be executed within. An algorithm chosen to perform the scheduling depends upon the given constraints on timing or resources. Post scheduling, hardware or resource allocation phase is performed where the resources are allocated to the operations to be executed and to the variables to stored, from the HLS library. A particular kind of resource (of certain specifications) to be allocated depend on the latency, power and area constraints of the design. For example, if the higher performance requirement is more important than the compact area for a given design then a faster resource (e.g. adder/multiplier) consuming higher area is preferred over a slower resource consuming lower area. Post allocating resources, binding phase is performed which decides which operation is to be assigned to which instance of the respective functional unit (FU) and which variable is to be assigned to which register. This determines the sharing of FU resources to execute a number of operations and that of registers to execute a number of storage variables, in the different control steps. Thus the binding phase provides the information of interconnect resources such as Muxes and Demuxes required for sharing of FUs and registers. Fig. 1.3(b) shows the scheduled, allocated and binded DFG of the 4-point DCT core, where s0 to s10 are the storage variables, T0 to T4 are the control steps, R, I, G and O are the four different registers, M1 and M2 are two multiplier resources and A1 is an adder resource. Post scheduling, allocation and binding phases, datapath and controller synthesis phase of the HLS process is performed. This phase synthesizes the RTL datapath of the design using the allocated FU resources, registers, and latches and using the Muxes

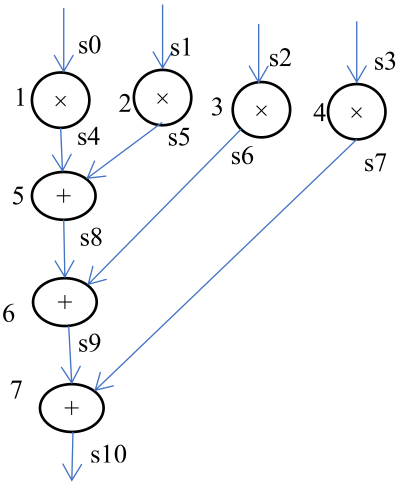


Fig. 1.3(a) DFG of 4-point DCT

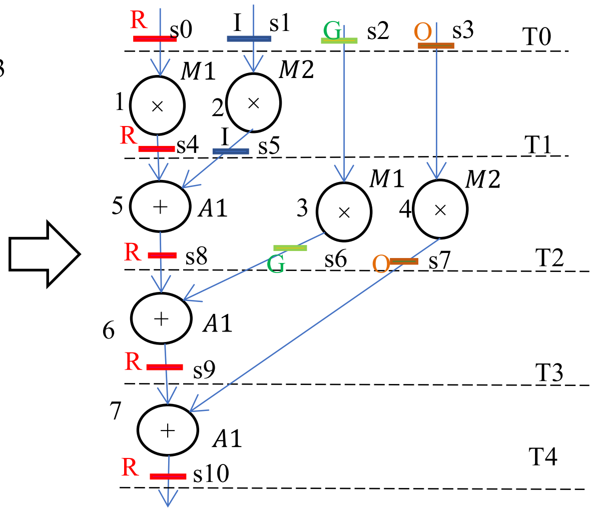


Fig. 1.3(b) Scheduled and hardware allocated 4point DCT using resource constraints of 1 (+) and 2(×)

and Demuxes (determined through the binding phase) and that the controller using the scheduling and dependency information of the operations. The controller provides the various control signals to enable the different units of the datapath in the respective control steps as per the scheduling. Post HLS, the design is subjected to the lower level design processes such as logic synthesis to obtain the corresponding gate level netlist and physical synthesis to obtain the respective layout design. Post obtaining the layout, it sent to a contract foundry for the chip fabrication.

Importance of HLS in IP core security: The IP core security for DSP and multimedia applications during the HLS process has a paramount importance. This is because applying a security mechanism may results into excessive area, power or delay overhead, violating the given design constraints [23]. However, the HLS integrated with the design space exploration (DSE) process [24] offers the flexibility of exploring an optimal architecture that satisfies the given area, power and latency constraints. This helps achieve a low cost secure architecture post employing the security mechanism during the HLS. Moreover, employing the security mechanism at lower abstraction levels for the DSP and multimedia applications is arduous due to their higher design complexity and non-availability of the designs (such as gate level netlist) at the lower levels. On the contrary, the DSP applications are readily available in the form of their algorithmic descriptions and their high level synthesis can easily be automated using the commercial or non-commercial tools to generate the corresponding RTL counterparts. Therefore, the security mechanisms can easily be integrated with the computer-aided-design (CAD) tools of HLS to generate the secured designs for the DSP and multimedia IP cores. Furthermore employing security during the HLS process also secures the various other forms of an IP at lower design levels such as firm IPs and hard IPs, along with the soft IPs. And we also have the security constraints more distributed throughout the design post synthesis, if we apply the security during the HLS [24], [25].

1.5. Organization of Thesis

The chapters of the thesis are organized as follows. Chapter 2 discusses the state-of-art with respect to proposed techniques. Chapter 3 discusses the proposed hardware steganography techniques for securing IP cores against piracy. Chapter 4 discusses the proposed hologram based structural obfuscation to thwart reverse engineering based attacks. Chapter 5 discusses the proposed double line of defense approach using integrated structural obfuscation and crypto-steganography to secure IP cores. Chapter 6 discusses the proposed double line of defense approach using integrated multi-key based structural obfuscation and physical level watermarking to secure IP cores. Chapter 7 discusses the proposed secured hardware accelerator design approach for image processing filters. Chapter 8 discusses the proposed techniques for securing functionally obfuscated DSP cores against removal attack. Chapter 9 discusses the proposed robust logic locking technique for preventing IP piracy. Chapter 10 discusses the experimental results of the proposed techniques and compares with the state-of-the-arts. Chapter 11 concludes the thesis and briefly discusses the future work.

Chapter 2

State of the Art

Some techniques have been proposed to counter the threats of IP piracy and hardware Trojan insertion for the past few years. This chapter discusses the state of the art techniques alongwith their limitations. This builds up the foundation for the proposed methodologies presented in this thesis. The first section presents state-of-the-art on handling IP piracy threat. The second section presents state-of-the-art on thwarting hardware Trojan insertion attack on DSP IP cores. The third section presents the objective of this thesis. The fourth section highlights the contributions of this thesis.

2.1. State of the Art on Handling IP Piracy Threat

The IP piracy threat results into huge impact on (i) the revenue and reputation of the IP creator/designer/owner and (ii) the system in which the IP core is deployed and its user. The IP piracy threat has been discussed in the section 1.3.1 of the chapter 1. The threat of IP piracy has been combated using either preventive or detective control mechanisms in the literature [1], [2].

2.1.1. Preventive control mechanism: To prevent the IP/IC piracy from happening, Roy et al. [35], proposed the EPIC (ending piracy of integrated circuits) technique which requires an external key generated by the IP owner to activate the chips. In this approach, the circuit is locked by inserting additional XOR gates on chosen non-critical paths. These XOR gates are controlled through an external key value. The normal functionality of the design is available only upon applying actual unlocking key value [35]. Further, the logic locking techniques were proposed by Yasin et al. [37], and Sengupta et al. [36]. In these logic locking based preventive controls over piracy, the IP core design functionality is locked by inserting an additional locking circuitry into the design. Logic locking, also referred to as functional obfuscation, techniques have been proposed for both combinational circuits and DSP circuits. The logic locking of combinational circuits is performed by inserting key gates (XOR and XNOR gates) [35], [37] or key driven multiplexers [38] at appropriate places into the design. And, the logic locking of DSP circuits is performed by inserting IP core locking blocks (ILBs) at some chosen output bits of the functional units into the design [36]. Performing logic locking results in a locked gate level netlist of the design. An adversary in the SoC design house or in a foundry cannot misuse or pirate the IP core without unlocking the design using a correct key value, thus thwarting the IP piracy.

Limitations: The logic locking technique is vulnerable to various kinds of popular attacks such as key sensitization attack [36], [37], removal attack [39], Boolean satisfiability (SAT) based attack [40], [41] and brute force attack. By launching the aforementioned attacks, the adversary or attacker aims at deriving the unlocked netlist. In the key sensitization, SAT and brute force attack, the attacker attempts to find the correct key value to unlock the design. However, the SAT attack is not applicable on complex DSP circuits because of multiplication operations involved [36]. Further in the removal attack, the attacker tries to remove the locking logic/circuitry from the locked netlist to obtain the unlocked version. If the attacker becomes successful in obtaining the unlocked design/netlist, the purpose of preventive control is defeated and the IP piracy becomes realizable.

2.1.2. Detective control mechanism: To enable the detection of IP piracy, hardware or IP watermarking technique was proposed by Hong and Potkonjak [26]. In this technique, the vendor's secret mark or signature is implanted into the IP core during the design process. During the verification or authentication process, the presence of vendor's watermark is detected into the design to identify whether the IP is pirated or authentic. The process of embedding watermark into a design can be performed at various levels of VLSI design abstraction, depending on the nature of the target circuits. For example, the watermarking for combinational circuit is performed during logic synthesis process. Kirovski et al. [27] proposed a watermarking method for implanting user and tool specific information into a combinational circuit during a pre-processing step of traditional logic synthesis. Design constraints for watermarking are generated using hashing such as SHA-256 and pseudo-random number generation. Further, Cui et al. [28] proposed a constraint-based adaptive watermarking method at logic synthesis level. In this method, some closed cones in an originally optimized combinational circuit are modulated for technology mapping. Furthermore, Cui and Chang [29] embeds watermark in combination circuits by substituting template. In order to embed watermark, this method replaces some specific cells with another template (that have the equivalent function) in the library without altering the topology and original functionality.

However in case of watermarking of DSP circuits, it is more efficient to embed signature constraints during the high level synthesis (HLS) process. There are few techniques in the literature which perform watermarking of DSP circuits during the HLS. Koushanfar et al. [30] proposed an IP protection technique based on hardware watermarking to combat IP piracy. This technique embeds watermark in the pre-synthesis phase of HLS or behavioral synthesis, thus influencing the end design. The watermark is embedded in the form of additional design constraints. The added extra constraints encode the author's signature into a binary number (combination of 0s and 1s) which is further represented in the 7-bit ASCII characters. To embed the author's signature during behavioral synthesis, high level description of the design is first converted in the control data flow graph (CDFG) representation. Further, CDFG is scheduled in control steps (CS). Thereafter, an interval graph (IG) is created wherein each node indicates a storage variable, and an edge between two nodes indicates overlapping of life time of two storage variables. Register allocation to these variables is performed by coloring the graph. The author signature is embedded by imposing the extra constraints during the register allocation of storage variables in the form of extra edges in the interval graph. This technique is more effective for large DSP applications. This approach assesses the protection strength in terms of probabilities of coincidence and resilience against tampering. Sengupta and Bhadauria [25], [31] proposed multi-variable watermarking for protecting DSP hardware accelerators against counterfeiting, cloning and false claim of ownership threats. This approach exploits particle swarm optimization based design space exploration (PSO-DSE) during HLS to obtain low cost optimal watermark. In this approach, authors signature is a combination of four distinct variables viz. 'i', 'I', 'T' and '!'. Hardware security constraints corresponding to the designer's chosen signature are embedded during the register allocation phase of HLS. Since the watermarking constraints are imposed during the single phase i.e. register allocation of HLS, hence referred as single phase watermarking. To embed the constraints, colored interval graph (CIG) framework is exploited, where each distinct color represents a distinct register, nodes indicates storage variables and an edge between two nodes indicates overlapping of life time of two storage variables. Each digit of multivariable signature is embedded as an additional edge between a node-pair in the CIG. In order to do so, this approach mapped each signature digit to hardware security constraints based on following encoding of aforementioned four signature variables: ‘i’ is encoded as an edge between node-pair of two prime nodes in the CIG, ‘I’ is encoded as an edge between node-pair of two even nodes, ‘T’ is encoded as an edge between node-pair of odd and even node and ‘!’ is encoded as an edge between node-pair of node number 0 and any integer. While forming node pairs to embed all signature digits, nodes are traversed in the increasing order of their number. Further, Le Gal and Bossuet [32] exploits in-synthesis phase of HLS for watermarking of DSP circuits. This approach targets the hardware that performs computationally intensive tasks in audio and video applications. This technique automatically inserts author’s watermark in order to reduce design overhead. In this concept of watermarking, empty time slots between successive high levels of data valid output are exploited to embed author’s signature. Therefore, this approach is viable for circuits having free output slots. The entire watermark is a set of sub-marks where each sub-mark is given by a mathematical relation. Input values, initial values and intermediate values of circuit compose the mathematical relation of a sub-mark. During the unused time slots of output, when data valid signal is dormant, the sub-marks are read as output values. The watermark remains invisible during static analysis because the sub-marks appear at output as dynamic transient values. Furthermore, Sengupta and Roy proposed a multi-phase watermarking technique [33], [34] based on seven variables author-signature. Signature combination comprising of seven variables is embedded during three different phases of HLS process viz. scheduling phase, hardware allocation phase and register allocation phase. Owing to large number of signature variables and embedding at three different phases, this watermarking technique is highly tamper tolerant and offers extremely low Pc. This is because the complex signature combination embedded at three phases of HLS renders the identification process of watermarking constraints into the design highly complicated and it is very less probable that the same watermarking solution is identified in an unsecured version.

Limitations: The detective control is a passive protection measure of IP cores. It does not actively prevent IP piracy by an adversary. Further, the signature used in the existing watermarking approaches is vulnerable to theft and misuse by the adversary [26]. If the chosen signature length, signature digit and their encodings into security constraints are compromised by the adversary then he/she can duplicate the true vendor's signature to cause the following impacts [2]: (i) misusing the compromised signature to fraudulently claim the ownership, thereby hindering the true IP vendor from proving his/her ownership (ii) evading IP counterfeiting detection by embedding the vendor's authentic signature (or secret mark) in counterfeit designs. Hence, this entails developing robust authentication marks (or secret marks) to enable strong detection of IP piracy.

2.2. State of the Art on Thwarting Hardware Trojan Insertion Attack

If an attacker becomes able to successfully reverse engineer the design to know its exact structure and internal details, he/she can insert the Trojan secretly into the design. Therefore, the designer's effort should be to hinder the attacker from knowing the correct structure of the design through reverse engineering (RE) [42] to secure against the hardware Trojan insertion attack [43]. By making the design structure un-obvious or un-interpretable to an adversary, the RE based attack of potential Trojan insertion can be thwarted [2], [8]. Structural obfuscation mechanism was proposed to obfuscate (make un-obvious) the design structure of an IP core to prevent against the possible Trojan insertion in an untrustworthy house [44], [45]. The potential sites for Trojan insertion could be a SoC design house or a foundry.

Lao and Parhi [44] utilized hierarchical contiguous folding (HCF) to performing the structural obfuscation. In this folding, all operations of one stage are performed before starting to perform next stage operations. More explicitly, Lao and Parhi [44] applied the obfuscation by varying the number of stages in the cascaded structure, resulting into several variation modes. For obfuscating DSP circuits, different variation modes can be implemented. Some modes produce functionally invalid outputs, however, the output can be meaningful from a signal processing point of view. Other modes can produce non-meaningful outputs. Manifold meaningful and non-meaningful modes (resulting from folding) have been utilized for obfuscation in this technique. Various modes of operations are regulated through configure data. The functional mode of a DSP design is configured by a reconfigurator which is enabled by a finite state machine (FSM). Further, this FSM is controlled by a key. If an invalid key/wrong configure data is applied, it results into either a meaningful but non-functional or non-meaningful mode. Thus folding based obfuscation results in many equivalent circuits to incur obscurity in the structure of a DSP design. The probability that a correct mode is activated is considerably reduced for achieving higher security through obfuscation. This approach mainly targets loop based DSP applications such as finite impulse response (FIR) filters etc. Further, Sengupta et al. [45] leveraged compiler based high level transformations (HLTs) to structurally obfuscate DSP hardware accelerators. Following compiler based high-level transformations have been exploited [45], [46]: redundant operation elimination (ROE), logic transformation, tree height transformation (THT), loop unrolling and loop invariant code motion. ROE technique eliminates those operations (or nodes in CDFG) whose inputs and operation type match with another operations/nodes in the CDFG. Logic transformation alters some operation types in the CDFG, maintaining the design functionality intact. THT technique causes some operations to be executed as parallel sub-computation rather than sequential execution. Thus it leads to either increase or decrease in the tree height. Loop unrolling technique unrolls the loop body based on unrolling factor. This changes the number of times reuse of FUs and also reduces design latency because of allowing parallelism. Loop invariant code motion technique moves those operations out of the loop which are not dependent on the loop iterations. Applying aforementioned complier based techniques significantly transform the CDFG of DSP application without affecting original functionality. These high level transformations considerably alter the RTL datapath of the DSP application post HLS. This alteration includes changes in size and number of Muxes and Demuxes, changes in interconnectivity of functional units (FUs) with Muxes, Demuxes and change in number of storage elements (registers) etc. In addition, Sengupta et al. [45] explored low-cost solution by integrating PSO-DSE framework with the HLS process. This PSO-DSE provides optimal resource constraints to schedule transformed/obfuscated graph which in turn leads to minimal design cost of structurally obfuscated design. Furthermore, Sengupta et al. [47] proposed THT based structural obfuscation for protecting JPEG codec hardware accelerators. And, Sengupta et al. [48] proposed hybrid transformations based structural obfuscation to protect fault secured DSP designs. A structurally obfuscated design is arduous to be successfully reverse engineered, hence thwarting hardware Trojan insertion attack.

Limitations: The applicability of existing structural obfuscation approaches rely on the nature of the intended application to be secured. The high level transformations employed in existing approaches may not be universally applied to all target applications. This demands alternative techniques which can be applied to wide variety of applications. Furthermore, the strength of obfuscation value needs to be enhanced /improved in order to achieve higher security against the potential Trojan insertion attack.

2.3. Objective of the Thesis

This thesis aims at developing novel techniques/methodologies for securing DSP and multimedia based IP cores against the foregoing hardware threats. This is achieved by setting out the following objectives:

1. To develop robust detective control mechanisms using IP core steganography for securing DSP and multimedia IP cores against the threat of IP piracy.
2. To develop a robust preventive control mechanism against the potential hardware Trojan insertion attack using hologram based structural obfuscation.
3. To develop double line of defense mechanisms to secure the DSP and multimedia IP cores to handle both the IP piracy and Trojan insertion

threats simultaneously by integrating structural obfuscation techniques with the hardware watermarking/steganography.

4. To develop a methodology for designing secured (obfuscated) and reconfigurable image processing filters hardware accelerators and a secured DFT processor.
5. To develop robust countermeasures against the potential removal attack to secure functionally obfuscated DSP cores.
6. To develop a robust preventing control mechanism against the threat of IP piracy using logic locking of DSP and multimedia IP cores.

2.4. Summary of the Contributions

- A novel detective control approach for securing against IP piracy using proposed entropy based IP core steganography. (publications: #2)
  - Proposes a novel 'IP core steganography' for enabling detection of piracy of DSP kernels.
  - The extent to which secret stego-information could be implanted is designer controlled through an 'entropy threshold' value.
  - Achieves reduced typical register overhead while improving the robustness of the IP core protection.
- A novel detective control approach for securing against IP piracy using proposed key-driven hash chaining based IP core steganography. (publications: #3, #19)
  - Proposes a novel key-driven hash-chaining based IP steganography to secure against 'counterfeit ICs/IPs with copied stego-mark'. This impedes an adversary from copying and misusing the authentic stego-mark to escape counterfeit detection process.
  - The proposed approach exploits manifold encoding schemes, switch based hash blocks (driven through stego-keys) alongwith regular hash blocks in the hash-chaining process.
  - Yields stronger security in terms of robust digital evidence (stego-mark) and larger key size at trivial area overhead.
- A novel methodology of hologram based structural obfuscation to secure DSP cores against reverse engineering based attacks. (publications: #1, #16)

  - Proposed work leverages a security image hologram feature to introduce a novel hologram based structural obfuscation technique for securing DSP cores at register transfer level.
  - Presents multiple algorithms for generating hologram based obfuscated design by integrating two DSP cores.
- A novel double line of defense methodology using proposed integrated tree height transformation based structural obfuscation and Crypto-steganography approach for securing JPEG processor and DFT processor. (publications: #8, #10, #11, #13)
  - Proposes first work towards securing JPEG codec hardware using double line of defense based on structural obfuscation and crypto-steganography to provide enhanced security.
  - Presents a design flow of generating a secured N-point DFT application specific processor using the proposed double line of defense.
  - The structural obfuscation is employed as 1st line of defense against Trojan insertion and the 2nd line of defense is deployed by embedding proposed crypto-based dual phase hardware steganography.
  - The hardware steganography as a second line of defense uses following security modules/properties to generate a robust stego-mark: substitution using S-box (confusion property), diffusion property, alphabetic encryption, alphabet substitution, byte concatenation and bit-encoding.
- A novel double line of defense methodology using proposed integrated multi-key based structural obfuscation and physical level watermarking approach (publications: #4, #12)
  - Proposes a design methodology for generating secured DSP circuits using double line of defense via key based structural obfuscation as preventive control and physical level watermarking as detective control
  - The obfuscation is performed using key-driven partition and key-driven folding knob based transformations combined with key-driven loop unrolling, key-driven ROE and key-driven THT.
  - The proposed physical-level watermarking is performed through early floorplanning of obfuscated DSP circuits.

- A novel HLS driven secured hardware accelerator design methodology for image processing filters (publication: #5)
    - Proposes a methodology of designing a hardware accelerator architecture using HLS process for 3×3 and 5×5 kernel filters of image processing applications.
    - Designs structurally obfuscated hardware accelerator architecture using high level transformations for both 3×3 and 5×5 filters.
    - Designs structurally obfuscated architectures of five types of 3×3 filters such as blurring, sharpening, vertical embossment, horizontal embossment and Laplace edge detection.
    - Proposes an obfuscated 3×3 filter hardware accelerator design in reconfigurable functionality mode.
- Novel SHA-512 based key generation hardware and anti-removal logic (ARL) hardware based techniques for securing functionally obfuscated DSP cores (publications: #5, #9, #15).
    - Proposes for the first time key-generation logic for ILBs reconfiguration using a custom SHA-512, to be used for larger size designs secured using functional obfuscation.
    - Also proposes key-generation logic using a lightweight ARL unit, to be used for smaller size designs secured using functional obfuscation.
    - Proposed logics are capable to reconfigure larger number of ILBs at lower design overhead than AES128 logic.
- A novel methodology for generating secured DSP cores using proposed robust logic locking (functional obfuscation) approach to thwart IP piracy (publications: #7, #21)
    - Proposes a robust DSP locking cell (DLC) structure to lock the DSP circuits against IP piracy.
    - Proposes re-configurability of DLCs based on AES128 output to enhance the resilience against removal attack.
    - Proposed DLCs render the brute force attack of extracting the actual key ineffective by enabling the true operation of DLCs only upon applying correct key in the first trial.

Chapter 3

Hardware Steganography Techniques for Securing IP Cores against Piracy

For the past few decades, the intellectual property (IP) piracy has posed a serious threat to the security of IP cores. The illegitimate use of IPs not only causes a revenue loss to a true IP vendor but also raises a grave concern about the reputation of the genuine IP vendor. This is because, an adversary may sell the counterfeit/fake components into the market as authentic ones under the brand name of the genuine vendor. Since a counterfeit IP may contain a hidden malicious logic inside and not be fully tested for reliable and safe operations, hence results in sabotaging the true vendor's reputation [1]-[3]. This entails developing a robust mechanism for enabling the detection of IP piracy. The detective control based security of IP cores can be employed during the design process at various levels of design abstraction. However in case of DSP and multimedia applications, the high level synthesis (HLS) process offers an effective and efficient way of integrating the security mechanism. The HLS process has various design phases viz. scheduling, allocation, binding and datapath synthesis, which can be exploited for applying the security. Moreover, the HLS process possesses the flexibility of controlling the design cost overhead and parametric constraints such as area, power and delay that may be affected due to integrating the security mechanism. IP watermarking is a very popular mechanism of detective control over IP piracy, which embeds the IP vendor's signature into the design in the form of hardware security constraints [8]-[11]. The details on IP piracy threat and the state of the art security mechanisms have been discussed in the chapters 1 and 2.

Novel techniques for detecting the piracy using IP core steganography are presented in this chapter. Following two proposed IP core steganography techniques are discussed in this chapter: (i) entropy based hardware steganography (ii) key-driven hash-chaining based hardware steganography. The first section of the chapter formulates the problem. The second section discusses the proposed entropy based IP core steganography technique under the following sub-sections: overview, the proposed steganography methodology with a motivational example, entropy based steganography detection process, demonstration of the entropy based steganography using a DSP application, the measure used for evaluating the security and limitations of entropy based steganography. The Third section discusses the proposed key-driven hash-chaining based IP core steganography methodology under the following sub-sections: overview, the proposed key-driven hash-chaining based steganography methodology with demonstration using a DSP application, steganography detection process, and the measure used for evaluating the security. Subsequently, the fourth section presents the metric for evaluating the impact of proposed steganography techniques on design cost. Finally, the fifth section concludes the chapter.

3.1. Problem Formulation

Given a data flow graph (DFG) representation of a target DSP application, module library, resource constraint Zi, along with the objective of securing IP cores against piracy and false claim of IP ownership, generate a secured stego-embedded design.

3.2. Entropy based Hardware Steganography

The proposed entropy based steganography methodology is discussed under the following sub-sections.

3.2.1. Overview

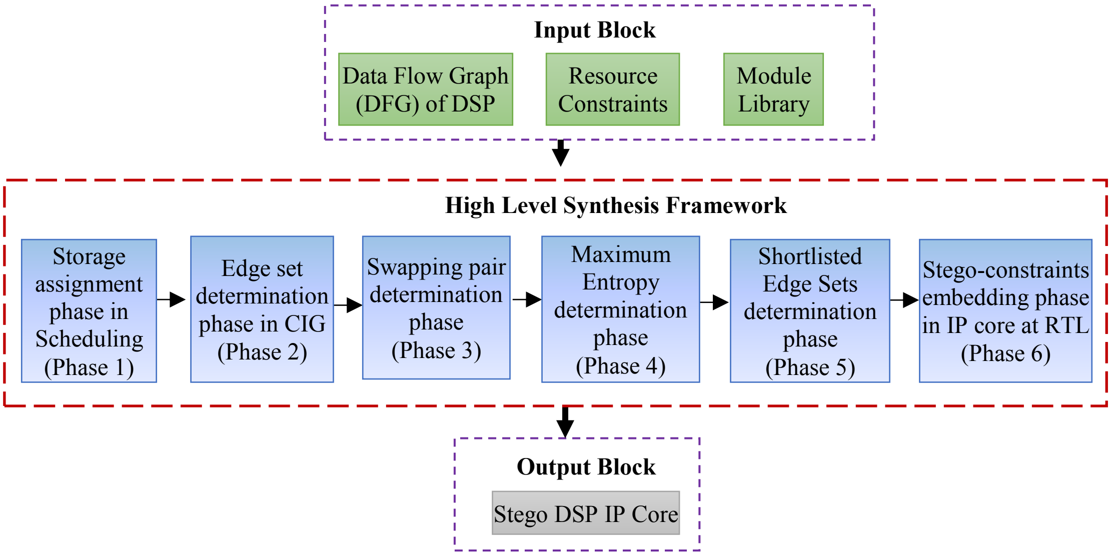
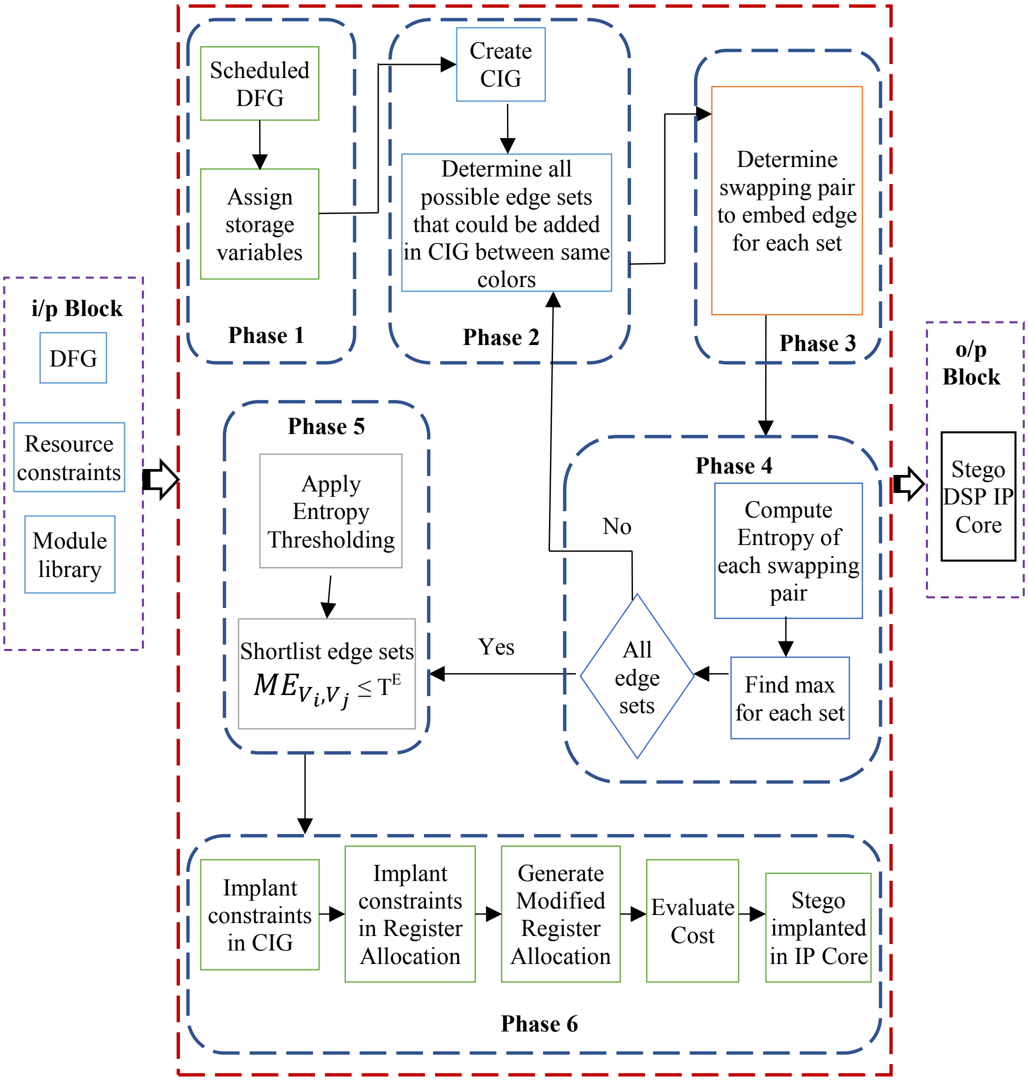

Input Block
Data Flow Graph (DFG) of DSP
Resource Constraints
Module Library
High Level Synthesis Framework
Storage assignment phase in Scheduling (Phase 1)
Edge set determination phase in CIG (Phase 2)
Swapping pair determination phase (Phase 3)
Maximum Entropy determination phase (Phase 4)
Shortlisted Edge Sets determination phase (Phase 5)
Stego-constraints embedding phase in IP core at RTL (Phase 6)
Output Block
Stego DSP IP Core

Fig. 3.1. Overview of the implanting hardware steganography in IP cores

The overview of the proposed steganography framework is shown in Fig. 3.1. As shown, the HLS framework is used to accomplish the process of IP steganography. Input to the HLS framework is a DFG of the intended DSP application and output is the steganography embedded IP core (Stego IP core). Further, the process of embedding steganography using the HLS framework is allocated to six different phases. Phase 1 takes the DFG of the target application as input and accomplishes the storage assignment process post scheduling. Phase 2 first forms a colored interval graph (CIG) [25] where the storage variables are denoted using different nodes in the graph and then performs determination of edge set where all the edges between two nodes of same colors in the CIG are listed. Further, phase 3 takes this edge set as input and performs the determination of swapping pairs for each edge in the set.



i/p Block
DFG
Resource constraints
Module library
Phase 1
Scheduled DFG
Assign storage variables
Phase 2
Create CIG
Determine all possible edge sets that could be added in CIG between same colors
Phase 3
Determine swapping pair to embed edge for each set
Phase 4
Compute Entropy of each swapping pair
Find max for each set
All edge sets
No
Yes
Phase 5
Apply Entropy Thresholding
Shortlist edge sets $ME_{V_i,V_j} \leq T^E$
Phase 6
Implant constraints in CIG
Implant constraints in Register Allocation
Generate Modified Register Allocation
Evaluate Cost
Stego implanted in IP Core
o/p Block
Stego DSP IP Core

Fig. 3.2. Embedding process of hardware steganography for an IP Core

Next, phase 4 determines the maximum entropy for each edge in the set. Then a sub-set of edges is shortlisted in the phase 5 based on the designer's chosen entropy threshold value. In the last, phase 6 embeds the shortlisted edges into the design in the form of constraints edges added to the CIG to generate a stego-IP core.

3.2.2. Elaborating entropy based IP core steganography with a motivational example

The proposed methodology is elaborated in this section using a motivational example. Fig. 3.2 highlights the details of the proposed methodology and its various phases are explained as follows:

Phase 1: Storage variables assignment in the scheduled DFG

The phase 1 takes the DFG of the DSP IP core to be secured as input and performs the scheduling step of HLS based on the resource constraints to generate the corresponding scheduled DFG (SDFG). Further, in this phase, storage variable assignment in the SDFG is performed wherein storage variables are assigned to the inputs and output of each operation. Successively, a colored interval graph (CIG) is created to show the binding of storage variables (nodes) to the minimum number of registers (colors). The storage variable assignment of a sample application is shown in Table 3.1, where V0 to V7 indicate the storage variables and Red (R), Blue (B) and Green (G) indicate the three distinct registers. The corresponding graphical representation in the form of CIG is shown in Fig. 3.3.

Phase 2: Edge set determination in the CIG

Table 3.1 Storage variable assignment of a sample application

Control step	Red	Blue	Green
CS 0	V0	V1	V2
CS 1	V3	V4	V5
CS 2	V6	---	V5
CS 3	V7	---	--

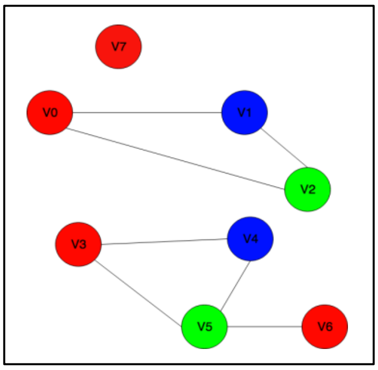


Fig. 3.3. CIG of the sample application

An edge between two nodes in the CIG shows that the life time of two storage variables are overlapping, hence the colors (register assignment) of both nodes in such pairs are distinct. This also indicates that the edges between same color nodes in the CIG can never be drawn. In this phase, all the possible edges (constituting the edge set) that could be added between the nodes of same colors are identified. For the sample application being explained, the set of all possible edges between the nodes of same colors is as follows: D= {<V0, V3>, <V0, V6>, <V0, V7>, <V1, V4>, <V2, V5>, <V3, V6>, <V3, V7>, <V6, V7>}.

Phase 3: Swapping pair determination

The input for phase 3 becomes all the edges mentioned in the set 'D' obtained

Table 3.2 (a): Swapping pairs and corresponding entropies (E_{V_i,V_j}) for the edge (V0, V3)

CS	Swapping pair	Swapping colors (Registers)	(E_{V_i,V_j})
CS0	(V0 ⇔ V1)	(R ⇔ B)	2
CS0	(V0 ⇔ V2)	(R ⇔ G)	2
CS1	(V3 ⇔ V4)	(R ⇔ B)	2
CS1	(V3 ⇔ V5) (V6 ⇔ V5)	(R ⇔ G) + (R ⇔ G)	3
CS2			

Table 3.2 (b): Swapping pairs and corresponding entropies for the edge (V0, V6)

CS	Swapping pair	Swapping colors (Registers)	(E_{V_i,V_j})
CS0	(V0 ⇔ V1)	(R ⇔ B)	2
CS0	(V0 ⇔ V2)	(R ⇔ G)	2
CS2	(V6 ⇒ --)	(R ⇒ B)	1
CS2	(V6 ⇔ V5) + (V3 ⇔ V5)	(R ⇔ G)	3

Table 3.2 (c): Swapping pairs and corresponding entropies for the edge (V0, V7)

CS	Swapping pair	Swapping colors (Registers)	(E_{V_i,V_j})
CS0	(V0 ⇔ V1)	(R ⇔ B)	2
CS0	(V0 ⇔ V2)	(R ⇔ G)	2
CS3	(V7 ⇒ --)	(R ⇒ B)	1
CS3	(V7 ⇒ --)	(R ⇒ G)	1

Table 3.2 (d): Swapping pairs and corresponding entropies for the edge (V1, V4)

CS	Swapping pair	Swapping colors	(E_{V_i,V_j})
CS0	(V1 ⇔ V0)	(B ⇔ R)	2
CS0	(V1 ⇔ V2)	(B ⇔ G)	2
CS1	(V4 ⇔ V3)	(B ⇔ R)	2
CS1 CS2	(V4 ⇔ V5) + (V5 ⇒ --)	(R ⇔ G) + (R ⇒ G)	2

Table 3.2 (e): Swapping pairs and corresponding entropies for the edge (V2, V5)

CS	Swapping pair	Swapping colors	E_{V_i,V_j}
CS0	(V2 ⇔ V0)	(B ⇔ R)	2
CS0	(V2 ⇔ V1)	(G ⇔ B)	2
CS1	(V5 ⇔ V4)	(G ⇔ B)	2
CS1 CS2	(V5 ⇔ V3) + (V5 ⇔ V6)	(G ⇔ R) + (G ⇔ R)	3

Table 3.2 (f): Swapping pairs and corresponding entropies for the edge (V3, V6)

CS	Swapping pair	Swapping colors	E_{V_i,V_j}
CS1	(V3 ⇔ V4)	(R ⇔ B)	2
CS2	(V6 ⇒ --)	(R ⇒ B)	1

Table 3.2 (g): Swapping pairs and corresponding entropies for the edge (V3, V7)

CS	Swapping pair	Swapping colors	E_{V_i,V_j}
CS1	(V3 ⇔ V4)	(R ⇔ B)	2
CS1	(V3 ⇔ V5) + (V5 ⇔ V6)	(R ⇔ G)	3
CS3	(V7 ⇒ --)	(R ⇒ B)	1
CS3	(V7 ⇒ --)	(R ⇒ G)	1

Table 3.2 (h): Swapping pairs and corresponding entropies for the edge (V6, V7)

CS	Swapping pair	Swapping colors	E_{V_i,V_j}
CS2	(V6 ⇒ --)	(R⇒ B)	1
CS2	(V6 ⇔ V5) + (V5 ⇔ V3)	(R ⇔ G)	3
CS3	(V7 ⇒ --)	(R ⇒ B)	1
CS3	(V7 ⇒ --)	(R ⇒ G)	1

in the previous phase. This phase determines the possible swapping pairs in the CIG for enabling the addition of an edge <Vi, Vj> between the node pairs Vi and Vj in the set 'D'. Essentially, the conflict in inserting an edge between node pair Vi and Vj can be resolved using multiple such possible swapping pairs. For the all edges determined in the previous phase, the possible swapping pairs are shown in Table 3.2.

Phase 4: Maximum entropy determination

This phase finds the entropy for each swapping pair of each edge mentioned in the edge set D. The entropy $E_{(Vi,Vj)}$ of a swapping pair of the edge <Vi, Vj> represents the number of color transformations needed to enable embedding of that particular edge in the CIG. Further, for all edges in the set 'D', the maximum value of entropy ($ME_{(Vi,Vj)}$) among all the possible swapping pairs for an edge <Vi, Vj> is determined.

Phase 5: **Eligible edges determination as stego-constraints**

The eligible edges are those edges in the edge set D which qualify for embedding into the CIG as steganography constraints. These edges become eligible based on a vendor specified threshold entropy value (T^E) following the given eligibility criteria:

$$\text{Edge eligibility} = \begin{cases} \text{yes}, & ME_{(Vi,Vj)} \leq T^E \\ \text{no}, & ME_{(Vi,Vj)} > T^E \end{cases} \tag{3.1}$$

For the sample application being discussed in this section, the eligible edges for $T^E = 2$ are as follows: <V0, V7>, <V1, V4>, and <V3, V6>.

Phase 6: Stego-constraints embedding

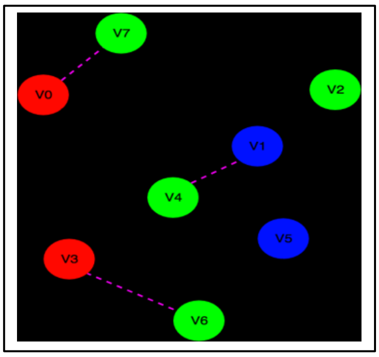


Fig. 3.4. Final CIG after implanting all the edges

Table 3.3 Storage assignment post implanting stego-constraints

CS	Red	Blue	Green
CS 0	V0	V1	V2
CS 1	V3	V5	V4
CS 2	---	V5	V6
CS 3	---	---	V7

In this phase, all the eligible edges are embedded into the design in the form of adding constraint edges to the CIG of the target DSP application. Since an edge cannot be directly added between the nodes of same colors, therefore the proposed algorithm seeks for the corresponding possible swapping pairs to enable the addition of the edge. For example, eligible edges obtained in the previous phase are implanted in the CIG using the following solutions:

(a) The embedding of edge <V0, V7> requires the color transformation of V7 from R to G in the CS3 as shown in Table 3.2(c).

(b) The embedding of edge <V1, V4> requires the color transformations of V4 from R to G and V5 from G to R in the CS1 as shown in Table 3.2(d).

(c) The embedding of edge <V3, V6> requires the color transformation of V6 from R to B in the CS2 as shown in Table 3.2(f).

Using the above solutions, the vendor’s entropy controlled stego-constraints are embedded into the design during the register binding step of the HLS process. Post embedding the constraints, the modified CIG of the sample application and the corresponding modified storage variable assignment are shown in Fig. 3.4 and Table 3.3 respectively. Thus generated a specific register binding of storage variables hides the vendor’s secret stego-information which enables the detection of IP piracy and resolution of IP ownership conflict during the verification process.

3.2.3. Steganography detection

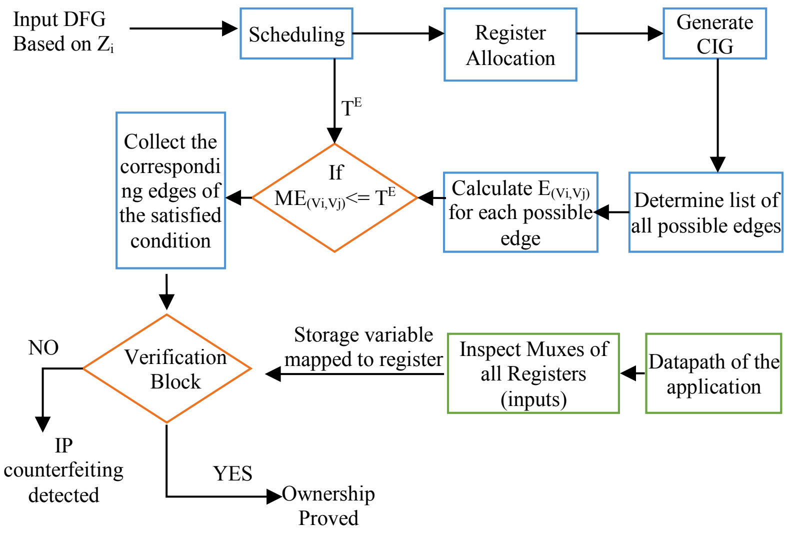

Input DFG
Based on Z_i
Scheduling
Register Allocation
Generate CIG
T^E
Collect the correspondi ng edges of the satisfied condition
If $ME_{(Vi,Vj)} <= T^E$
Calculate $E_{(Vi,Vj)}$ for each possible edge
Determine list of all possible edges
NO
Verification Block
Storage variable mapped to register
Inspect Muxes of all Registers (inputs)
Datapath of the application
IP counterfeiting detected
YES
Ownership Proved

Fig. 3.5. Proposed entropy based steganography detection process

Detection of the stego-information in the DSP IP core is a vital and essential process for resolving IP ownership conflict and detecting piracy. The detail of the proposed steganography detection process is highlighted in Fig. 3.5. As shown, the detection is performed by extracting the hidden information from the design and verifying it with the stego-constraints obtained using the claimant’s steganography process based on the entropy threshold value ‘T^E.

3.2.4. Demonstration of the entropy based steganography using 8-point DCT core

An 8-point DCT is a DSP algorithm which is used in the JPEG compression process to convert the pixel intensities from spatial domain to the frequency domain representation [54]. The scheduled DFG of 8-point DCT based on the resource constraints of 1 adder (A1) and 4 multipliers (M1, M2, M3 and M4) is shown in Fig. 3.6. As shown in the scheduled DFG, total eight control steps (CS) are required to schedule all the operations, and total eight registers viz. Violet, Indigo, Blue, Green, Yellow, Orange, Red, and Black are used to execute 23 storage variables (V0-V22) of the design. Further, a CIG is created from the scheduled DFG and the edge set ‘D’ is determined. The potential

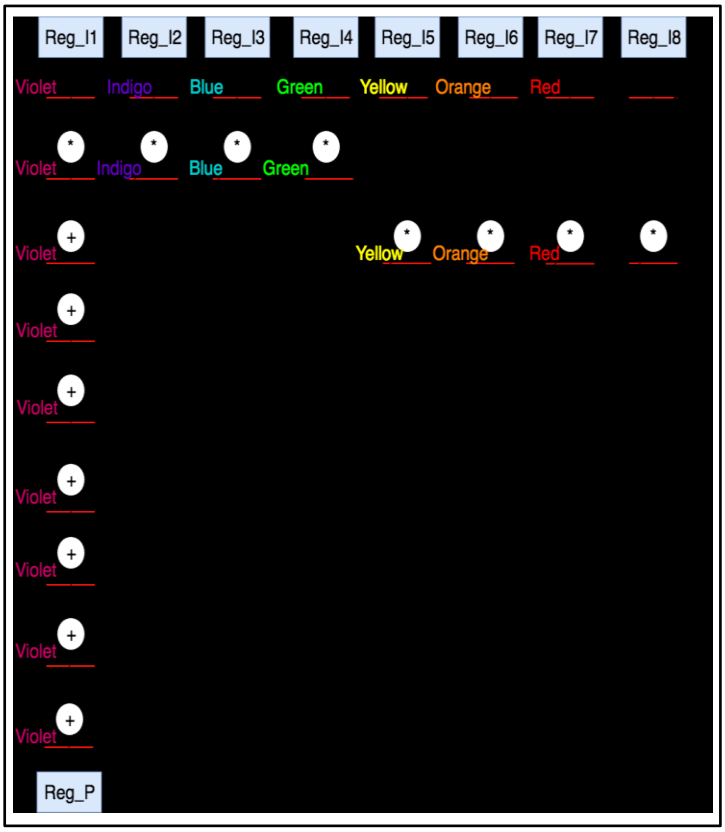


Fig. 3.6. Scheduled and hardware allocated 8-point DCT using 1 (+) and 4 (×)

edges in the set ‘D’ are as follows: <V0, V8>, <V0, V16>, <V0, V17>, <V0, V18>, <V0, V19>, <V0, V20>, <V0, V21>, <V0, V22>, <V1, V9>, <V2, V10>, <V3, V11>, <V4, V12>, <V5, V13>, <V6, V14>, <V7, V15>, <V8, V16>, <V8, V17>, <V8, V18>, <V8, V19>, <V8, V20>, <V8, V21>, <V8, V22>, <V16, V17>, <V16, V18>, <V16, V19>, <V16, V20>, <V16, V21>, <V16, V22>, <V17, V18>, <V17, V19>, <V17, V20>, <V17, V21>, <V17, V22>, <V18, V19>, <V18, V20>, <V18, V21>, <V18, V22>, <V19, V20>, <V19, V21>, <V19, V22>, <V20, V21>,<V20, V22>, < V21, V22>. Further, as per the proposed approach, maximum entropy for each edge in the set is determined. The corresponding maximum entropy values for all edges in the set ‘D’ are 4, 7, 7, 7, 7, 7, 7, 3, 2, 3, 4, 4, 5, 6, 7, 7, 7, 7, 7, 7, 7, 4, 3, 4, 4, 5, 6, 7, 4, 4, 5, 6, 7, 4, 5, 6, 7, 5, 6, 7, 6, 7, 7 respectively. Further, the aforementioned edge eligibility criteria is applied to shortlist the eligible edges based on the entropy threshold value T^E=5. This provides the eligible edges to be embedded into the CIG in the form of stego-constraints. The embedding of stego-constraints leads to a modified register binding of storage variables, indicating the vendor’s secret information hidden into the design. Post embedding stego-constraints, the scheduled DFG of 8-point DCT is shown in Fig. 3.7. Further a stego-embedded RTL datapath is generated after the HLS process, thus

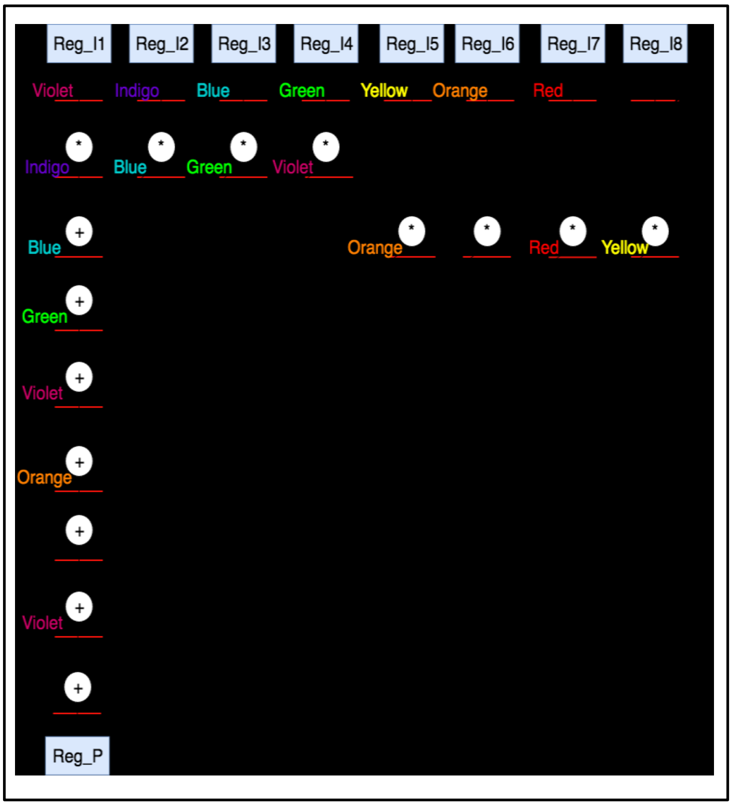


Fig. 3.7. The stego-embedded scheduled and hardware allocated 8-point DCT

enabling the detection of IP piracy.

3.2.5. Metrics for evaluating security of entropy based steganography

The effectiveness of the proposed approach is measured using the security achieved and its impact on design cost. The security is measured in terms of a probability of coincidence (Pc) metric which indicates the probability of coincidently detecting the same stego-information into a design of the same application which is not secured using the proposed steganography approach. Hence, it is expected to be as low as possible. The following formula is used to compute the Pc metric [25], [30]:

$$Pc = \left(1 - \frac{1}{G}\right)^{f} \tag{3.2}$$

Where, G denotes the number of registers before embedding steganography, and f denotes the number of stego constraints added to the CIG. The Pc can be minimized by adding larger number of security constraints into the design, indicating higher strength of the stego-mark.

3.2.6. Limitations of entropy based steganography

The entropy based steganography approach generates the stego-constraints using the secret design data (the initial edge set) and a key-parameter (entropy). If this information is compromised to an adversary, then s/he can regenerate or duplicate the stego-constraints to misuse them for IP piracy or fraudulently claiming the IP ownership. The compromised/copied stego-mark can be misused in a counterfeit design by the adversary to evade the counterfeiting detection process. This entails developing a robust steganography approach which should be arduous to be compromised/ attacked by the adversary.

3.3. Key-driven Hash Chaining based IP Core Steganography

The proposed key-driven hash-chaining based steganography is a robust approach for securing the IP cores against piracy. In this approach, the stego-constraints generation process involves a robust hash-chaining process and a larger size key in order to counter the attacker's malicious intent of copying and misusing the vendor's authentic stego-mark. This gets rid of the limitations of entropy based steganography approach. Further, the stego-constraints are embedded into the design during two distinct phases of HLS, rendering the stego-mark highly strong. The proposed key-driven hash-chaining based steganography methodology is discussed under the following sub-sections.

3.3.1. Overview

This steganography approach generates the stego-constraints using a robust hash-chaining process which is regulated using a larger size vendor's key (stego-key). Once the stego-constraints are obtained, they are imposed onto the target DSP design during register binding phase and functional unit (FU) vendor allocation phases of HLS. Embedding stego-constraints during dual phases of HLS enhances the quality of vendor's stego-mark hidden into the design. Fig. 3.8 shows the overview of the proposed dual-phase key-driven hash chaining based steganography approach. The inputs and output of the proposed approach are highlighted in the figure itself. The generic flow of the proposed approach is as follows: (i) performing the scheduling, allocation and binding steps on DFG of the target DSP application using the module library and resource constraints, (ii) generating various encoded bitstreams of the DSP application using proposed encoding rules (ii) performing key-driven

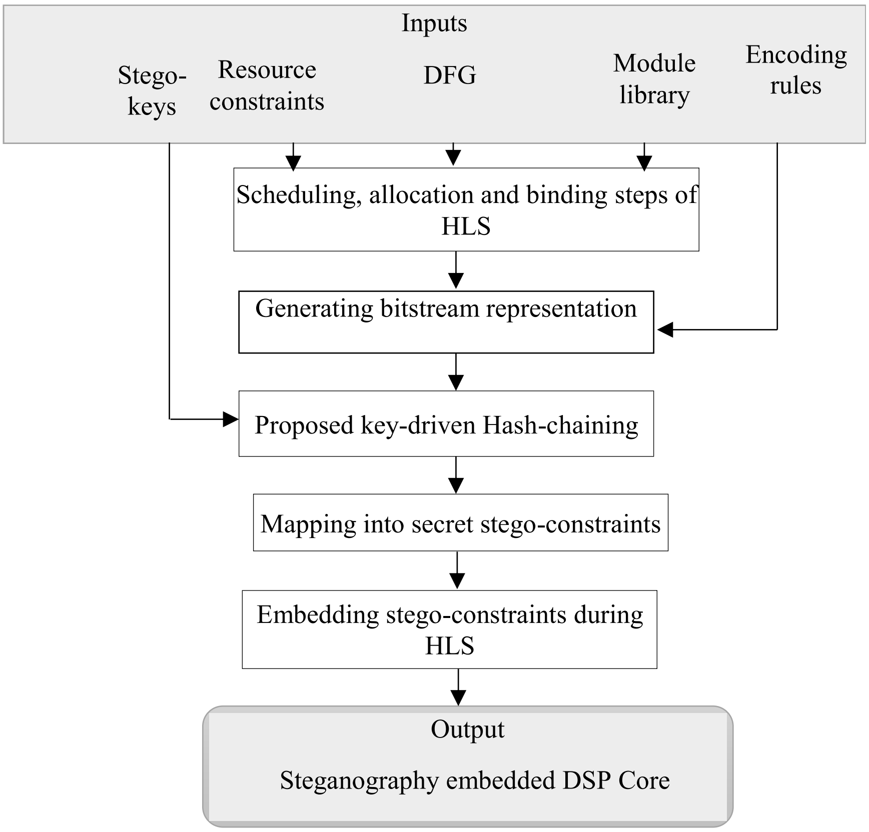

Inputs
Stego-keys
Resource constraints
DFG
Module library
Encoding rules
Scheduling, allocation and binding steps of HLS
Generating bitstream representation
Proposed key-driven Hash-chaining
Mapping into secret stego-constraints
Embedding stego-constraints during HLS
Output
Steganography embedded DSP Core

Fig. 3.8. Overview of key-driven hash-chaining based steganography

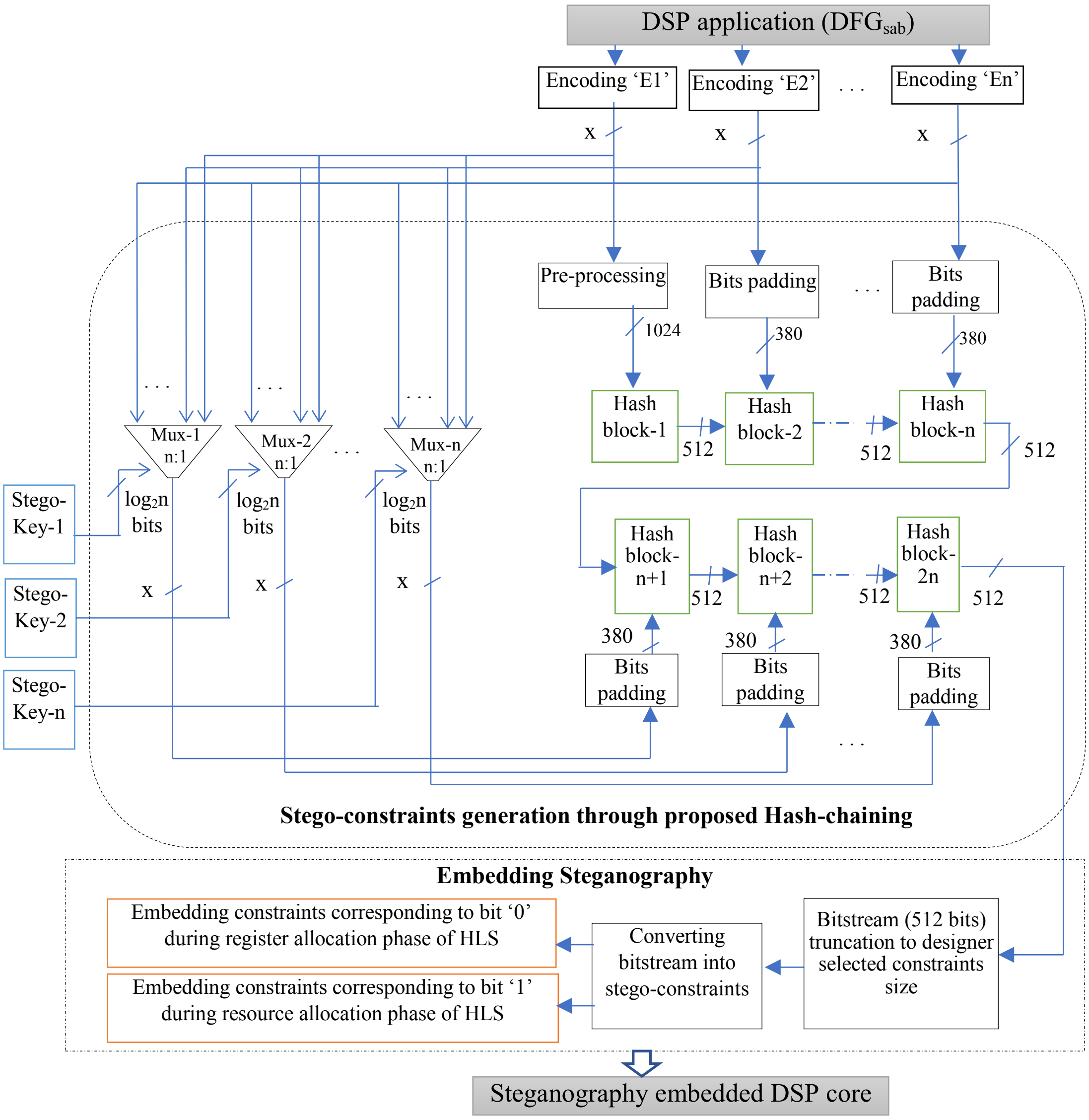

DSP application (DFG$_{sab}$)
Encoding 'E1'
Encoding 'E2'
Encoding 'En'
x
x
x
Pre-processing
Bits padding
Bits padding
1024
380
380
Hash block-1
Hash block-2
Hash block-n
512
512
512
Mux-1 n:1
Mux-2 n:1
Mux-n n:1
log$_2$n bits
log$_2$n bits
log$_2$n bits
Stego-Key-1
Stego-Key-2
Stego-Key-n
x
x
x
Hash block-n+1
Hash block-n+2
Hash block-2n
512
512
512
380
380
380
Bits padding
Bits padding
Bits padding
Stego-constraints generation through proposed Hash-chaining
Embedding Steganography
Embedding constraints corresponding to bit '0' during register allocation phase of HLS
Embedding constraints corresponding to bit '1' during resource allocation phase of HLS
Converting bitstream into stego-constraints
Bitstream (512 bits) truncation to designer selected constraints size
Steganography embedded DSP core

Fig. 3. 9. Details of proposed key-driven hash-chaining based steganography

hash-chaining process taking the encoded bitstreams as inputs (ii) mapping the output bitstream of hash-chaining process to stego-constraints using the mapping rules (iii) embedding the secret stego-constraints into the design during dual phases (register binding phase and FU vendor allocation phase) of HLS. The embedded stego-constraints into the design act as a vendor's secret digital evidence to proving the authenticity of the genuine IP and identifying the fake ones.

3.3.2. Details of key-driven hash chaining based IP core steganography with demonstration using 8-point DCT core

Figure 3.9 shows the details of the proposed methodology and it is elaborated with a demonstration on 8-point DCT core under the following steps.

(a) Scheduling, allocation and binding of input DFG

This step performs scheduling, allocation and binding of DFG of the input DSP application using the resource constraints of one adder and four multipliers and the given module library. For example, Fig. 3.10 shows the DFG post performing scheduling, allocation and binding phases of HLS. Here, the vendor allocation to FUs has been performed using the two-vendor allocation scheme in which an FU instance F_b^a (e.g. $M_b^a\ or\ A_b^a$) bears the vendor type ‘a’ and instance number ‘b’. In the two vendor allocation scheme, the variable ‘a’ can take up only two possible values either 1 or 2. Further, in the scheduled, allocated and binded DFG (abbreviated as DFG$_{\text{sab}}$), V0 to V22 are the storage variables, P, I, V, G, Y, O, R and B are the eight distinct registers as shown in Fig. 3.10. Thus obtained DFG$_{\text{sab}}$ is used for generating various encoded bitstreams.

(b) Generating encoded bitstreams

The DFG$_{\text{sab}}$ is encoded into various bitstream representations using the vendor’s encoding rules. The encoding rules encode each operation of DFG$_{\text{sab}}$ into either bit ‘0’ or ‘1’, hence the length of the encoded bitstream is same as the number of operations (nodes) in the application (DFG). Some proposed

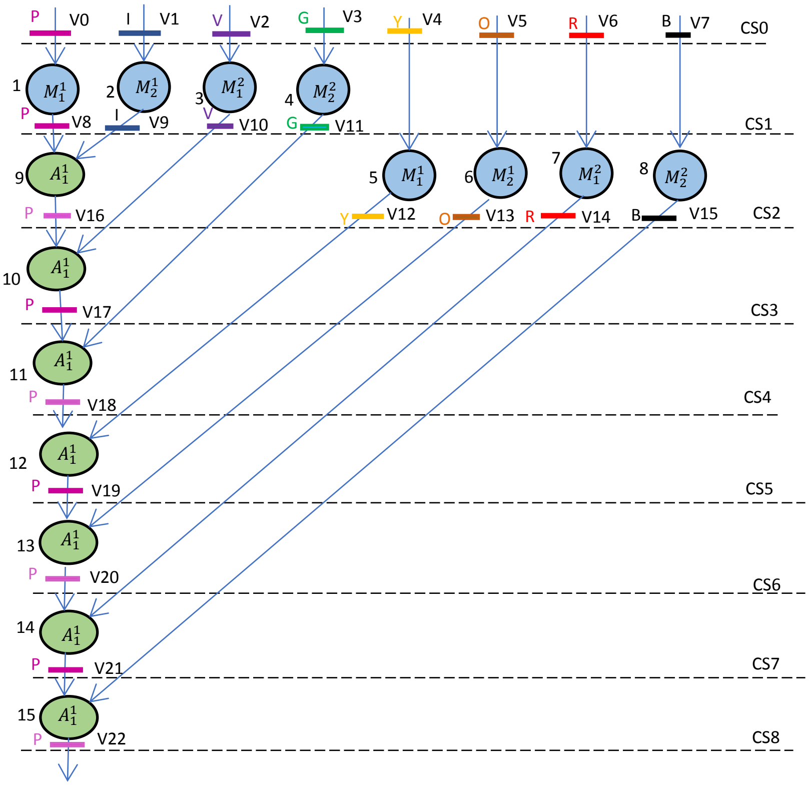


Fig. 3.10. DFG$_{\text{sab}}$ of 8-point DCT using 1 A and 4 M before implanting steganography

encoding rules are as follows:

1. *Encoding rule E1*: If operation number (opn#) and respective CS# are both even then the opn is encoded as bit ‘0’ otherwise bit ‘1’.
2. *Encoding rule E2*: If operation number and respective CS number both bear the same parity (both either even or odd) then the opn is encoded as bit ‘0’. However, if both bear the different parity then the opn is encoded as bit ‘1’.
3. *Encoding rule E3*: If operation number and respective CS# are both odd then the opn is encoded as bit ‘0’ otherwise bit ‘1’.
4. *Encoding rule E4*: If operation number and respective CS number both bear the different parity then the opn is encoded as bit ‘0’. However, if both bear the same parity then the opn is encoded as bit ‘1’.
5. *Encoding rule E5*: If operation number and respective CS# are both prime then the opn is encoded as bit ‘0’ otherwise bit ‘1’.
6. *Encoding rule E6*: If operation number and respective CS# are both prime then the opn is encoded as bit ‘1’ otherwise bit ‘0’.
7. *Encoding rule E7*: If the greatest common divisor of operation number and the respective CS number is one then the opn is encoded as bit ‘0’. However, if the greatest common divisor of operation number and the respective CS number is not one then the opn is encoded as bit ‘1’
8. *Encoding rule E8*: If the operation number modulo respective CS number is zero then the opn is encoded as bit ‘0’. However, if the operation number modulo respective CS number is not zero then the opn is encoded as bit ‘1’.
9. *Encoding rule E9*: If the CS number is equal to second odd sequence of operation number then the opn is encoded as bit ‘0’ otherwise bit ‘1’.

For demonstration, the encoded bitstream representations of DFG_{sab} of 8-point DCT (shown in Fig 3.10) for the aforementioned nine encoding rules are respectively as follows:

“E1: 111110101111111”, “E2: 010110101111111”, “E3: 010111111111111”, “E4: 101001010000000”, “E5: 111101011111111”, “E6: 000010100000000”, “E7: 000001010000010”, “E8: 000010101111101”, “E9: 111111111111111”

However, an application having x operations can have 2^x possible encoded bitstream presentations using the same number of encoding rules.

(c) Generating a hashed bitstream using key-driven hash-chaining process

The encoded bitstreams of the DSP application are fed to the key-driven hash chaining process to generate a hashed bitstream. If there are n number of encoded bitstreams then 2×n hash blocks are used in the hash chain. Each hash block performs a SHA-512 algorithm on 1024-bit input data where the 1024-bit data is formed using the following procedure:

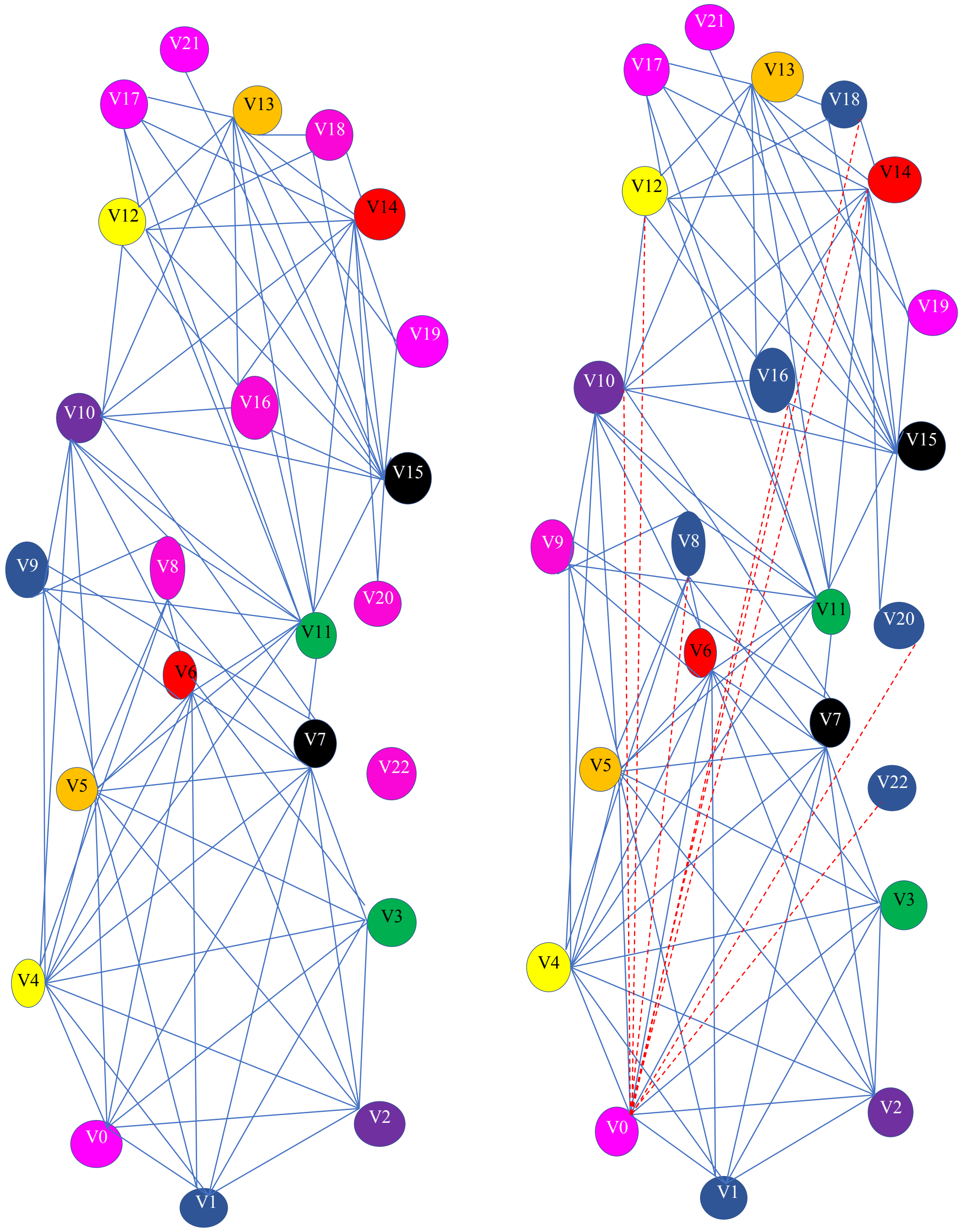


Fig. 3.11 (a). CIG of 8-point DCT before steganography

Fig. 3.11 (b). CIG of 8-point DCT after steganography

For the 1st hash block in the hash chain: the encoded bitstream of length x-bit is first appended with '1' followed by sequence of '0' bits to form a 896-bit chunk. Further, the 128-bit representation of the length 'x' of encoded bitstream is appended to the 896-bit chunk to form 1024-bit input to the 1st hash block.

For the remaining hash blocks in the chain: the 512-bit output of previous hash-block is concatenated with a 4-bit chunk "1000" followed by 380-bit output of bits-padding block and 128-bit representation of the length '512 bits' of previous hash, to form 1024-bit input to the remaining hash blocks. The 380-bit output of bits-padding block is generated by padding designer's chosen (380-x) bits before the x-bit long encoded bitstream.

Further, in the hash-chaining process, the ith hash block uses the ith bitstream where 'i' varies from 1 to n (the number of encoded bitstreams). However, the remaining n- hash blocks are key driven where the encoded bitstreams used by a key-driven hash block is determined by the stego-key value of size $\lceil \log_2 n \rceil$ bits. The total stego-key size is computed to be is $n \times \lceil \log_2 n \rceil$ bits as there are n number of key-driven hash blocks in the hash-chain. The output of final hash-block is 512-bit hashed bitstream.

For the 8-point DCT application being demonstrated, nine (n=9) encoding rules are used to generate the 9 encoded bitstreams which are processed by 2n=18 hash blocks through the hash-chain process. The hash block number 10 to 18 are the key driven hash blocks which use the following keys respectively: "1000, 0111, 0110, 0101, 0100, 0011, 0010, 0001, 0000".

(d) Mapping the hashed bitstream into stego-constraints

The 512-bit long hashed bitstream is first truncated to the designer selected size of stego-constraints. For the demonstration on DCT core, let's say the chosen constraints size is 24 bits and the truncated bitstream is as follows: "111100010011101000011000" which contains 13 zeros and 11 ones. The bits of the truncated hashed bitstream are mapped to the stego-constraints using the following mapping rules:

'0': An edge added between node pair (even, even) of the CIG

'1': Odd operations are assigned to FU of vendor type 1 and even operations are assigned to FU of vendor type 2.

Where, the CIG corresponding to the DFG$_{sab}$ of a DSP application graphically represents the register binding of storage variables.

For the 8-point DCT application, the corresponding CIG is shown in Fig. 3.11(a) which is exploited to embed secret stego-constraints into the design.

(e) Embedding stego-constraints into the design

The obtained stego-constraints are embedded into the design in the form of extra edges added to the CIG and a specific FU vendor allocation to the respective operation. The stego-constraints corresponding to bit '0' and bit '1' are implanted in the design during register binding and FU vendor allocation phase respectively of HLS. The constraint edges added to the CIG corresponding to the 13 zeros (obtained in the previous step) are as follows: <V0,V2>, <V0,V4>, <V0,V6>, <V0,V8>, <V0,V10>, <V0,V12>, <V0,V14>, <V0,V16>, <V0,V18>, <V0,V20>, <V0,V22>, <V2,V4>, <V2,V6>. These edges are deliberately added to the CIG one by one, indicating the vendor's stego-information embedded into the design during register binding phase of HLS. Sometimes, adding extra edges may result into conflict as no edge can exist between two nodes of similar color. This conflict is resolved through local valid alterations

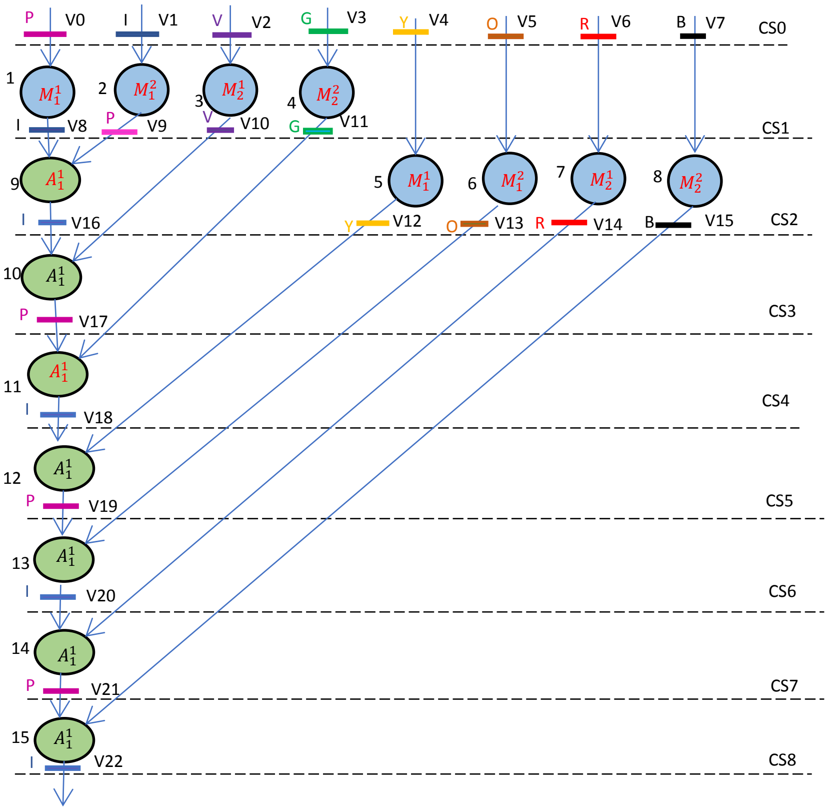


Fig. 3.12. Scheduled and hardware allocated 8-point DCT after implanting steganography

in the colors of the nodes. However, in some cases, additional colors (registers) may be needed to accommodate the constraints. The CIG post embedding stego-constraints corresponding to bit 0 are shown in Fig. 311(b). Further, the constraints corresponding to bit '1' specify the constraint-based allocation of FU vendor type to the respective operations in the DFG_{sab}. Post embedding all the constraints, the DFG_{sab} is shown in Fig. 3.12. Thus obtained stego-DFG_{sab} is transformed into the stego-RTL by performing datapath and controller synthesis phases of HLS. The stego-RTL is a secured soft IP core of the respective DSP application, enabling the detection of IP piracy.

In the proposed steganography approach, the amount of digital evidence hidden into the design is higher because of inserting constraints during two distinct phases of HLS process. This results in a stronger stego-mark leading to the robust detection of IP piracy.

3.3.3. Steganography detection

The detail of the key-driven hash chaining based steganography detection process is highlighted in Fig. 3.13. As shown, the detection is performed by

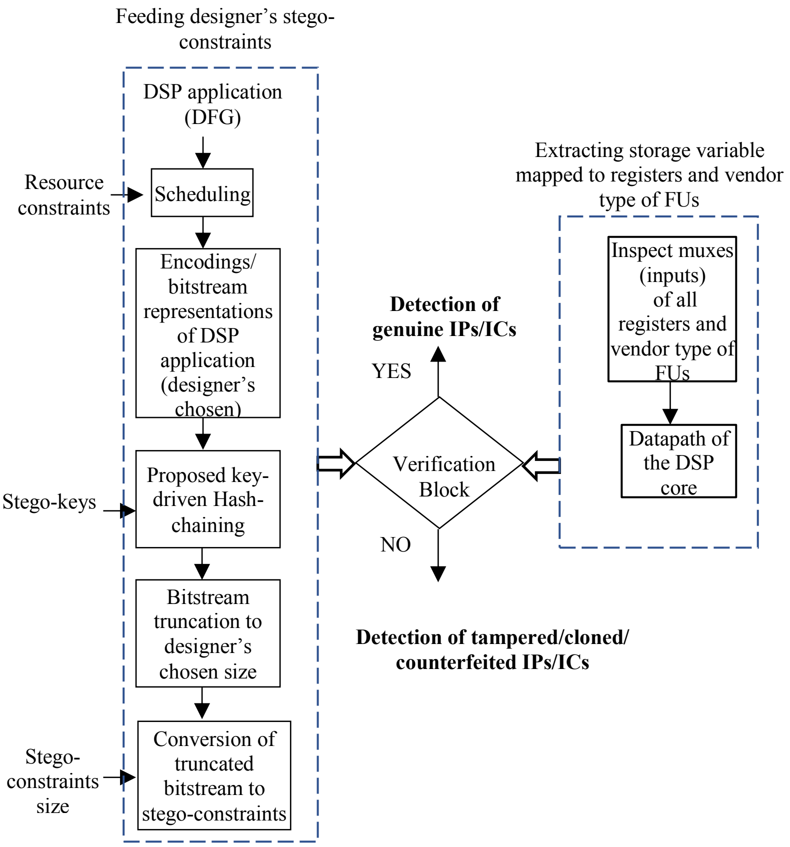

Feeding designer's stego-constraints
DSP application (DFG)
Resource constraints
Scheduling
Encodings/ bitstream representations of DSP application (designer's chosen)
Stego-keys
Proposed key-driven Hash-chaining
Bitstream truncation to designer's chosen size
Stego-constraints size
Conversion of truncated bitstream to stego-constraints
Detection of genuine IPs/ICs
YES
Verification Block
NO
Detection of tampered/cloned/ counterfeited IPs/ICs
Extracting storage variable mapped to registers and vendor type of FUs
Inspect muxes (inputs) of all registers and vendor type of FUs
Datapath of the DSP core

Fig. 3.13. Detection process of key-driven hash chaining based steganography

extracting the hidden information from the design and verifying it with the stego-constraints obtained using the key-driven hash chaining process.

3.3.4. Metrics for evaluating security of key-driven hash-chaining based steganography

Security of key-driven hash chaining based steganography is evaluated using the following metrics: (i) probability of coincidence (Pc) metric (ii) maximum possible size of stego-key (iii) the attacker's maximum effort of decoding the valid stego-key (iv) attacker's effort in terms of finding encoded bits (v) attacker's total effort in determining the stego-constraints.

(i) *Probability of coincidence of key-driven hash-chaining based steganography*: This metric is measured using the following formula:

$$\mathrm{Pc} = \left(1 - \frac{1}{\mathrm{G}}\right)^{\mathrm{f1}} \times \left(1 - \frac{1}{\pi_{\mathrm{i}=1}^{\mathrm{y}} \mathrm{N(Zi)}}\right)^{\mathrm{f2}} \tag{3.3}$$

Where, the first term in the equation corresponds to the Pc due to embedding constraints in the register binding phase and the second term corresponds to the Pc due to embedding constraints in the FU vendor allocation phase. Here, in the first term, G denotes the number of registers before embedding steganography, and f1 denotes the number of stego constraints added to the CIG. Further, in the second term, 'y' denotes the types of resources in the DSP application, N(Zi) denotes the number of instances of each FU type and f2 denotes the number of stego constraints added during the FU vendor allocation phase.

(ii) *Maximum possible size of stego-key*: The maximum possible size of the stego-key (key_m^s) is computed as follows:

$$\mathrm{key}_{\mathrm{m}}^{\mathrm{s}} = \mathrm{U} \times \lceil \log_2 \mathrm{U} \rceil \text{ bits} \tag{3.4}$$

Where, 'U' denotes the total possible encodings for a DSP application. The value of U is computed to be 2^x, where 'x' is the number of operations in the DSP application. The U=2^x also represents the total number of key-driven hash blocks that can be chained in the hash-chaining process. Further, $\lceil \log_2 U \rceil$ is the size of the stego-key used to drive each key-driven hash block.

(iii) *Attacker's maximum effort of decoding the valid stego-key*: This metric is given by the following equation:

$$\mathrm{A}_{\mathrm{m}}^{\mathrm{sk}} = 2^{\mathrm{U}\times\lceil\log_2\mathrm{U}\rceil} \tag{3.5}$$

Where, $\mathrm{A}_{\mathrm{m}}^{\mathrm{sk}}$ denotes attacker's max effort in terms of deducing the stego-key.

(iv) *Attacker's effort in terms of finding encoded bits*: This metric is given by the following equation:

$$\mathrm{A}_{\mathrm{m}}^{\mathrm{eb}} = 2^{380\times 2\mathrm{n}} \tag{3.6}$$

Where, $\mathrm{A}_{\mathrm{m}}^{\mathrm{eb}}$ denotes the attacker's effort in terms of finding the encoded bits, 380 bits is the output of a padding block and 2n is the total number of hash blocks in the hash-chaining process.

(v) *Attacker's total effort in determining the stego-constraints:* This metric is given as follows using eq. (3.5) and eq. (3.6):

$$\mathrm{A}_{\mathrm{T}}^{\mathrm{s}} = \mathrm{A}_{\mathrm{m}}^{\mathrm{sk}} \times \mathrm{A}_{\mathrm{m}}^{\mathrm{eb}}$$
$$\mathrm{A}_{\mathrm{T}}^{\mathrm{s}} = 2^{\mathrm{U}\times\lceil\log_2\mathrm{U}\rceil} \times 2^{380\times 2\mathrm{n}}$$
$$\mathrm{A}_{\mathrm{T}}^{\mathrm{s}} = 2^{(\mathrm{U}\times\lceil\log_2\mathrm{U}\rceil+(380\times 2\mathrm{n}))} \tag{3.7}$$

Where, A_T^S denotes the attacker's total effort in determining the stego-constraints imposed onto the intended design.

3.4. Metric for Evaluating Impact of Proposed Steganography Techniques on Design Cost

Embedding stego-constraints to the designs may impact the design cost. This is because, adding the constraints may require additional hardware or control steps to accommodate them. Therefore, to evaluate the feasibility of the proposed steganography approach, the design cost post embedding stego-constraints is required to be computed. The following function is used to compute the design cost:

$$C_f(Zi) = \beta_1 \frac{L_T}{L_{max}} + \beta_2 \frac{A_T}{A_{max}} \tag{3.8}$$

Where, $\mathrm{C_f(Zi)}$ is the cost of the design scheduled using the resource constraints Zi, further, $\mathrm{L_T}$ and $\mathrm{A_T}$ denote the design delay and area at the given resource constraints, $\mathrm{A_{max}}$ and $\mathrm{L_{max}}$ denote the maximum design area and delay within the possible design space, β_1 and β_2 denote the weights which are kept to be 0.5 to assign equal preference to both delay and area.

3.5. Summary

The IP core piracy threat was handled using the proposed IP core steganography methodologies. The proposed entropy based steganography approach offers the flexibility of controlling the amount of stego-information to be inserted into the design using an entropy threshold value. The IP designer/vendor can vary the entropy threshold from low to high value to achieve the higher strength of the stego-mark. Additionally, the proposed approach is measured in terms of its security and design cost to evaluate its effectiveness. Further, the key-driven hash-chaining based hardware steganography approach was discussed in this chapter. This approach involves vendor's large size stego-key and a robust hash-chaining process to generate the stego-constraints. This approach overcomes the potential threat of evading IP counterfeiting due to a copied stego-mark in the fake designs. Achieving a robust security against IP piracy, while incurring negligible cost overhead, is the strength of the proposed steganography approaches. The experimental results of the entropy based steganography and key-driven hash chaining based steganography have deeply been analyzed in the chapter 10 of this thesis.

Chapter 4

Hologram based Structural Obfuscation to Thwart Reverse Engineering based Attacks

Reverse engineering (RE) is a process of extracting a desired higher level of abstraction from a given lower level of abstraction of an intended design by analyzing its internal details. It can be performed at following various levels: (i) to extract the layout of the design from the IC through de-packaging, delayering, imaging and pattern recognition technique (ii) to extract the gate level netlist from the layout of the design through analyzing transistors interconnectivity and matching it against the standard cells (logic gates) in the library (iii) to extract high level functionality (e.g. datapath and controller) from a given gate level netlist by partitioning it into sub-circuits and then matching them against the RTL components in the module library. RE can be legally performed for the aim of teaching, analyzing, or evaluating the ideas or methods applied in the intended circuitry. This is supported by the Semiconductor Chip Protection Act, USA. However, reverse engineering has a darker side as well. An adversary in an untrustworthy design house or foundry may perform the RE to realize his/her ill intentions of stealing the design intents or inserting a malicious logic (hardware Trojan) inside the design. More explicitly, the RE can result into IP piracy and Trojan attack which may not only harm the vendor's or designer's revenue but also ruin his/her credibility. The RE based attack can be thwarted by making RE as hard and time consuming as possible. Considering the today's attacker competence, the absolute security against RE cannot be deployed. However, rendering the illegal actions by the attacker/adversary highly expensive through RE is considered as enough security. Towards the protection against RE by a potential adversary, structural obfuscation is a technique of internal architecture concealment which makes the design structure unobvious to be interpreted by the adversary hence hindering the RE process [1]-[4].

This chapter presents a novel structural obfuscation technique based on the security image hologram feature to secure the DSP circuits against the RE based attack of IP piracy and potential hardware Trojan insertion. The first section formulates the problem. The second section discusses the hologram based structural obfuscation approach under the following sub-sections: overview, elaborating hologram based obfuscation technique and metric for evaluating the security achieved. Next, the third section demonstrates the generation of hologram obfuscated design. Further, the fourth section highlights the similarity between the hologram obfuscated design and a security image hologram. Finally, the fifth section summarizes the chapter.

4.1. Problem Formulation

Given the data flow graph (DFG) representations of two target DSP applications, module library, resource constraint, along with the objective of securing IP cores against RE based attacks, generate a secured (structurally obfuscated) integrated RTL design of DSP cores.

4.2. Hologram based Structural Obfuscation Approach

This section discusses the proposed hologram based structural obfuscation approach under the following sub-sections:

4.2.1. Overview

The overview of the hologram based structural obfuscation technique is depicted in Fig. 4.1. This obfuscation technique takes inputs in the form of

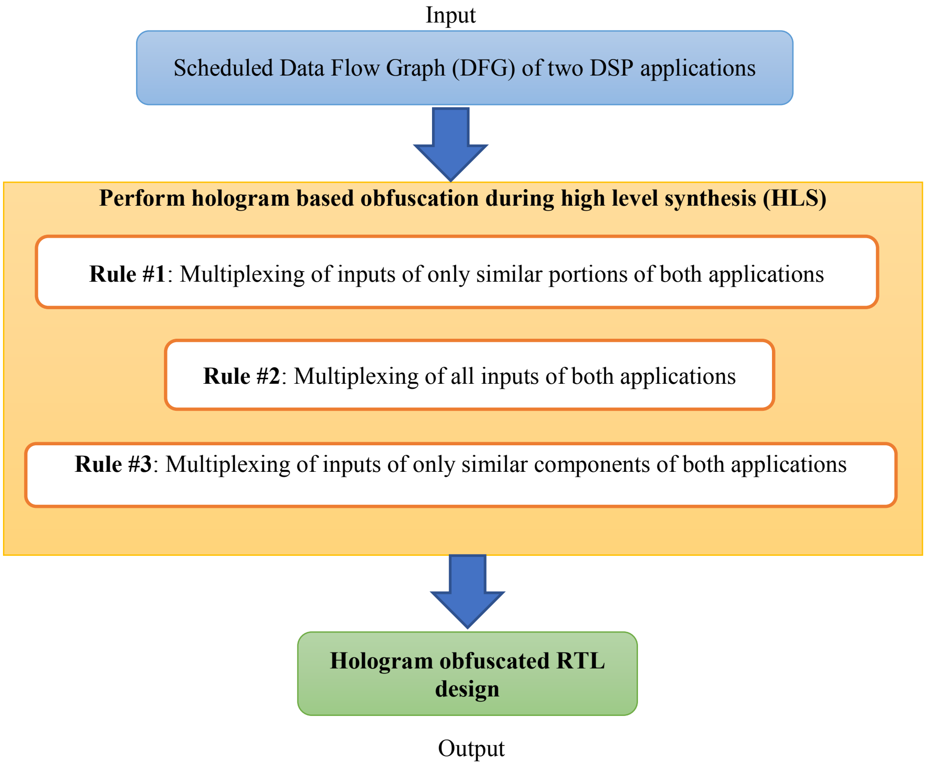

Input
Scheduled Data Flow Graph (DFG) of two DSP applications
Perform hologram based obfuscation during high level synthesis (HLS)
Rule #1: Multiplexing of inputs of only similar portions of both applications
Rule #2: Multiplexing of all inputs of both applications
Rule #3: Multiplexing of inputs of only similar components of both applications
Hologram obfuscated RTL design
Output

Fig. 4.1. Overview of Hologram based obfuscation approach

data flow graph (DFG) of two DSP applications and generates a hologram obfuscated design at the output. Thus generated obfuscated design is a camouflaged integrated RTL datapath of the input DSP applications. To perform the hologram based obfuscation technique, following three rules (highlighted in Fig. 4.1) are applied:

(i) *The first rule*: only a sub-set of inputs of two DSP applications are subjected to multiplexing to generate the obfuscated datapath.

(ii) *The second rule*: all inputs of two DSP applications are subjected to multiplexing to generate the obfuscated datapath.

(iii) *The third rule*: only inputs of similar operations of both DSP applications are subjected to multiplexing.

To produce a hologram obfuscated design, the application of the above mentioned rules depends on the level of similarity of two intended DSP applications.

4.2.2. Elaborating hologram based obfuscation technique

This section elaborates the proposed approach of generating a hologram based obfuscated design. Fig. 4.2 shows the details of the process of deploying hologram based structural obfuscation which employs three different rules during the datapath synthesis phase of the HLS process. As shown in the figure, the DFG forms of two DSP applications (DSP-1 and DSP-2) are first subjected to pre-synthesis phase of HLS where scheduling, allocation and binding are performed. The pre-synthesis phase produces scheduled, allocated and binded DFG (DFG_{sab}) of two DSP applications. Further, DFG_{sab}-1 and DFG_{sab}-2 are subjected to datapath synthesis phase of HLS where proposed hologram based obfuscation is applied to generate a structurally obfuscated integrated RTL datapath of DSP-1 and DSP-2. The elaboration of proposed rules for hologram based obfuscation is as follows:

(i) The first rule: multiplexing sub-set of inputs of two DSP applications:

If a portion of DFG_{sab}-1 of DSP-1 matches with the DFG_{sab}-2 of DSP-2 (the matched portions of DSP-1 and DSP-2 contain similar operations with similar

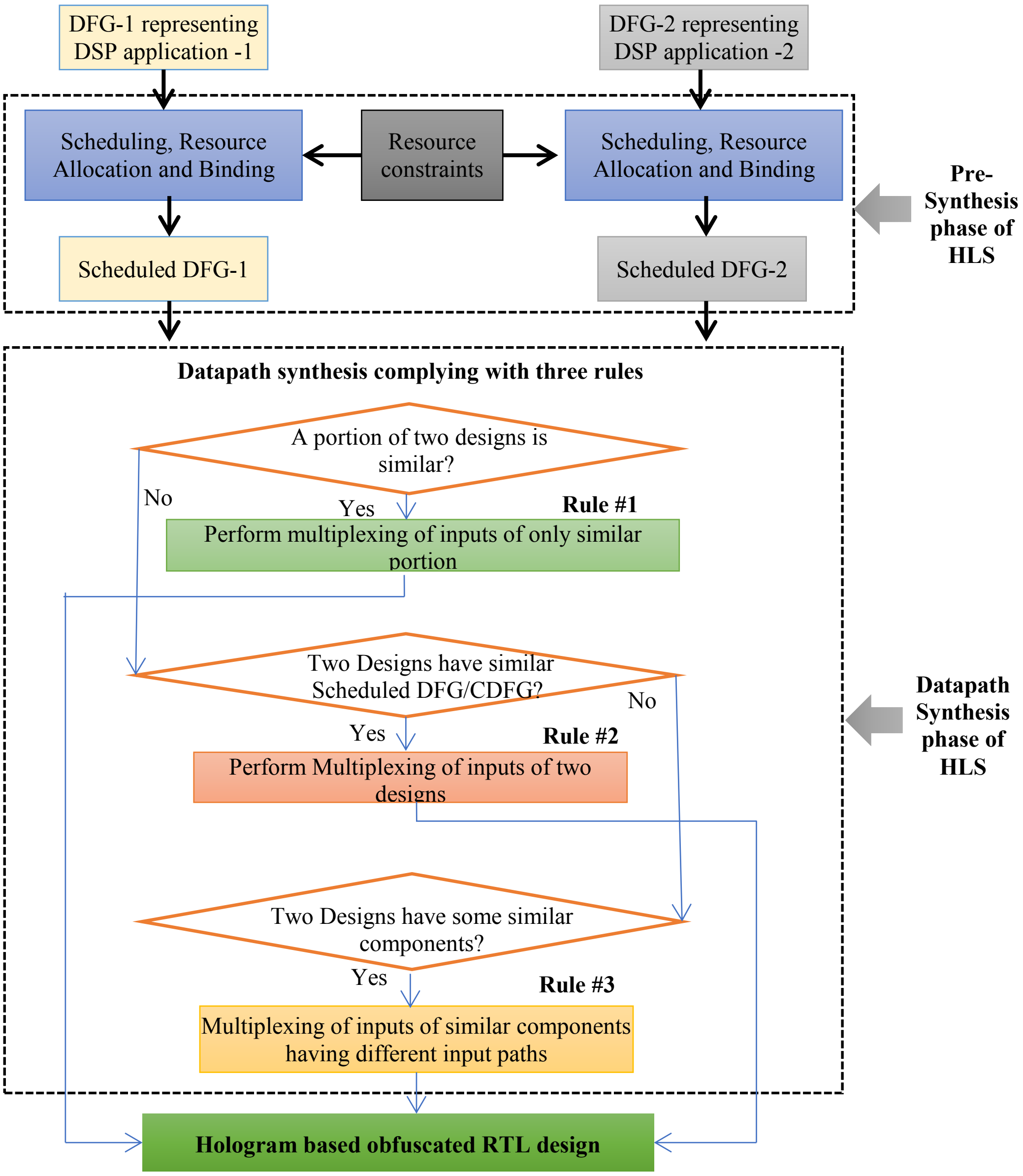

DFG-1 representing DSP application -1
DFG-2 representing DSP application -2
Scheduling, Resource Allocation and Binding
Resource constraints
Scheduling, Resource Allocation and Binding
Scheduled DFG-1
Scheduled DFG-2
Pre-Synthesis phase of HLS
Datapath synthesis complying with three rules
A portion of two designs is similar?
No
Yes
Rule #1
Perform multiplexing of inputs of only similar portion
Two Designs have similar Scheduled DFG/CDFG?
No
Yes
Rule #2
Perform Multiplexing of inputs of two designs
Datapath Synthesis phase of HLS
Two Designs have some similar components?
Yes
Rule #3
Multiplexing of inputs of similar components having different input paths
Hologram based obfuscated RTL design

Fig. 4.2. Flow of generating a Hologram obfuscated DSP design

input-output connectivity), then the datapath synthesis phase undergoes following actions:

1. The inputs of only matched portions of DSP-1 and DSP-2 applications are multiplexed using multiplexers (Muxes) in the integrated datapath.
2. The outputs of DSP-1 and DSP-2 applications are also multiplexed using a multiplexer in the integrated datapath.

In the obfuscated integrated datapath, switching between two designs is executed using multiplexers of size 2×1. And, the switching is regulated using a designer control input ‘C’ which acts as a select line for the switching

Muxes. At control input ‘C’=0, one DSP core becomes functional by taking inputs through switching Muxes whereas at ‘C’=1, another DSP core becomes functional in the obfuscated integrated datapath. This rule of hologram based obfuscation is demonstrated in section 4.3 by selecting finite impulse response (FIR) filter application as DSP-1 and infinite impulse response (IIR) filter application as DSP-2.

(ii) The second rule: multiplexing of all inputs of two DSP applications:

If the DFG_{sab}-1 of DSP-1 fully matches with the DFG_{sab}-2 of DSP-2 (i.e. possessing identical number of inputs and outputs and same operations with the same input-output connectivity), then the datapath synthesis phase undergoes following action:

1. All inputs of DSP-1 and DSP-2 are multiplexed using Muxes in the integrated datapath.

At control input ‘C’=0, one DSP core becomes functionally active whereas at ‘C’=1, another DSP core becomes functionally active.

(iii) The third rule: multiplexing of the inputs of same operations in two DSP applications:

The applicability of this rule depends on the presence of similar operations (nodes) in the DFG_{sab}-1 of DSP-1 and DFG_{sab}-2 of DSP-2 with different inputs and output. If some operations are similar in two DSP applications then the datapath synthesis phase undergoes following action:

1. The inputs of the functional modules corresponding to the similar operations of two DSP applications are multiplexed using Muxes in the integrated datapath. The functional modules of DSP-1 get inputs when the control signal ‘C’ is =0 whereas the functional modules of DSP-2 get inputs when the control signal ‘C’ is =1. In this fashion, functionality of one of the application is activated at a time using the control input ‘C’.

Post applying these rules of hologram based obfuscation on DFG_{sab}-1 of DSP-1 and DFG_{sab}-2 of DSP-2 during the datapath synthesis phase of HLS framework, a structurally obfuscated integrated RTL design is produced. The structurally obfuscated RTL design acts as a common datapath for both DSP cores. The hologram based obfuscation approach enables the camouflaging of two DSP cores into a single RTL datapath to ensure the internal architecture concealment or structural obfuscation. Thus produced structurally obfuscated design is harder to be reverse engineered by an adversary, hence thwarting the theft of original design intents and also the potential insertion of hardware Trojan in an untrustworthy design house or foundry.

4.2.3. Metric for evaluating security of hologram based obfuscation

A strength of obfuscation (S_n) metric is used to analyze the security achieved using the proposed hologram based obfuscation technique. The S_n metric is given as follows:

$$S_n = \frac{G_c^f}{G_c^T} \tag{4.1}$$

Where, G_c^f denote the number of gates affected due to applying obfuscation and G_c^T denote total number of gates in the respective un-obfuscated design.

Further, the number of gates affected (G_c^f) post structural obfuscation is computed using the following equation:

$$G_c^f = \Delta G_c^{fb} + G_c^{ip} \tag{4.2}$$

Where, ΔG_c^{fb} denotes the difference in gate count between obfuscated design and un-obfuscated version and G_c^{ip} denote the number of gates altered in terms of input connectivity post obfuscation.

4.3. Demonstration on Generating Hologram Obfuscated FIR-IIR Filter Integrated Datapath

This section elaborates the process of generating structurally obfuscated FIR-IIR filter integrated datapath using the proposed hologram based obfuscation approach. The hologram based structural obfuscation mechanism is a promising solution to protect both the IIR and the FIR filter cores concurrently at low cost, against the RE based attacks. The first rule of the hologram based obfuscation is applied to generate the structurally obfuscated integrated datapath of FIR-IIR filter. The generic equation of IIR and FIR filters are as follows:

(i) IIR filter equation:

Y[n]=b0*X[n]+ b1*X[n-1]+ b2*X[n-2]+b3*X[n-3]- a1*Y[n-1]- a2*Y[n-2]-a3*Y[n-3] (4.3)

Where, Y[n] is the output of IIR filter, a1 to a3 and b0 to b3 are the input coefficients of IIR filter, X[n] is the current input, X[n-1], X[n-2] and X[n-3] are the previous inputs of IIR filter and Y[n-1], Y[n-2] and Y[n-3] are the previous outputs of IIR filter.

(ii) FIR filter equation:

Y'[n]=h0*X'[n]+h1*X'[n-1]+h2*X'[n-2]+h3*X'[n-3] (4.4)

Where, Y'[n] is the output FIR filter, h0 to h3 are the input coefficients of FIR filter, X'[n] is the current input and X'[n-1], X'[n-2] and X'[n-3] are the

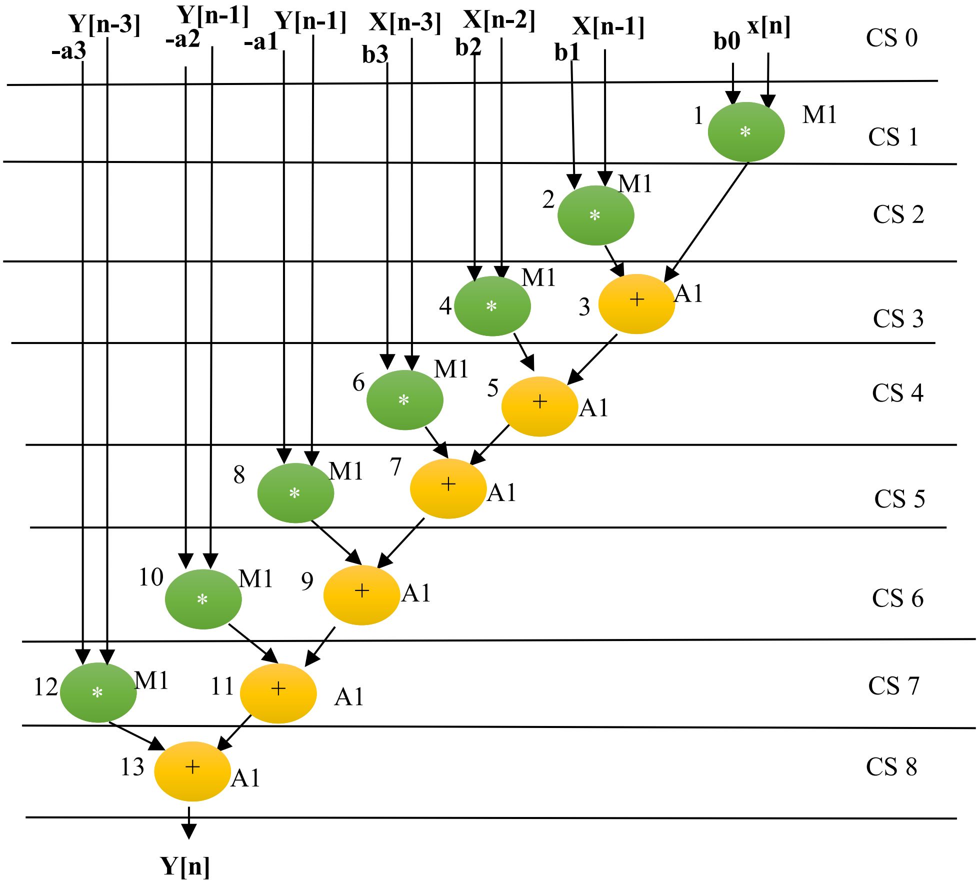


Fig. 4.3. Scheduling of IIR filter based on 1 adder and 1 multiplier

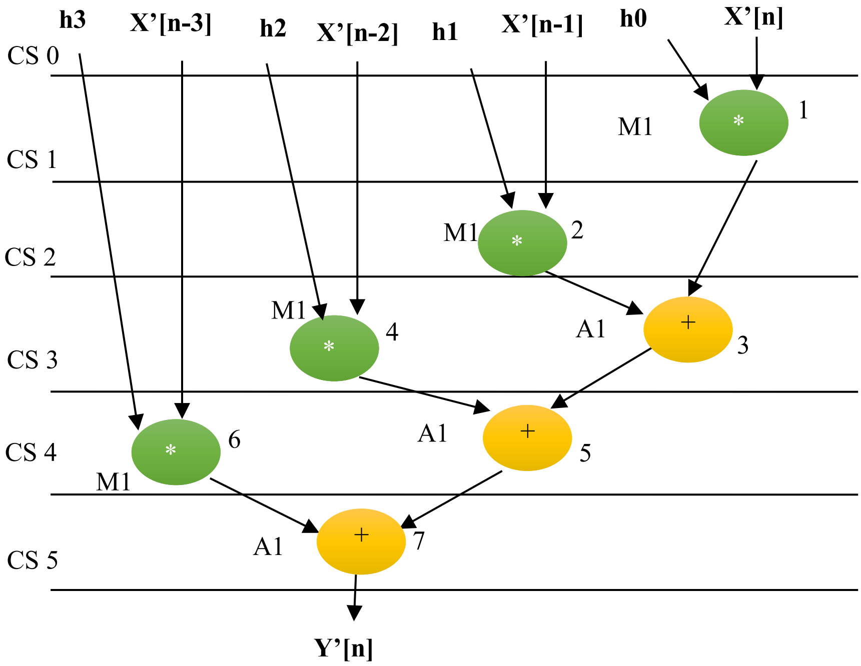


Fig. 4.4. Scheduled DFG of FIR filter based on 1 adder and 1multiplier

previous inputs of FIR filter.

Firstly, the algorithmic representations of IIR and FIR filters in their given mathematical form are converted into the corresponding DFG representations. Thereafter DFGs are subjected to the pre-synthesis phase of HLS where scheduling, allocation and binding are performed using the resource constraints of one adder (A1) and one multiplier (M1), resulting into DFG_{sab}-1 and DFG_{sab}-2. The Fig. 4.3 and Fig. 4.4 show the DFG_{sab}-1 of IIR filter and DFG_{sab}-2 of FIR filter respectively, where CS denotes the control step.

Post obtaining the DFG_{sab}-1 and DFG_{sab}-2, they are subjected to the first rule of hologram based obfuscation approach during the datapath synthesis phase of HLS process. As per the rule, inputs of only matched portions of DFG_{sab}-1 and DFG_{sab}-2 are subjected to multiplexing using 2:1 Muxes (switching elements of the hologram) and output of both DFG_{sab} are also subjected to multiplexing using a single 2:1 Mux (another switching element of the hologram) during the datapath synthesis phase of HLS. Activation of either of the filter application in the integrated datapath depends on the value of control input 'C'. In other words, the specific bit value of the control input 'C' manifests the respective datapath architecture in the hologram design. Specifically, the functionality of IIR and FIR filter in the hologram obfuscated datapath is enabled at IP designer controlled input ‘C’=0 and ‘C’=1 respectively. Post applying the first rule of proposed hologram based obfuscation during the datapath synthesis phase of HLS, we obtain an obfuscated integrated RTL datapath of both IIR+FIR filter cores simultaneously as shown in Fig. 4.5.

4.4. Similarity of Hologram Obfuscated Design with a Security Image Hologram

The proposed structurally obfuscated hologram design is motivated from the security image hologram feature. In a security image hologram, switching or

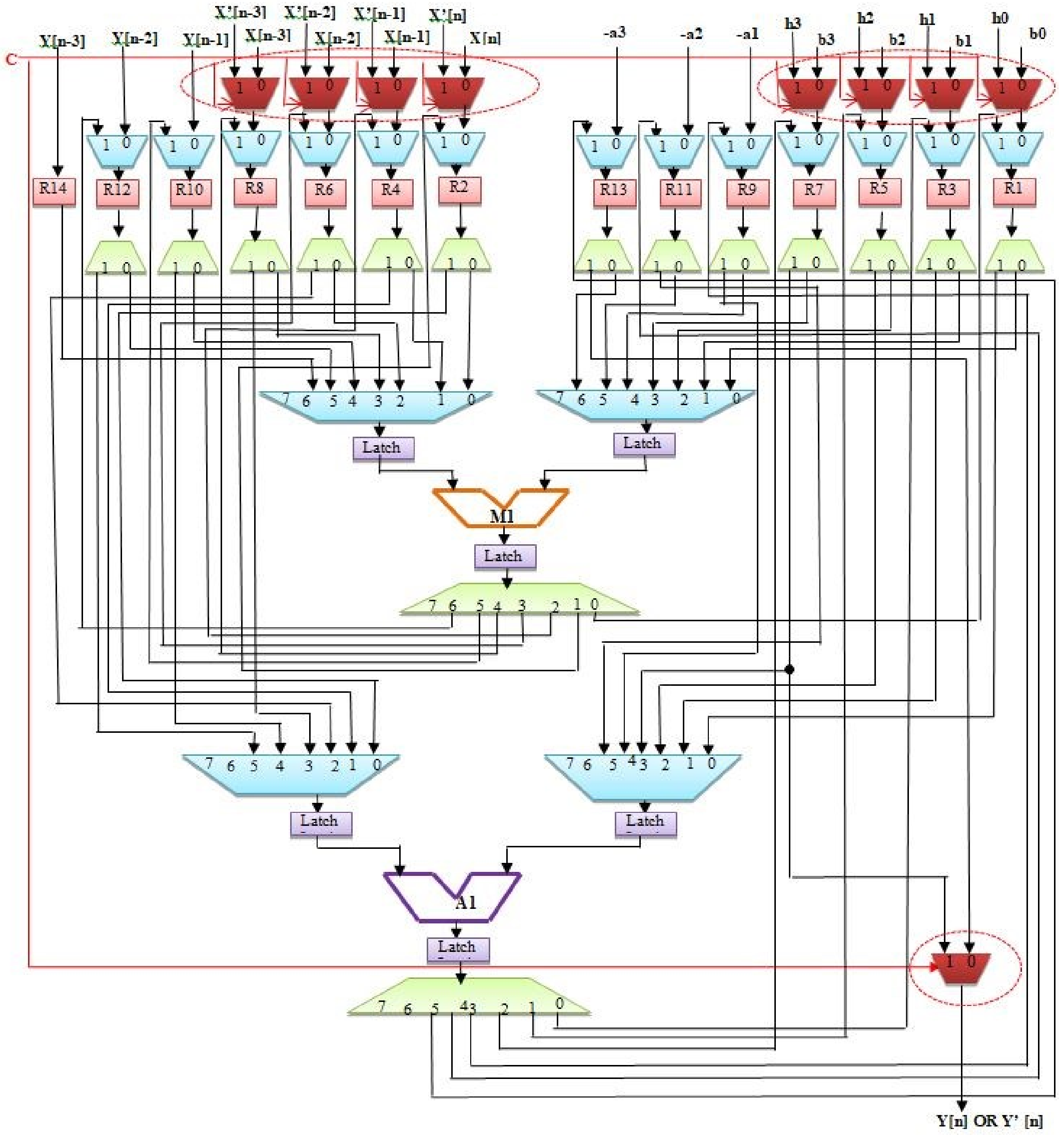


Fig. 4.5. Hologram based structurally obfuscated integrated RTL design of IIR and FIR filter cores (note: switching Muxes are highlighted in red dotted ovals)

flipping elements are embedded during their creation. These flipping elements execute the switching between two (or more) images when the image is viewed from different viewing angles [55]. Similarly in a structurally obfuscated hologram design, a number of additional multiplexers (acting as flipping elements) are exploited to realize the hologram feature. These multiplexers execute the switching between two designs integrated in the single RTL datapath, when a specific bit (control input 'C') value is applied.

4.5. Summary

The hardware threats of potential Trojan insertion and stealing the original design intents can be realized by an adversary if the RE can be performed successfully. The potential RE based attacks can be handled by performing the concealment of internal architecture of the design using a structural obfuscation technique. We proposed a novel hologram based structural obfuscation technique which makes the RE arduous for an attacker, hence preventing against the potential Trojan insertion and IP theft attacks. We proposed multiple rules of generating hologram based obfuscated designs during the datapath synthesis phase of the HLS process. The security using the proposed structural obfuscation was evaluated using strength of obfuscation metric which measure the total affected gate count post obfuscation w.r.t. the un-obfuscated counterpart.

Chapter 5

Double Line of Defense Approach using Integrated Structural Obfuscation and Crypto-steganography to Secure IP Cores

This chapter presents a double line of defense approach to secure IP cores against IP piracy and potential hardware Trojan insertion (resulting from reverse engineering) by an adversary in an untrustworthy design house or foundry. A structural obfuscation mechanism is performed during high level transformation to deploy the first line of defense to counter the threat of potential hardware Trojan insertion. Further, crypto based steganography is performed during high level synthesis (HLS) to deploy the second line of defense to counter the threat of IP piracy. The chapter also demonstrates the structural obfuscation and crypto-steganography based double of defense approach on joint photographic expert group (JPEG) compression processor and a discrete Fourier transform (DFT) processor.

Outline of the chapter is as follows. The first section formulates the problem. The second section discusses the double line of defense approach under the following sub-sections: overview, elaborating structural obfuscation acting as a first line of defense and crypto-steganography acting as a second line of defense, detection of steganography and metric for evaluating the security achieved using double line of defense. Further, the third section demonstrates the securing of application specific processors using double line of defense for the following two applications (i) JPEG compression (ii) DFT. Finally, the fourth section summarizes the chapter.

5.1. Problem Formulation

Given the data flow graph (DFG) representation of a target application, module library, resource constraint, along with the objective of securing IP cores against IP piracy and hardware Trojan insertion attacks, generate a secured (structurally obfuscated and stego-embedded) IP core.

5.2. The Double Line of Defense during High Level Synthesis Process for Securing IP Cores

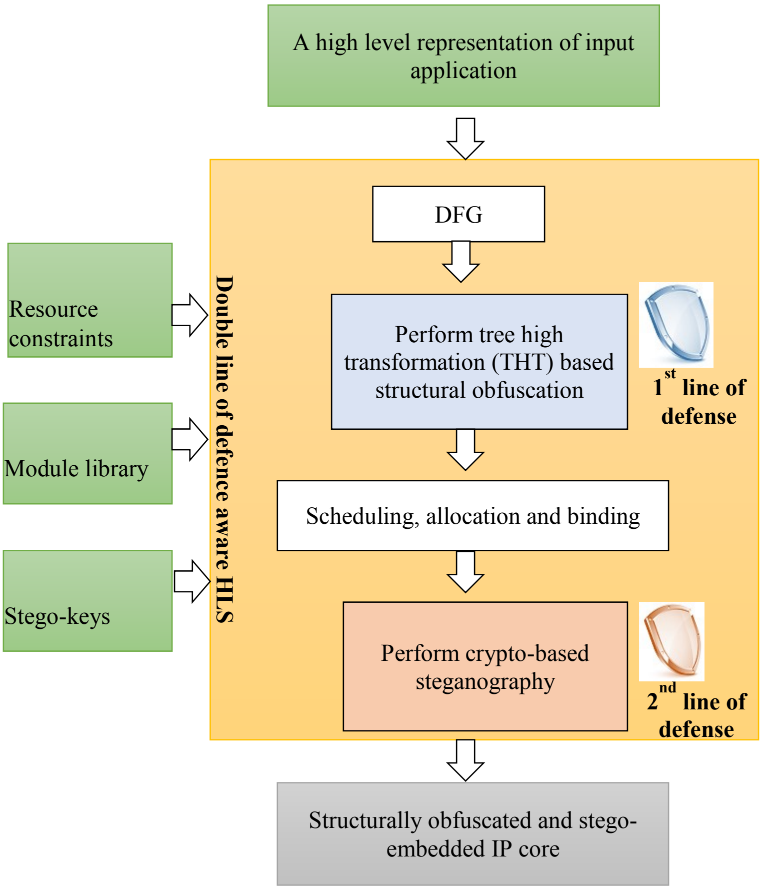

A high level representation of input application
DFG
Resource constraints
Module library
Stego-keys
Double line of defence aware HLS
Perform tree high transformation (THT) based structural obfuscation
1st line of defense
Scheduling, allocation and binding
Perform crypto-based steganography
2nd line of defense
Structurally obfuscated and stego-embedded IP core

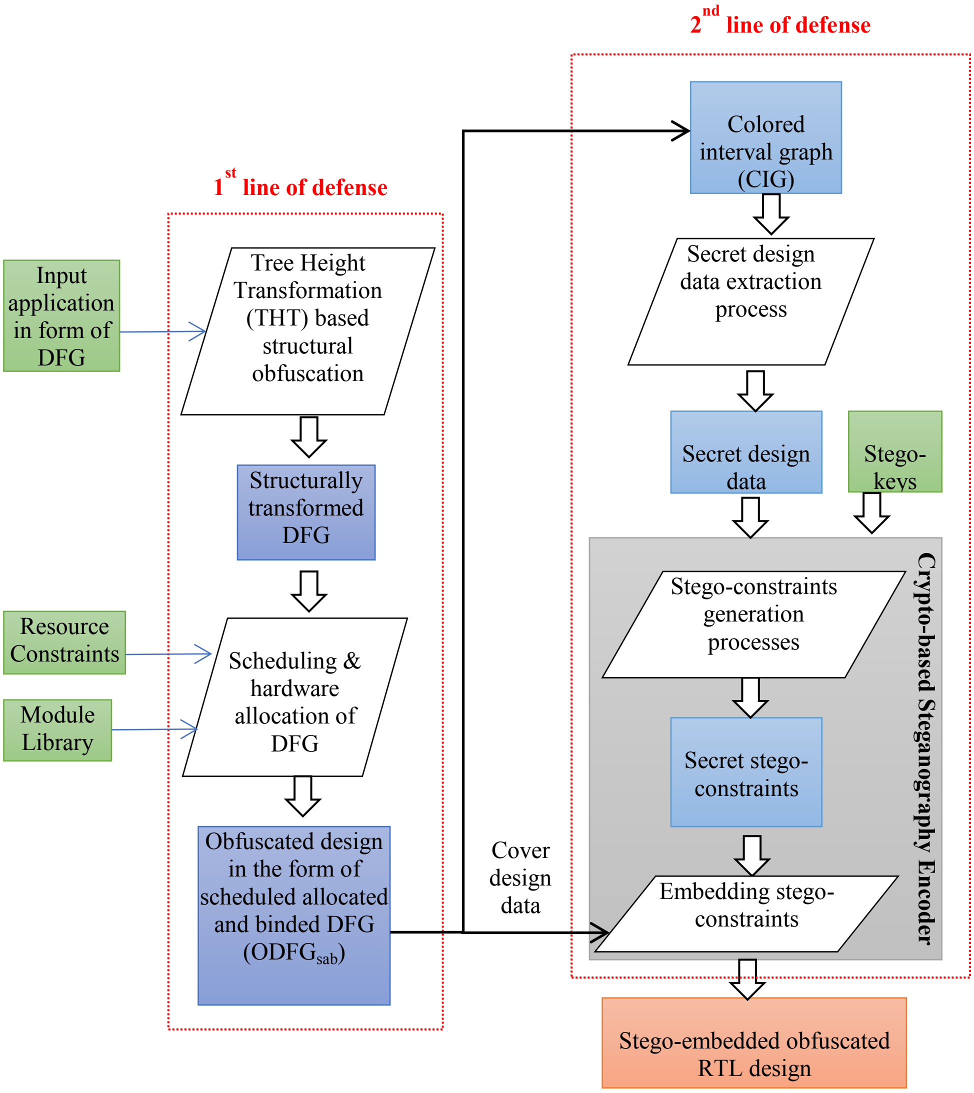
Fig 5.1. Overview of double line of defense based security mechanism for securing IP cores

In an untrustworthy design house or foundry, the IP cores are susceptible to both piracy and hardware Trojan insertion threat. We employ a double line of defense mechanism to counter both threats simultaneously. The double of defense approach for securing the IP cores is discussed under the following sub-sections:

5.2.1. Overview

An overview of the proposed double line of defense approach is shown in Fig. 5.1. As highlighted in the figure, the double line of defense technique is unified with the HLS process, leading to a security aware HLS framework. In the security aware HLS framework, the primary input is a high level description (e.g. C/C++ code or mathematical function) of input application. The major steps of securing the IP cores by performing double line of defense during the HLS process are as follows: (i) converting algorithmic description of the input application into corresponding data flow graph (DFG) (ii) subjecting the DFG to tree height transformation (THT) based structural obfuscation that works as first line of defense (iii) performing scheduling, allocation and binding of structurally obfuscated DFG using resource constraints and module library (iv) performing crypto-based steganography (as a second line of defense) on the obfuscated scheduled, allocated and binded DFG ($ODFG_{sab}$) using a large size stego-key. Post applying structural obfuscation and crypto-steganography as double line of defense during HLS process, a secured RTL design of intended application is produced at the output. The first and second line of defense is discussed in more detail as follows:

5.2.2. Elaborating structural obfuscation and crypto-steganography



2nd line of defense
1st line of defense
Input application in form of DFG
Tree Height Transformation (THT) based structural obfuscation
Structurally transformed DFG
Resource Constraints
Module Library
Scheduling & hardware allocation of DFG
Obfuscated design in the form of scheduled allocated and binded DFG (ODFGsab)
Cover design data
Colored interval graph (CIG)
Secret design data extraction process
Secret design data
Stego-keys
Stego-constraints generation processes
Secret stego-constraints
Embedding stego-constraints
Crypto-based Steganography Encoder
Stego-embedded obfuscated RTL design

Fig. 5.2. Flow of the process of securing JPEG codec processor using structural obfuscation (first line of defense) and crypto-based steganography (second line defense)

based double line of defense approach

The double line of defense approach is elaborated in Fig. 5.2. The details are discussed as follows:

(i) Structural obfuscation as a first line of defense

Applying structural obfuscation in designs conceal their internal architectures to thwart reverse engineering (RE), thus hindering backdoor (Trojan) insertion and IP theft. We applied a high level transformation to realize the structural obfuscation where tree height of DFG is transformed to create an obfuscated DFG. In order to apply the tree height transformation (THT), the serial execution flow of addition operations in the DFG is broken and parallel sub-computations are enabled. Thus applied THT based structural obfuscation incurs considerable alterations in the structure of the design in terms of the following: (a) alterations in the interconnectivity of high level components such as adders, and multipliers etc. (b) alterations in the total count of interconnect binding units such as Muxes and Demuxes (c) alterations in the total count of storage units such as registers. These alterations makes the design structure unobvious to be interpreted (through RE) by an attacker. This prevents an attacker from launching RE based attacks of backdoor insertion into hardware and the design theft.

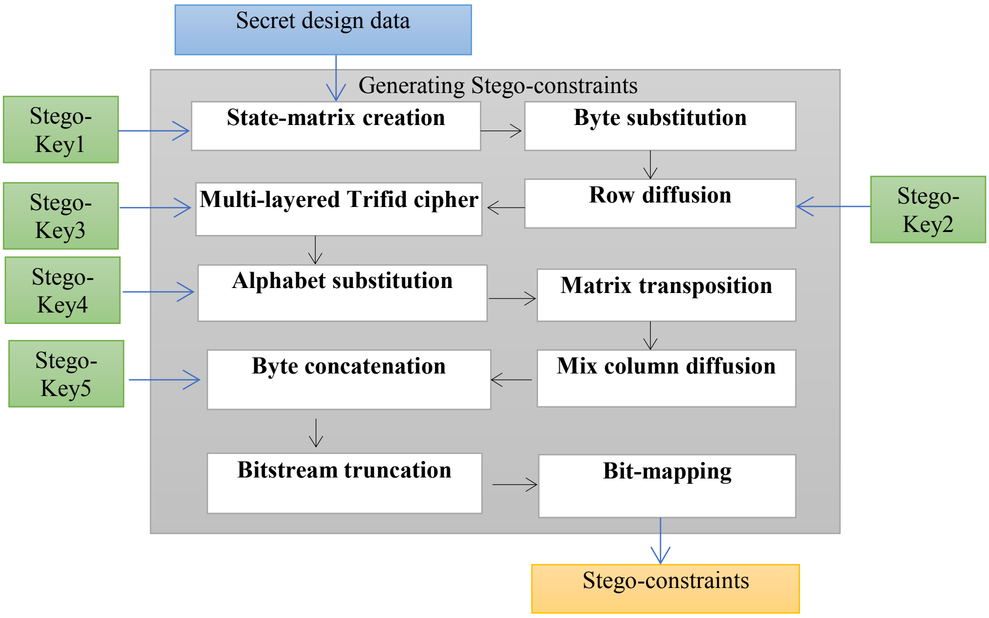
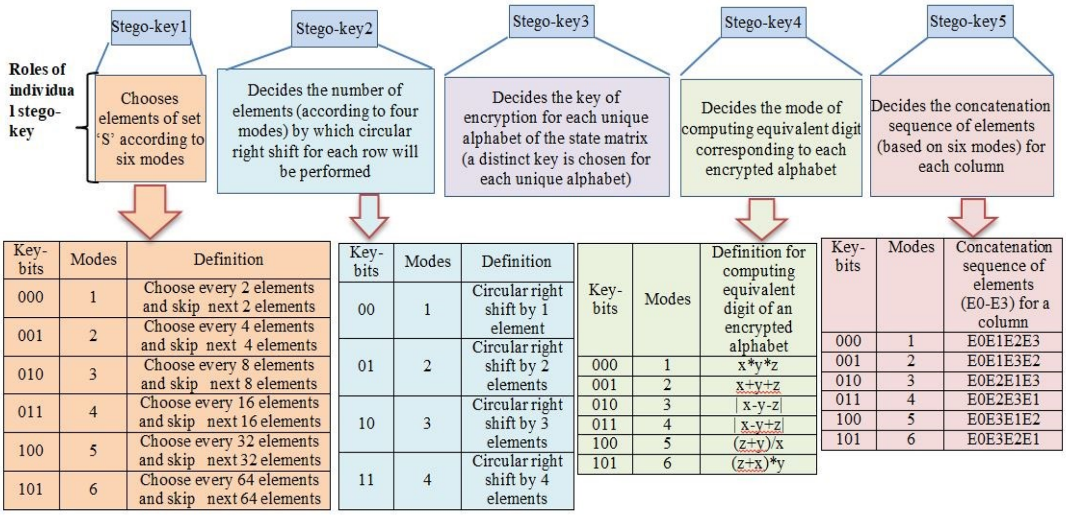


Fig. 5.3. Steps of stego-constraints generation process of crypto-based steganography encoder system

Fig. 5.2 shows the flow of applying first line of defense using structural obfuscation. As shown in the figure, the DFG of input application is first subjected to THT based transformation to create a corresponding obfuscated DFG. Further it is subjected to scheduling allocation and binding steps of HLS to produce an obfuscated scheduled, allocated and binded DFG (ODFG$_{sab}$). Further, ODFG$_{sab}$ of input application is fed to the process of applying second line of defense based on crypto-steganography technique.

(ii) Crypto-steganography as a second line of defense

The crypto-steganography technique embeds secret stego-constraints into the design during two distinct steps of HLS process viz. register allocation step and functional unit (FU) vendor allocation step. The embedded stego-constraints into a design enable the identification of authentic and pirated (counterfeited or cloned) designs during the detection process. In the proposed approach, the stego-constraints (or stego-information) are generated through various steps of crypto-steganography encoder system. As shown in Fig 5.2, the crypto-steganography encoder system requires following inputs to generate the stego-information: (a) secret design data and (b) stego-key. The secret design data is extracted from the colored interval graph (CIG) which in turn is created from the ODFG$_{sab}$ of input application as shown in Fig. 5.2. The secret design data is defined as follows. It is a set 'S' of indices pairs of nodes of same colors in the CIG. The details of stego-constraints generation and


Roles of individual stego-key
Stego-key1
Chooses elements of set 'S' according to six modes
Stego-key2
Decides the number of elements (according to four modes) by which circular right shift for each row will be performed
Stego-key3
Decides the key of encryption for each unique alphabet of the state matrix (a distinct key is chosen for each unique alphabet)
Stego-key4
Decides the mode of computing equivalent digit corresponding to each encrypted alphabet
Stego-key5
Decides the concatenation sequence of elements (based on six modes) for each column

Key-bits	Modes	Definition
000	1	Choose every 2 elements and skip next 2 elements
001	2	Choose every 4 elements and skip next 4 elements
010	3	Choose every 8 elements and skip next 8 elements
011	4	Choose every 16 elements and skip next 16 elements
100	5	Choose every 32 elements and skip next 32 elements
101	6	Choose every 64 elements and skip next 64 elements

Key-bits	Modes	Definition
00	1	Circular right shift by 1 element
01	2	Circular right shift by 2 elements
10	3	Circular right shift by 3 elements
11	4	Circular right shift by 4 elements

Key-bits	Modes	Definition for computing equivalent digit of an encrypted alphabet
000	1	x*y*z
001	2	x+y+z
010	3	|x-y-z|
011	4	|x-y+z|
100	5	(z+y)/x
101	6	(z+x)*y

Key-bits	Modes	Concatenation sequence of elements (E0-E3) for a column
000	1	E0E1E2E3
001	2	E0E1E3E2
010	3	E0E2E1E3
011	4	E0E2E3E1
100	5	E0E3E1E2
101	6	E0E3E2E1

Fig. 5.4. Roles and various modes of stego-key1 to stego-key5

embedding process are discussed below.

(1) Stego-constraints generation

The crypto-steganography encoder system performs the steps shown in Fig. 5.3 to generate the stego-information in the form of a bitstream. These steps are elaborated as follows:

(i) State-matrix creation

A state matrix is formed using selected elements of the set representing the secret design data. The elements are selected based on stego-key1 and the state matrix is created by arranging four selected elements in each row. There are various modes of choosing the elements depending on the value of 3-bit stego-key1 as shown in Fig. 5.4.

(ii) Byte substitution

Each element of the state matrix is subjected to the byte substitution or nonlinear bit-manipulation performed using forward S-box. The relationship of final stego-constraints with the stego-key is obscured (Shannon's property of confusion) using this process.

(iii) Row diffusion

The row diffusion process obscures the relationship of stego-constraints with the input secret design data in the matrix (Shannon's property of diffusion). The row diffusion is driven through stego-key2. The value of stego-key2 determines the amount of circular right shift to be performed in each row. Let's say the number of rows in the matrix are 'W', then the stego-key2 size is 2×W bits. The role of stego-key2 and its various modes are highlighted in Fig. 5.4. As per the definition of different modes, the rows of the state matrix are subjected to diffusion.

(iv) Trifid cipher based encryption

The Trifid cipher provides certain amount of confusion and diffusion to obscure the relationship of the stego-constraints with the secret design data and stego-keys. The Trifid cipher based encryption is accomplished on each unique alphabet of the matrix. The encryption key for each unique alphabet is determined by the stego-key3. The chosen encryption key contains 27 unique characters to encrypt each unique alphabet. Because of total 27! possible permutations of 27 characters, the number of bits needed to indicate the key for an alphabet is $\lceil(\log_2(27!)\rceil$. If the number of unique alphabets in the matrix post diffusion is N_A then the total size of stego-key3 to encrypt all unique alphabets is = $N_A \times \lceil(\log_2(27!)\rceil$. To perform the encryption of an alphabet, the 27 characters of the key are divided into three 3x3 matrices. An encrypted alphabet is represented in the form of a 3-digit value "xyz", where x, y and z denote the row number, column number and the matrix number in which the input alphabet is located.

(v) Alphabet substitution

Post obtaining encrypted alphabets in the form of 3-digit value "xyz", the equivalent single digit is calculated based on stego-key4. There are various modes of stego-key4 which decide the mathematical expression to be used to calculate the equivalent single digit for each encrypted alphabet, as shown in Fig. 5.4. Therefore, the size of the stego-key4 is computed to be $N_A \times \lceil(\log_2(\text{number of modes for calculating single digit equivalent})\rceil$. Post obtaining the equivalent single digits, they are used to substitute the corresponding alphabets of state matrix.

(vi) Matrix transposition

In this step, the matrix is transposed.

(vii) Mix column diffusion

Each column of the transposed matrix is subjected to mix column diffusion to incur the Shannon's property of diffusion. A circulant MDS (Maximum Distance Separable) matrix is used to perform the mix column diffusion.

(viii) Byte concatenation

To generate a sequence of bytes, the elements (bytes) of each column in the updated matrix are concatenated. However, the concatenation for each column is performed based on the value of stego-key5. There are different modes of concatenation based on the value of stego-key5 as shown in Fig. 5.4. The size of stego-key5 is computed to be (number of columns)$\times \lceil (\log_2(\text{number of modes for concatenation})\rceil$. Post concatenating all bytes of the state matrix, the obtained byte sequence is converted into a bitstream. Further, the bitstream can be truncated to the designer's specified size.

(ix) Bit mapping

The bit '0' and bit '1' in the truncated bitstream are mapped into respective stego-constraints to enable the embedding into the design during the HLS process. The mapping of bit '0' and bit '1' is given below.

'0'→ add an edge between node pair (even, even) of CIG

'1'→ odd operations are allocated to FU of vendor type 1 and even operations are allocated to FU of vendor type 2

(2) Stego-constraints Embedding during HLS

The stego-constraints corresponding to bit '0' and bit '1' are embedded into the design during register allocation and FU vendor allocation phase respectively. As per the mapping of bit '0' into stego-constraints, the obtained constraint edges are embedded into the CIG as additional edges. The embedding of constraint edges, in some cases, may require additional colors/registers, thus resulting into design overhead. Further as per the mapping of bit '1' into stego-constraints, the operations of the $ODFG_{sab}$ are allocated to the particular FU vendor type specified through the mapping rule. Thereby, the stego-information is implanted into the design during two different phases of HLS process. The embedded stego-information enables the detection of IP theft/piracy.

5.2.3. Detection of IP piracy using crypto-steganography

The detection of steganography in the intended designs enables the identification of counterfeiting and cloning. The three major processes are involved in detecting steganography information: (i) secret stego-constraints generation process (ii) concealed stego-constraints extraction from stego-embedded RTL datapath of the design (iii) matching of generated and extracted stego-information to confirm the existence of vendor's stego-mark into the design.

5.2.4. Metric used to evaluate the security of double line of defense

Following metric are used to evaluate the security achieved using structural obfuscation and crypto-steganography based double line of defense:

1. Strength of structural obfuscation:

The measure of strength of structural obfuscation is the amount of gates affected owing to change in overall gate count and the alterations in the interconnectivity of gates. It is important to note that the modification in the number and size of RTL components affects the gates of the design; therefore the change in amount of gates does not follow any fix pattern hence hindering the attacker in deducing the correct structure of the design.

2. Probability of coincidence of crypto based dual phase steganography:

This metric is measured using the following formula:

$$\mathrm{Pc} = \left(1 - \frac{1}{\mathrm{G}}\right)^{\mathrm{f1}} \times \left(1 - \frac{1}{\pi_{\mathrm{i=1}}^{\mathrm{y}} \mathrm{N(Zi)}}\right)^{\mathrm{f2}} \qquad (5.1)$$

Where, the first term in the equation corresponds to the Pc due to embedding constraints in the register binding phase and the second term corresponds to the Pc due to embedding constraints in the FU vendor allocation phase. Here, in the first term, G denotes the number of registers before embedding steganography, and f1 denotes the number of stego constraints added to the CIG. Further, in the second term, ‘y’ denotes the types of resources in the DSP application, N(Zi) denotes the number of instances of each FU type and f2 denotes the number of stego constraints added during the FU vendor allocation phase.

3. Total stego-key size:

Total stego-key size (S_T) (in bits) of the crypto-based steganography is given as follows:

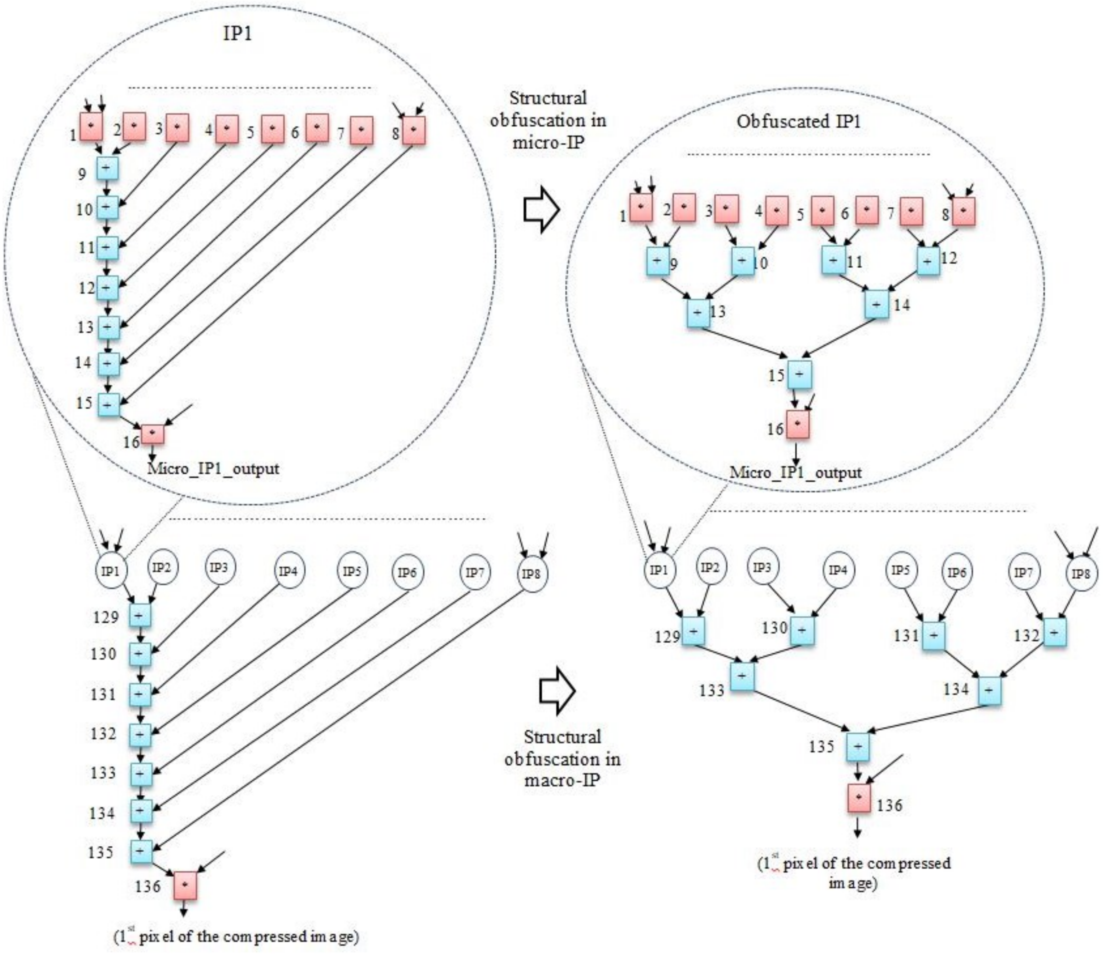
S_T= 3 bits+ 2×W+ (N_A)× ┌(log$_2$(27!)┐ + (N_A) × ┌(log$_2$(number of modes for calculating single digit equivalent)┐ + (number of columns) × ┌(log$_2$(number of modes for concatenation)┐ (5.2)

5.3. Demonstration of Securing Application Specific Processors using Double Line of Defense

The proposed double line of defense approach is applied on following two application specific processors for demonstration: (i) JPEG compression processor (ii) DFT processor.

5.3.1. Securing JPEG compression processor

The application specific processor of the JPEG compression application can be designed in the RTL form using the HLS process. The HLS process first takes the algorithmic description of the computational intensive portion of the JPEG compression application as input and creates a DFG. The computational intensive portion of a JPEG application is the DCT transformation using the 2D-DCT coefficient matrix followed by compression using the quantization matrix. The equations that compute the DCT transformation and quantization are presented in detail in [8], [47]. The corresponding DFG is shown in Fig. 5.5(a).


IP1
Structural obfuscation in micro-IP
Obfuscated IP1
Micro_IP1_output
IP1
IP2
IP3
IP4
IP5
IP6
IP7
IP8
Structural obfuscation in macro-IP
(1st pixel of the compressed image)
(1st pixel of the compressed image)

(a) (b)

Fig. 5.5. (a) DFG of JPEG compression application (b) THT based obfuscated DFG

To secure the JPEG compression processor using structural obfuscation and crypto-steganography based double line of defense against the threats of potential backdoor insertion and IP piracy, following steps are performed:

(1) Applying structural obfuscation based first line of defense

The DFG of JPEG compression application is subjected to THT based structural transformation technique to produce obfuscated DFG as shown in Fig. 5.5(b). This obfuscated DFG is further subjected to scheduling, allocation and binding steps of HLS using the resource constraints of 3 multipliers and 3 adders, resulting into $ODFG_{sab}$. Further, this $ODFG_{sab}$ is fed to the crypto-steganography process for deploying second line of defense.

(2) Applying crypto steganography based second line of defense

The $ODFG_{sab}$ of JPEG compression application is applied with crypto-based steganography in the following steps:

(i) Obtain a register allocation information or CIG from the $ODFG_{sab}$.

(ii) Extract the secret design data from the register or color assignment of $ODFG_{sab}$.

(iii) Apply the various steps of crypto-steganography based on the following values of stego-keys:

Stego-key1: “001” (mode-2: select 4 elements and skip 4 elements)

Stego-key2: “11-10-00-01-00-10-10-10-11-10-00-00-10-01-11-11-11-11-10-00-00-10-10-11-01-11-11-01-11-01-00-11-11-11-00-11-01-11”

Stego-key3:

To encrypt the alphabet ‘a’ = v$qawsedrftgyhujikolpzmxncb

To encrypt the alphabet ‘b’ = qawsedrftgyhujik$olpzmxncbv

To encrypt the alphabet ‘c’ = olpzmxncbv$qawsedrftgyhujik

To encrypt the alphabet ‘d’= gyhujik$olpzmxncbvqawsedrft

To encrypt the alphabet ‘e’= ftgyhujikolpzmxncbv$qawsedr

To encrypt the alphabet ‘f’= lpzmxncbvqawsedrftgyhujik$o

Stego-key4: “010-001-100-101-011-001”

Stego-key5: “000-001-010-011-100-101-001-011-010-100-100-000-100-100-011-010-001-000-100-101-011-010-001-000-101-011-001-000-100-101-011-010-001-011-101-011-011-100”

The total size of stego-key is computed to be 775 bits using eq. (5.2).

(iv) Post applying the crypto-steganography, the generated bitstream is truncated to the size of 400.

(v) Post mapping the bits of truncated bitstream into stego-constraints (using the mapping rules discussed in section 5.2.2), the constraints are added during the register allocation and FU vendor allocation phase of HLS.

Post adding the stego-constraints to the design, datapath is synthesized to generate the stego-embedded and obfuscated RTL design of application specific processor of JPEG compression application. Because of applying the proposed double line of defense, the processor becomes secured against the hardware threats of potential backdoor insertion and piracy.

5.3.2. Securing DFT processor

Discrete Fourier Transform (DFT) is a conversion of a signal from its discrete-time representation to a discrete-frequency representation. In order to design a secured application specific processor for a DFT application, the proposed double line of defense mechanism is integrated with the HLS design process. In the security aware HLS design flow, the mathematical form of a DFT application is first subjected to conversion into corresponding DFG

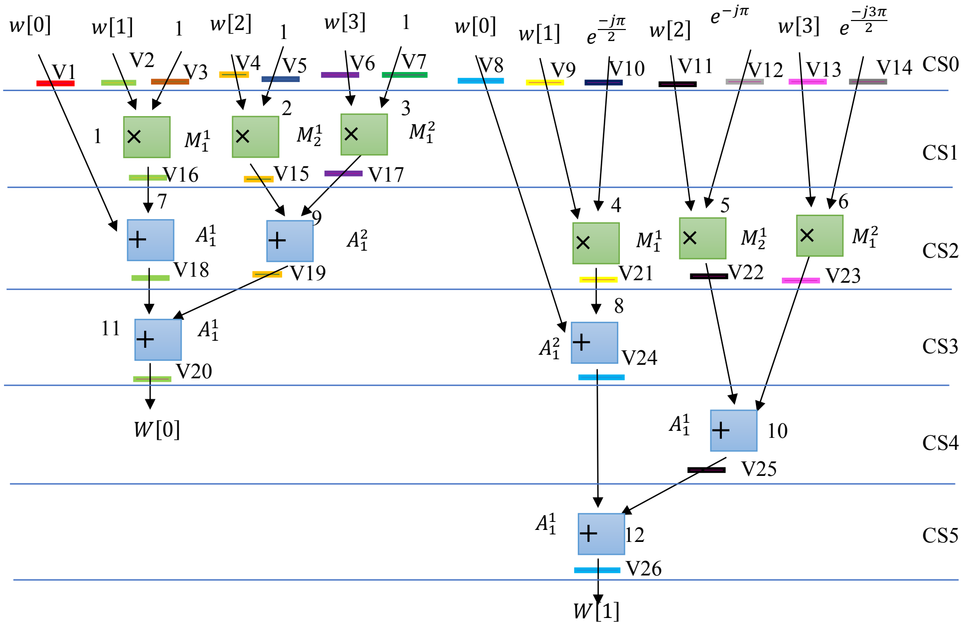


Fig. 5.6. ODFG$_{sab}$ of obfuscated 4-point DFT based on 3M and 2A

Note: For an FU resource F_b^a, superscript ‘a’ indicates the vendor type and subscript ‘b’ indicates the instance number representation. In the proposed approach, the DFG of 4-point DFT application is constructed to enable the computing of two output values (W[0] and W[1]) concurrently to accelerate the execution. Further the DFG is subjected to following double line of defense mechanism during HLS process.

(1) Applying structural obfuscation based first line of defense

The DFG of 4-point DFT application is subjected to THT based structural transformation technique to produce obfuscated DFG. This obfuscated DFG is further subjected to scheduling, allocation and binding steps of HLS using the resource constraints of 3 multipliers and 2 adders, resulting into ODFG$_{sab}$ of DFT application as shown in Fig. 5.6. Further, this ODFG$_{sab}$ is fed to the crypto-steganography process.

(2) Applying crypto steganography based second line of defense

The ODFG$_{sab}$ of DFT application is applied with crypto-based steganography in the following steps:

(i) Obtain a register allocation information or CIG from the ODFG$_{sab}$ of DFT application.

(ii) Extract the secret design data from the register or color assignment.

(iii) Apply the various steps of crypto-steganography based on the

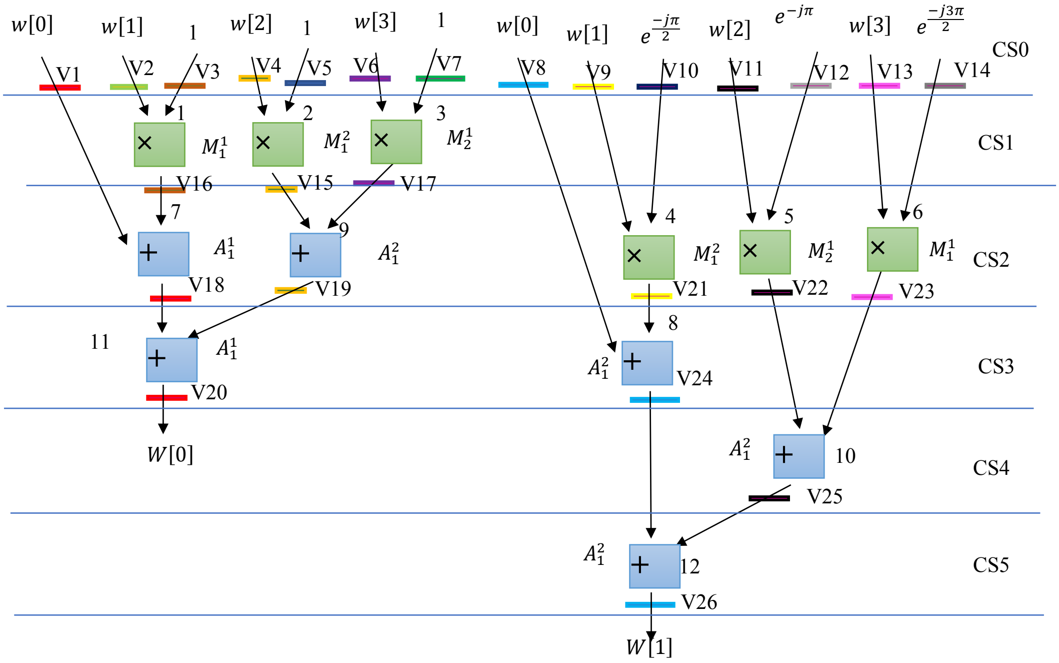


Fig. 5.7. ODFG$_{sab}$ of 4-point DFT post embedding stego-information

following values of stego-keys:

Stego-key1: “001” (mode-2: select 4 elements and skip 4 elements)

Stego-key2: “01 00”

Stego-key3:

To encrypt the alphabet ‘a’ = v#qawsedrftgyhujikolpzmxncb

To encrypt the alphabet ‘b’ = qawsedrftgyhujik#olpzmxncbv

To encrypt the alphabet ‘d’= gyhujik#olpzmxncbvqawsedrft

To encrypt the alphabet ‘f’= lpzmxncbvqawsedrftgyhujik#o

Stego-key4: “001 001 100 100”

Stego-key5: “001 000”

(iv) Post applying the crypto-steganography, the generated bitstream is truncated to the size of 27.

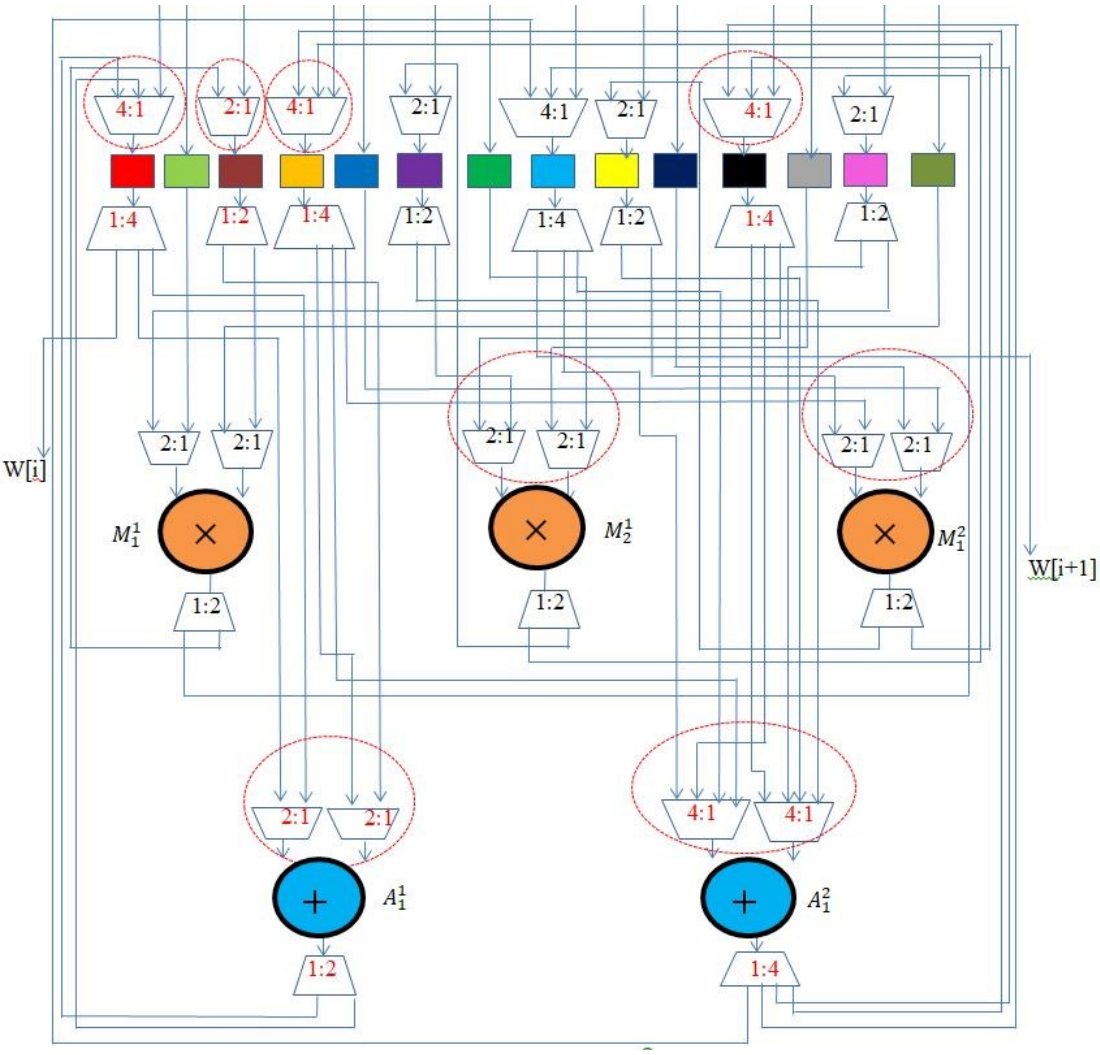


Fig. 5.8. Secured 4-point DFT processor at RTL (note: red ovals highlight the change in input of Muxes due to embedded stego-information)

(v) Post mapping the bits into stego-constraints (using the mapping rules discussed in section 5.2.2), the constraints are added during the register allocation and FU vendor allocation phase of HLS. Post adding the stego-constraints, the $ODFG_{sab}$ is shown in Fig. 5.7. Due to adding the stego-constraints, the storage variables are subjected to constraint based register allocation and the operations are subjected to constraints based FU vendor allocation as shown in Fig. 5.7.

Further, datapath is synthesized to generate the stego-embedded and obfuscated RTL design of application specific processor of DFT application. Fig. 5.8 shows the RTL design of secured (structurally obfuscated and stego-embedded) DFT processor.

5.4. Summary

Backdoor insertion and piracy both pose serious threats to hardware security. This chapter discussed a double line of defense mechanism where structural obfuscation is applied to combat the potential backdoor insertion threat and crypto-steganography is applied to combat the IP piracy threat. The metrics employed to measure the security using the double line of defense approach were also discussed in the chapter. Further, we demonstrated the process of generating secured application specific processor IPs for JPEG compression and DFT applications.

Chapter 6

Double Line of Defense Approach using Integrated Multi-key based Structural Obfuscation and Physical Level Watermarking to Secure IP Cores

This chapter presents a double line of defense approach employing multi-key based structural obfuscation as preventive control against potential backdoor insertion and physical level watermarking as detective control against IP piracy. The chapter also demonstrates the structural obfuscation and physical level watermarking based double of defense approach on a finite impulse response (FIR) filter core. Outline of the chapter is as follows. The first section formulates the problem. The second section discusses the double line of defense approach under the following sub-sections: overview, elaborating multi-key based structural obfuscation acting as a first line of defense and physical level watermarking acting as a second line of defense, detection of watermark and metric for evaluating the security achieved using double line of defense. Further, the third section demonstrates the securing of FIR filter core using the double line of defense. Finally, the fourth section summarizes the chapter.

6.1. Problem Formulation

Given the data flow graph (DFG) representation of a target application, module library, resource constraint, along with the objective of securing IP cores against IP piracy and hardware Trojan insertion attacks, generate a secured (structurally obfuscated and watermark embedded) IP core.

6.2. The Double Line of Defense during High Level and Physical Synthesis Processes for Securing IP Cores

The proposed double line of defense mechanism is employed to counter both backdoor (Trojan) insertion (resulting from reverse engineering attack) and IP piracy threats simultaneously. We exploited two different design synthesis processes to employ the double line of defense viz. high level synthesis and physical synthesis. The double of defense approach for securing the IP cores is discussed under the following sub-sections:

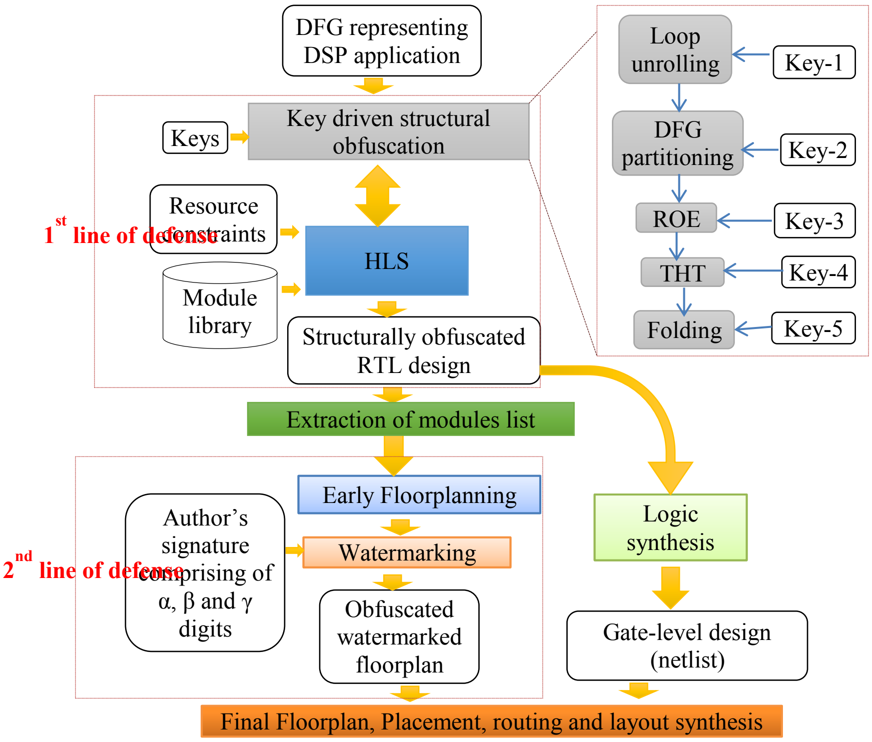

DFG representing DSP application
Keys
Key driven structural obfuscation
Resource constraints
1st line of defense
HLS
Module library
Structurally obfuscated RTL design
Loop unrolling
Key-1
DFG partitioning
Key-2
ROE
Key-3
THT
Key-4
Folding
Key-5
Extraction of modules list
Early Floorplanning
Author's signature comprising of α, β and γ digits
2nd line of defense
Watermarking
Obfuscated watermarked floorplan
Logic synthesis
Gate-level design (netlist)
Final Floorplan, Placement, routing and layout synthesis

Fig.6.1. The flow of proposed key-driven structural obfuscation and physical level watermarking based double line of defense

6.2.1. Overview

The overview of the proposed double line of defense approach is shown in Fig. 6.1. As highlighted in the figure, the structural obfuscation based first line of defense technique is unified with the HLS process, rendering the HLS framework security aware. In the security aware HLS framework, the primary input is a DFG form of input application. The major steps of applying first line of defense during the HLS process are as follows: (i) subjecting the DFG to the key driven five different structural obfuscation technique viz. key driven loop unrolling, key driven partitioning, key driven redundant operation elimination, key driven tree height transformation, and key driven folding transformation alongwith performing scheduling, allocation and binding using resource constraints and module library to obtain obfuscated scheduled, allocated and binded DFG (ODFG$_{sab}$) (ii) synthesizing the obfuscated RTL datapath secured against the potential backdoor insertion. Thereafter, we perform the extraction of the list of RTL components from the structurally obfuscated datapath. Further, the major steps of applying second line of defense during the physical synthesis process are as follows: (i) creating an

Table 6.1 Roles and key-size of different keys used for proposed structural obfuscation

Keys	Role	Key size in bits
Key-1	To regulate the Unrolling Factor (UF)	┌log$_2$(UF$_{max}$)┐
Key-2	To regulate the number of cuts applied to partition the DFG	┌ (log$_2$ (Max. cut) ┐
Key-3	To regulate ROE across the partitions	┌ (log$_2$ (Max. RO) ┐
Key-4	To regulate THT across the partitions	┌ (log$_2$ (Max. THT) ┐
Key-5	To regulate the folding of resources across the partitions	┌ (log$_2$ (Max. folding) ┐

early floorplan using the RTL components (ii) applying IP vendor's signature (composed of multiple variables viz. α, β and γ) to the early floorplan to obtain a watermarked floorplan. Post applying the watermarking, subsequent phases of physical synthesis such as creating final floorplan, placement and routing are performed to obtain the structurally obfuscated and watermarked IP.

6.2.2. Elaborating multi-key driven structural obfuscation and physical level watermarking based double line of defense approach

The double line of defense approach is elaborated as follows:

(i) Multi-key based structural obfuscation as a first line of defense

The DFG of input DSP application is subjected to structural obfuscation by applying five different key-driven techniques. Table 6.1 shows the role of each structural obfuscation key (SOK) and the corresponding size in bits. The applied key-driven techniques for structural obfuscation are elaborated as follows:

(a) Key-driven loop-unrolling technique

The loop body of a DSP application is unrolled in this technique, as per designers' chosen SOK-1 indicating the loop unrolling [49] factor (UF) value. Table 6.1 highlights the role and size of SOK-1. This structural obfuscation technique results into alterations in the architecture in terms of change in size of Mux/Demux, number of functional units (FUs) in case of unconstrained resources, and number of storage elements in the RTL design, thus rendering the reverse engineering harder to an attacker.

(b) Key-driven DFG partitioning technique

This technique applies m cuts to the unrolled DFG of DSP application in order to partition it into total 'm+1' partitions. The number of cuts applied is driven through the SOK-2. The DFG is portioned in such a way that each resulting partition holds atleast two connected nodes (operations). Table 6.1 highlights the role and size of SOK-2. The partitioning based structural obfuscation technique incurs massive changes in terms of interconnectivity of Muxes/Demuxes and FU resources, adding to significant obscurity in the RTL structure of the intended DSP circuit.

(c) Key-driven redundant operation elimination technique

An operation/node is considered to be redundant if its parent and operation type are same as an existing node in the DFG. This structural obfuscation technique is applied to the partitions of the DFG, where the redundant operations are removed based on the value of SOK-3. Table 6.1 highlights the role and size of SOK-3. This type of obfuscation affects the size and complexity of the Mux/Demux interconnections in the RTL.

(d) Key-driven tree height transformation technique

The key-driven tree height transformation technique alters the data dependency of addition operations by converting serial computations into parallel computations, based on the value of SOK-4. Table 6.1 highlights the role and size of SOK-4. This technique adds in the obfuscation by incurring the variations in the interconnectivity of FUs and Muxes/Demuxes inputs/outputs.

(e) Key-driven folding transformation technique

Post performing the scheduling of each individual obfuscated partition (generated after applying above four structural obfuscation techniques), the key-driven folding transformation technique is applied to enhance the obscurity. In an obfuscated scheduled partition, the folding with factor 2 is performed on two operations of same type executing in different control steps to enable their execution through the same respective FU resource. The number of instances where the folding transformation is applied is driven through the value of SOK-5. Table 6.1 highlights the role and size of SOK-5. This technique adds in the obfuscation by incurring the structural variation in terms of reduction in FUs and increase in size of several Mux/Demux and storage elements.

Once the key-driven structural obfuscation techniques are applied, the datapath of individual obfuscated partitions are synthesized. Further, the individual obfuscated datapaths are integrated together to generate the single obfuscated RTL datapath of the target DSP application. This process is followed by extraction of following RTL modules from the structurally obfuscated datapath: adders, multipliers, subtractors, comparator, Muxes and Demuxes. A sorted list 'T' of RTL modules is prepared based on their decreasing size [56]. This list 'T' of RTL modules is used in the physical synthesis process to construct an early floorplan.

(ii) Physical level watermarking as a second line of defense

The early floorplanning stage of physical synthesis process [57] is exploited to embed the vendor's watermark that enables detection of IP piracy. The early floorplan is constructed using the RTL modules. To do so, the list 'T' of RTL modules is traversed from left to right to fetch the modules one by one and place them so as to grow the floorplan diagonally. Once the early floorplan is constructed, following steps are performed to embed the watermark.

(a) Choosing vendor's signature composed of three variables viz. α, β and γ. The variables have following mapping into corresponding watermarking constraints:

α → Odd FU module takes the position on the top of even FU module, by swapping two FU modules of same type.

β→ Odd Mux takes the position on the top of even Mux, by swapping two Muxes of same size.

γ→ Odd Demux takes the position to the right of even Demux, by swapping two Demuxes of same size.

(b) Three different sorted lists T1, T2 and T3 of FU, Mux and Demux modules respectively are prepared.

(c) For embedding, α, β and γ digits of the signature, the sorted lists T1, T2 and T3 respectively are traversed. During embedding, if swapping of modules (FUs or Muxes or Demuxes) satisfies the watermarking constraints (or participates in implanting a signature digit (α or β or γ)) then the module pair is removed from the respective list (T1 or T2 or T3).

Further, the updated lists are exploited to embed remaining respective digits of the signature.

Post embedding the signature digits, we obtain the watermarked floorplan of the obfuscated design of target DSP application.

6.2.3. Detection of IP piracy using watermark

The detection of watermark is accomplished in the final floorplanned design file generated through the physical design tool. The following inputs are required to generate the final floorplanned design file during the physical design phase: (i) Verilog file of the design netlist (ii) obfuscated watermarked early floorplan file (iii) configuration file (iv) library files (f) other applicable files. Further, the arrangement (positions) of modules in the floorplanned design file is checked according to the vendor's signature digits. By examining the presence of secret watermark in the design, IP piracy can be detected.

6.2.4. Metric used to evaluate the security of double line of defense

Following metrics are used to evaluate the security achieved using multi-key based structural obfuscation and physical level watermarking based double line of defense:

1. Strength of structural obfuscation:

The strength of structural obfuscation is measured in terms of change in gate count. The change in overall gate count due to obfuscation depends on the change in size and number of the Muxes and Demuxes and number of storage elements which in turn depends on the potential resource sharing and type of the application (i.e. number of operations and their dependency) as well as the applicability of each design transformation technique of obfuscation. More explicitly, the gate count changes due to applying various techniques of structural obfuscation. Moreover, alteration in the interconnectivity of various RTL resources also adds in considerable structure modification in the design. This makes the design structure unidentifiable for an adversary, thus hindering malicious effort of backdoor insertion and IP theft.

2. Total structural obfuscation key size:

Total size of structural obfuscation key (S_{SOK}) (in bits) is given as follows:

S_{SOK}= (┌log$_2$(UF$_{max}$)┐)+(┌ (log$_2$ (Max. cut)┐)+(┌(log$_2$ (Max. RO) ┐)+(┌ (log$_2$(Max. THT)┐)+(┌(log$_2$(Max. folding)┐) (6.1)

3. Probability of coincidence of physical level watermarking:

This metric is measured using the following formula:

$$Pc = \left(\prod_{a=1}^{\alpha} \frac{1}{(\sum_{u\in U_r}\frac{u(u-1)}{2})-a++}\right) * \left(\prod_{b=1}^{\beta} \frac{1}{(\sum_{x\in X_v}\frac{x(x-1)}{2})-b++}\right) * \left(\prod_{c=1}^{\gamma} \frac{1}{(\sum_{d\in D_e}\frac{d(d-1)}{2})-c++}\right) \quad (6.2)$$

Where, ‘u’ indicates number of instances of FU type U_r, where r is the total types of FUs; ‘x’ indicates number of Muxes of size X_v, where v indicates various sizes of Mux in the design; ‘d’ indicates number of Demuxes of size D_e, where e indicates various sizes of Demux in the design. Further, the variables a, b and c vary in the following range: $0\leq a\leq \alpha-1$, $0\leq b\leq \beta-1$, $0\leq c\leq \gamma-1$, where a, b and c represent the corresponding count of swapping pairs embedded for α, β and γ digits respectively. In the eq. (6.2), the first term indicates the Pc corresponds to embedding α digits, the second term indicates the Pc corresponds to embedding β digits and the third term indicates the Pc corresponds to embedding γ digits.

4. Tamper tolerance:

The proposed watermark is tamper tolerant or resilient against the removal attack as it covertly inserts vendor’s signature consisting of three distinct variables (α, β and γ) in the early floorplan of the design. The following metric is used to measure the tamper tolerance (T_S) ability of the watermark:

$$T_S = Q^Z \quad (6.3)$$

Where, Q denotes the number of distinct variables in the chosen signature and Z denotes the size of the signature. The value of T_S also represents the total signature space. Larger the signature space, higher is resilience of watermark against the removal attack.

6.3. Demonstration of Securing IP Cores using Double Line of Defense

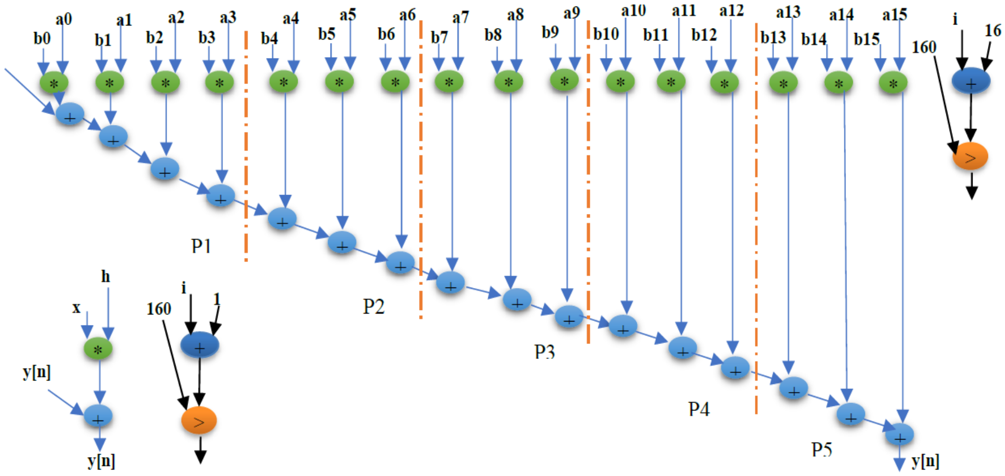
The demonstration of the proposed double line of defense is performed using FIR filter application. The following equation represents the 160-tap FIR filter:

$$Y[n] = \sum_{i=1}^{160} h[i] * X[n-i] \qquad (6.4)$$

Here, Y[n], h[i] and X[n-i] are the output, constant and input to FIR filter respectively, where variable ‘i’ is varied from 1 to 160. The corresponding DFG is shown in Fig. 6.2(a). The multi-key driven structural obfuscation and physical level watermarking based double line of defense is applied as follows:

(1) Applying multi-key based structural obfuscation on DFG of FIR filter

The multi-key based structural obfuscation is applied in following steps: (i) the DFG is unrolled based on UF=16 as shown in Fig. 6.2(b), to perform unrolling based structural transform. (ii) The unrolled DFG is partitioned into 5 partitions by applying 4 cuts. The applied cuts are highlighted in Fig. 6.2(b) using red dotted lines. (iii) Further, THT based structural obfuscation technique is performed on all five partitions as shown in Fig. 6.3 (note: ROE is not applicable due to absence of redundant nodes). (iv) All partitions are scheduled followed by applying folding transformation at four different instances (highlighted in red dotted ovals) of the scheduled obfuscated DFG as shown in Fig. 6.4.



(a) (b)

Fig. 6.2(a) DFG representing 160-tap FIR filter (b) loop unrolled FIR filter with UF=16

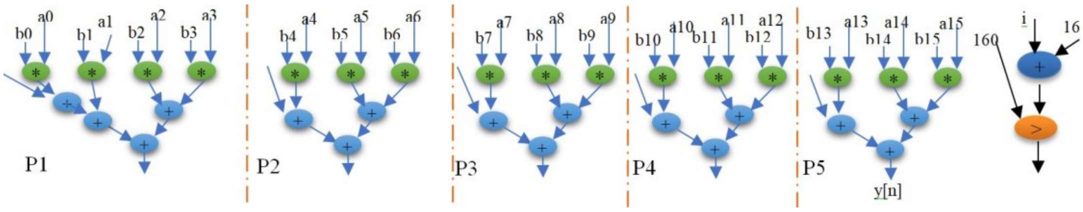
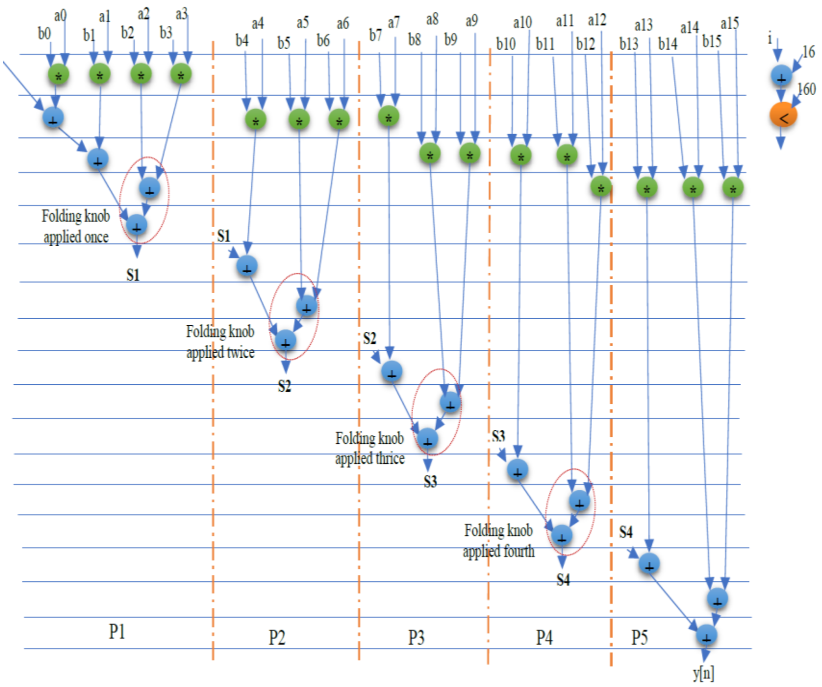


Fig. 6.3 Post applying THT based obfuscation in all partitions



a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15
i
16
160
Folding knob applied once
S1
Folding knob applied twice
S2
Folding knob applied thrice
S3
Folding knob applied fourth
S4
P1
P2
P3
P4
P5
y[n]

Fig. 6.4. Obfuscated scheduled FIR filter with applied folding at 4 instances

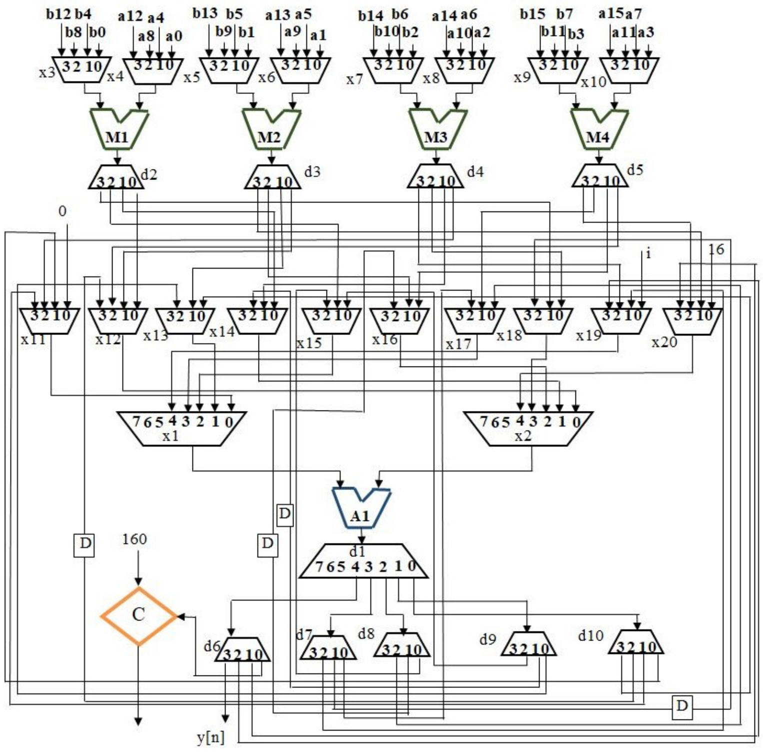


Fig. 6.5. Key-driven structurally obfuscated RTL datapath of FIR filter

Post performing various key driven techniques of structural obfuscation, the datapaths of individual partitions is synthesized followed by integrating into a single obfuscated RTL datapath of FIR filter core as shown in Fig. 6.5. Further, a following list ‘T’ of RTL modules (arranged in the decreasing order of their size) is extracted: T={M1, M2, M3, M4, d1, x1, x2, C, d2, d3, d4, d5, d6, d7, d8, d9, d10, x3, x4, x5, x6, x7, x8, x9, 10, x11, x12, x13, x14, x15,

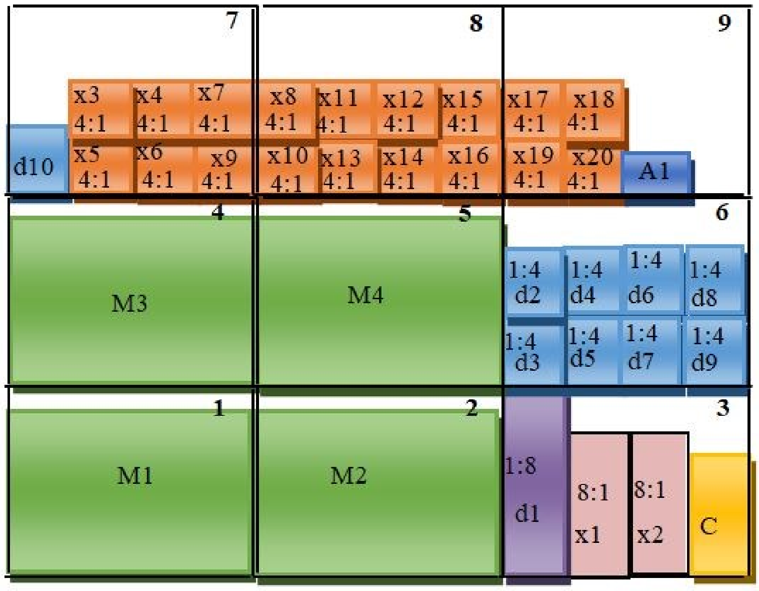


Fig. 6.6. Early floorplan of obfuscated FIR filter

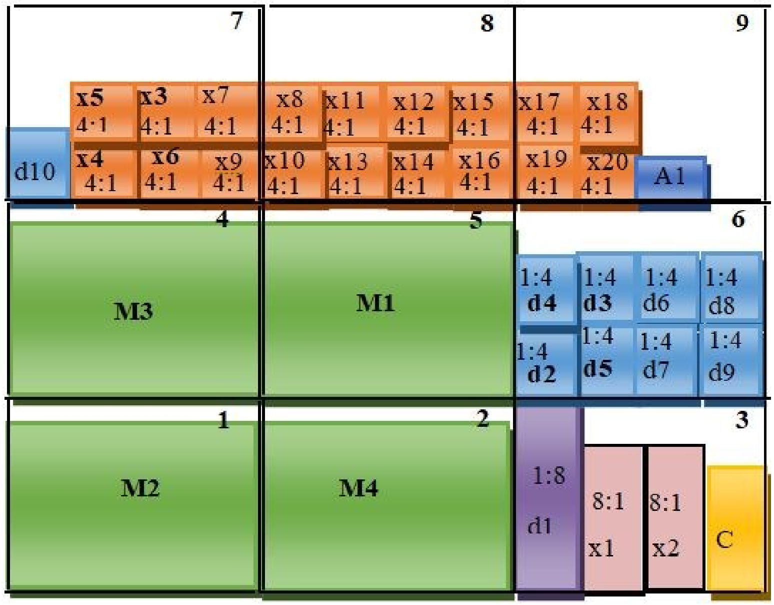


Fig. 6.7. Watermarked floorplan of obfuscated FIR filter (note: the change in the position of modules due to embedding signature is highlighted using bold)

x16, x17, x18, 19, x20, A1}. Using this list of RTL modules, an early floorplan is created as shown in Fig. 6.6.

(2) Applying physical level watermarking on structurally obfuscated FIR filter core

The physical level watermarking is applied on the early floorplan of the structurally obfuscated FIR filter core. Let's assume the designer's signature to be "αβαβγγ". As per the mapping of signature digits into watermarking constraints, the embedding of watermark is performed on early floorplan. Post embedding the aforementioned signature, the watermarked floorplan of structurally obfuscated FIR filter is shown in Fig. 6.7. Post obtaining the watermarked floorplan, the different phases of physical synthesis are performed to obtain an obfuscated and watermark embedded layout of FIR filter application.

6.4. Summary

This chapter discussed a double line of defense mechanism where multi-key driven structural obfuscation is applied to combat the potential backdoor insertion threat and physical level watermarking is applied to combat the IP piracy threat. In the double line of defense approach, the structural obfuscation technique and watermarking are applied during HLS process and physical synthesis process respectively. The metrics employed to measure the security using the double line of defense approach were also discussed in the chapter. Further, we demonstrated the process of generating secured IP (structurally obfuscated and watermarked) for the FIR filter application.

Chapter 7

Secured Hardware Accelerator Design Approach for Image Processing Filters

Image processing has important applications in extracting desired information from images to facilitate detection of objects and individuals etc. Moreover, the image processing applications find utility in security systems such as detection of license plates of vehicles, biometric fingerprinting, character recognition systems, robotics vision and healthcare systems etc. The various types of filters such as blurring, edge detection, and embossment etc are exploited to acquire useful information from an input image. In case of real time image processing, a general purpose processor cannot provide the desired performance in dealing with larger size of images. With the rapid evolution of digital imaging technology, image processing applications are progressively becoming computational-intensive due to increasing complexity of algorithm and larger image sizes. Thus, real-time processing of images entails expediting the performance [50]-[52]. Further, low power is also a constraint of mobile devices. Therefore, low power and high performance requirement encourages execution of image processing functions through a dedicated hardware accelerator.

Security of hardware accelerator filter design against reverse engineering (resulting into secret Trojan insertion) is highly relevant due to globalization of the chip design process involving offshore untrusted design houses and foundry. These offshore houses cannot be blindly trusted as an adversary may illegally reverse engineer the design netlist of the hardware accelerator filter design and insert Trojan secretly [8]. Thus Trojan infected filter designs could unknowingly be integrated in the products such as digital camera, cell phones, webcam etc., and therefore raising grave concern of consumers' safety. The consumers' safety and reliability due to Trojan can be compromised in one or more of the following forms: leakage of secret information, excessive heat dissipation, functional failure due to negative bias temperature instability (NBTI) stress, performance degradation, denial of service etc. [1], [2]. Employing structural obfuscation to the proposed hardware accelerator filter designs thwart reverse engineering (RE) and hence Trojan insertion by making

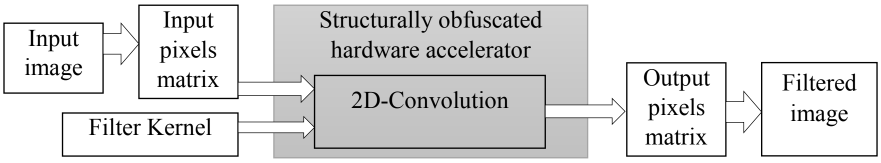


Fig. 7. 1. Generating filtered image using secured hardware accelerator of image processing filters

the design functionality and structure un-obvious to understand by an attacker. Further, employing structural obfuscation through high level transformation in high level synthesis (HLS) framework is aptly suitable for data intensive filter hardware accelerators designs as they rely on HLS framework. Fig. 7.1 highlights the process of generating filtered images using structurally obfuscated hardware accelerator for image processing filters.

This chapter discusses a novel approach of designing hardware accelerator architecture for image processing filters of 3×3 and 5×5 kernels using high level synthesis process. Further, structural obfuscation mechanism is added to the proposed approach for designing secured (structurally obfuscated) hardware accelerator architecture for both 3×3 and 5×5 filters. And, this chapter also discusses the designing of structurally obfuscated 3×3 filters for five specific image processing applications such as blurring, sharpening, vertical embossment, horizontal embossment and Laplace edge detection. Furthermore, a secured 3×3 filter design in reconfigurable functionality mode is presented where a specific image processing functionality can be configured using a control input. Outline of the chapter is as follows. The first section formulates the problem. The second section discusses the design approach of a secured 3x3 filter hardware accelerator under the following sub-sections: mathematical foundation of a 3x3 filter design, generating structurally obfuscated 3x3 filter hardware accelerator, designing re-configurable structurally obfuscated 3x3 filter architecture and generating secured five different application specific 3x3 filter designs. Further, the third section discusses the design approach of a secured 5x5 filter hardware accelerator under the following sub-sections: mathematical foundation of a 5x5 filter design and generating structurally obfuscated 5x5 filter hardware accelerator. Finally, the fourth section summarizes the chapter.

7.1. Problem Formulation

Given the mathematic functions of 3x3 and 5x5 filter of image processing applications, along with module library, resource constraint and the objective of securing the designs against potential hardware Trojan insertion attacks, generate secured (structurally obfuscated) hardware accelerators architecture.

7.2. Design Approach of a Secured 3x3 Filter Hardware Accelerator

Some image processing applications that involve 3×3 filter to produce filtered image are as follows: (i) blurring, (ii) edge detection, (iii) sharpening, (iv) vertical embossment and (v) horizontal embossment etc. This section discusses the mathematic foundation of a 3x3 filter processing followed by generating structurally obfuscated filter hardware accelerator in reconfigurable functionality mode. Further, we also discuss 3x3 filter designs for aforementioned five application specific image processing applications.

7.2.1. Mathematical foundation of a 3x3 filter design

An image of size I×J pixels can be represented using an I×J matrix $[A]_{(I\times J)}$ as follows.

$$A = \begin{bmatrix} X_{00} & X_{01} & \cdots & (J-1) \\ X_{10} & X_{11} & \cdots & X_{1(J-1)} \\ \vdots & \vdots & \ddots & \vdots \\ X_{(I-1)0} & X_{(I-1)1} & \cdots & \text{-1)}(J-1) \end{bmatrix}_{I\times J} \tag{7.1}$$

X_{ij} is a pixel value of the input image where i and j vary from 0 to I-1 and 0 to J-1 respectively. Further, a generic filter or kernel matrix of size n×m is represented by $[F]_{(n\times m)}$. For the kernel of size 3×3, the matrix [F] is given below:

$$F= \begin{bmatrix} f_{00} & f_{01} & f_{02} \\ f_{10} & f_{11} & f_{12} \\ f_{20} & f_{21} & f_{22} \end{bmatrix}_{3\times 3} \tag{7.2}$$

f_{pq} denotes kernel values, where p and q both vary from 0 to 2.

To compute the filter output, 2D-convolution is performed between input and kernel matrix. The 'same convolution' provides output image of same size as input image. To execute 'same convolution', the input matrix is padded with g number of zero-rows and zero-columns based on the following padding rule:

$$g = \frac{(f-1)}{2} \tag{7.3}$$

Where, f denotes the size of filter i.e. f=3 for 3×3 kernels. The enhanced input matrix, post padding, is shown below:

$$A = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & X_{00} & X_{01} & \cdots & {}_{)(J-1)} & 0 \\ 0 & X_{10} & X_{11} & \cdots & X_{1(J-1)} & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & X_{(I-1)0} & X_{(I-1)1} & \cdots & {}_{-1)(J-1)} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}_{N\times M} \tag{7.4}$$

Where, N×M is the dimension of enhanced input matrix which is given as (I+2)×(J+2) and a pixel value is represented by a_{ij} where i and j vary from 0 to N-1 and 0 to M-1 respectively.

Suppose [O] is an output matrix which contains the output of the same convolution between the input and kernel matrix. The dimension of [O] is given as (N-n+1)×(M-m+1). The following 'for loop' is used to compute the output matrix [O]:

$$\text{for } (w=0;\ w<(N-n+1) \times (M-m+1);\ w{+}{+})\{$$

$$O_w = \sum_{i,p=\text{lower value}}^{i,p=\text{upper value}} \Big(\sum_{j,q=\text{lower value}}^{j,q=\text{upper value}} a_{ij} \times f_{pq} \Big) \quad \} \tag{7.5}$$

For 3×3 filters, p and q take the values from 0 to 2 (i.e. lower value is 0 and upper value is 2) during calculating each output value O_w. And values of i and j vary from 0 to N-1 and M-1 respectively across the entire output matrix computation. During computation of each output value, lower and upper values of i and j vary in the window of 3 (because kernel matrix of size 3×3 slides over the modified input matrix). For example, for computing 1st output value O_0 using (7.5), lower and upper values of i and j are 0 and 2 respectively. Hence, O_0 is given as follows:

$$O_0 = \sum_{i=0,p=0}^{i=2,p=2} \Big(\sum_{j=0,q=0}^{j=2,q=2} a_{ij} \times f_{pq} \Big) \tag{7.6}$$

Further, this equation is expanded as follows:

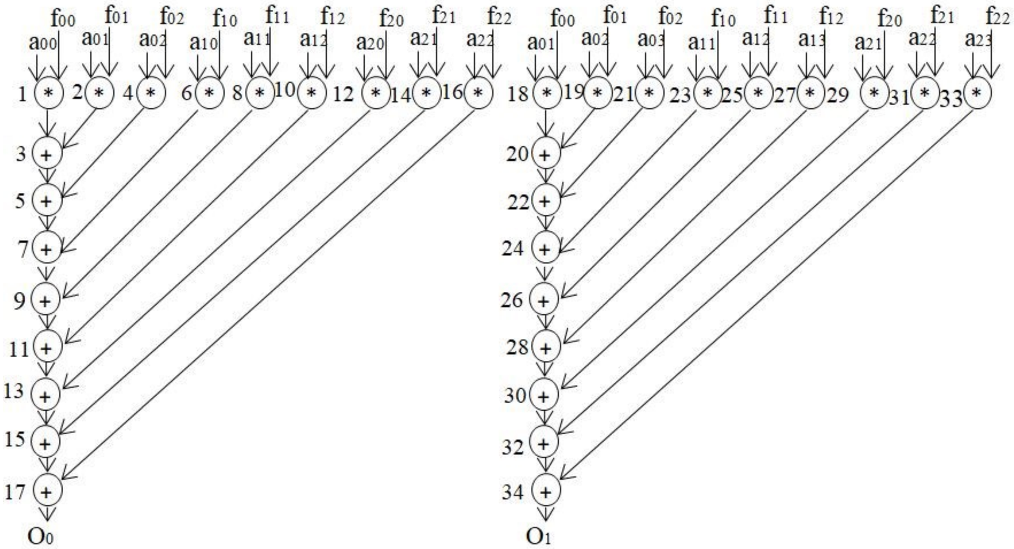

f00 f01 f02 f10 f11 f12 f20 f21 f22
a00 a01 a02 a10 a11 a12 a20 a21 a22
1 2 4 6 8 10 12 14 16
3 5 7 9 11 13 15 17
O0
f00 f01 f02 f10 f11 f12 f20 f21 f22
a01 a02 a03 a11 a12 a13 a21 a22 a23
18 19 21 23 25 27 29 31 33
20 22 24 26 28 30 32 34
O1

Fig. 7.2. Loop unrolled DFG of 3×3 image filter application

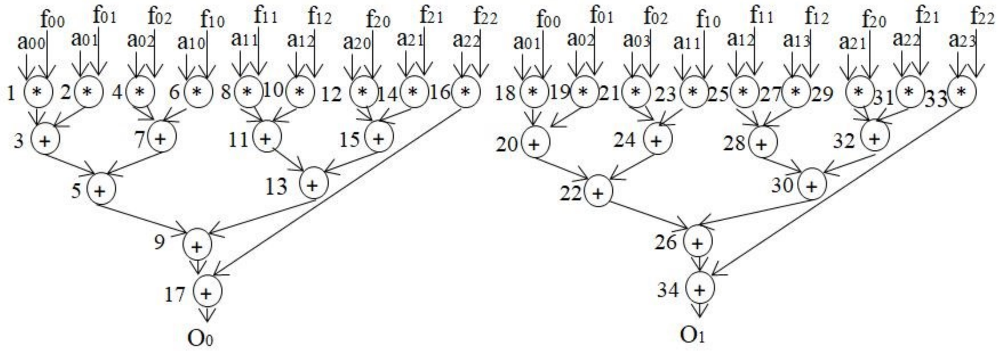

f00 f01 f02 f10 f11 f12 f20 f21 f22
a00 a01 a02 a10 a11 a12 a20 a21 a22
1 2 4 6 8 10 12 14 16
3 7 11 15
5 13
9
17
O0
f00 f01 f02 f10 f11 f12 f20 f21 f22
a01 a02 a03 a11 a12 a13 a21 a22 a23
18 19 21 23 25 27 29 31 33
20 24 28 32
22 30
26
34
O1

Fig. 7.3. THT obfuscated loop unrolled DFG of 3×3 image filter application

$$\begin{aligned} O_0 = {} & \left[\left(a_{00} \times f_{00}\right) + \left(a_{01} \times f_{01}\right) + \left(a_{02} \times f_{02}\right)\right] + \\ & \left[\left(a_{10} \times f_{10}\right) + \left(a_{11} \times f_{11}\right) + \left(a_{12} \times f_{12}\right)\right] + \\ & \left[\left(a_{20} \times f_{20}\right) + \left(a_{21} \times f_{21}\right) + \left(a_{22} \times f_{22}\right)\right] \end{aligned} \tag{7.7}$$

7.2.2. Generating structurally obfuscated 3x3 filter hardware accelerator

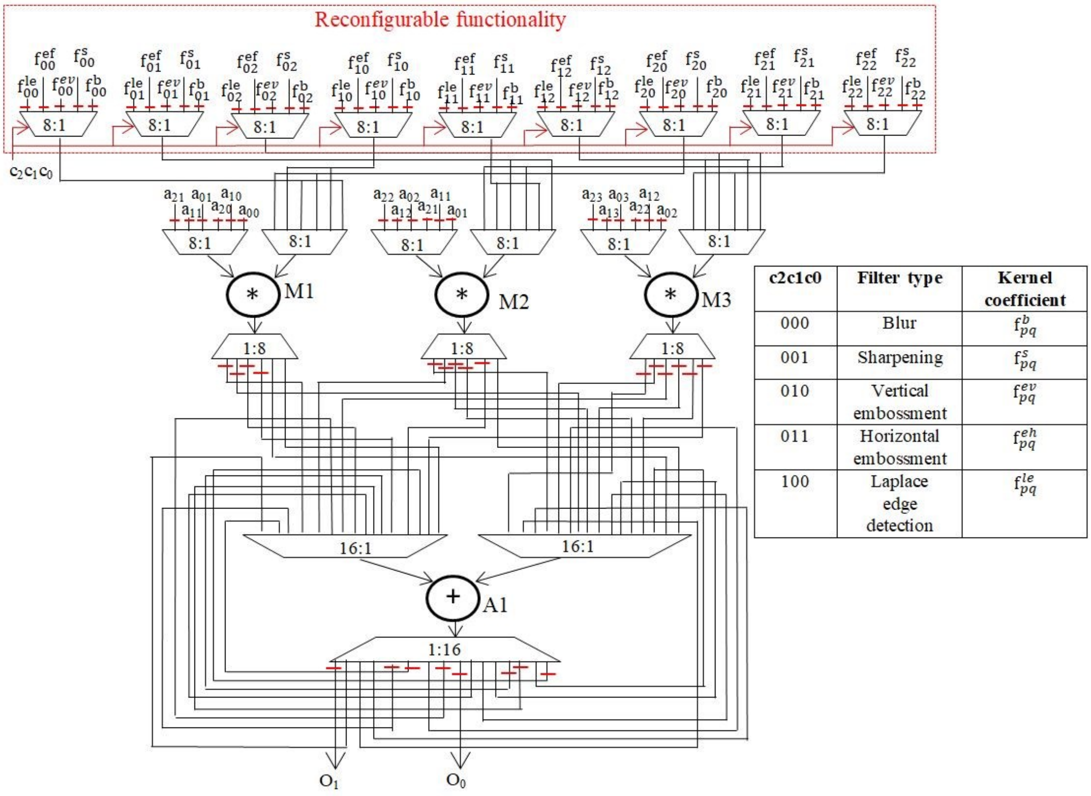
Using the eq. (7.7) as an algorithmic description of 3x3 filter application, the corresponding hardware accelerator can be designed in the form of RTL through the HLS process. Further in order to generate a secured 3x3 filter design, structural obfuscation mechanism is integrated with the HLS design process. Following two structural transformations are applied for obfuscation: (i) loop unrolling (ii) tree height transformation. In the loop unrolling based transformation, the ‘for loop’ represented in eq. (7.5) is unrolled twice. The loop unrolled DFG computes O_0 and O_1 concurrently as shown in Fig. 7.2, where the O_0 is computed using eq. (7.7) and O_1 is computed using the following equation:

$$\begin{aligned} O_1 = {} & \left[\left(a_{01} \times f_{00}\right) + \left(a_{02} \times f_{01}\right) + \left(a_{03} \times f_{02}\right)\right] + \\ & \left[\left(a_{11} \times f_{10}\right) + \left(a_{12} \times f_{11}\right) + \left(a_{13} \times f_{12}\right)\right] + \\ & \left[\left(a_{21} \times f_{20}\right) + \left(a_{22} \times f_{21}\right) + \left(a_{23} \times f_{22}\right)\right] \end{aligned} \tag{7.8}$$

Further, the unrolled DFG is applied with THT based obfuscation where some sequential addition operations are executed in parallel as shown in Fig. 7.3. These obfuscation techniques would impact the design structure by incurring the changes in the size and number of Muxes and Demuxes, interconnection path and storage resource count, thereby rendering the design structure non-interpretable by an attacker. This thwarts the adversary from performing reverse engineering (RE) and potential backdoor insertion. Further, the obfuscated DFG is subjected to scheduling phase of HLS using FU constraints of 3 multipliers (M) and 1 adder (A). Post perming HLS, we obtain a structurally obfuscated RTL design of 3x3 filter hardware accelerator.

7.2.2.1. Designing re-configurable structurally obfuscated 3x3 filter hardware accelerator architecture

The functionality of different 3×3 filter applications can be incorporated into a



Reconfigurable functionality
f^{ef}_{00} f^{s}_{00} f^{le}_{00} f^{ev}_{00} f^{b}_{00}
f^{ef}_{01} f^{s}_{01} f^{le}_{01} f^{ev}_{01} f^{b}_{01}
f^{ef}_{02} f^{s}_{02} f^{le}_{02} f^{ev}_{02} f^{b}_{02}
f^{ef}_{10} f^{s}_{10} f^{le}_{10} f^{ev}_{10} f^{b}_{10}
f^{ef}_{11} f^{s}_{11} f^{le}_{11} f^{ev}_{11} f^{b}_{11}
f^{ef}_{12} f^{s}_{12} f^{le}_{12} f^{ev}_{12} f^{b}_{12}
f^{ef}_{20} f^{s}_{20} f^{le}_{20} f^{ev}_{20} f^{b}_{20}
f^{ef}_{21} f^{s}_{21} f^{le}_{21} f^{ev}_{21} f^{b}_{21}
f^{ef}_{22} f^{s}_{22} f^{le}_{22} f^{ev}_{22} f^{b}_{22}
8:1
$c_2c_1c_0$
a_{21} a_{01} a_{10} a_{11} a_{20} a_{00}
a_{22} a_{02} a_{11} a_{12} a_{21} a_{01}
a_{23} a_{03} a_{12} a_{13} a_{22} a_{02}
M1
M2
M3
1:8
16:1
A1
1:16
O_1
O_0

c2c1c0	Filter type	Kernel coefficient
000	Blur	f^{b}_{pq}
001	Sharpening	f^{s}_{pq}
010	Vertical embossment	f^{ev}_{pq}
011	Horizontal embossment	f^{eh}_{pq}
100	Laplace edge detection	f^{le}_{pq}

Fig. 7.4. Structurally obfuscated hardware accelerator architecture for 3×3 image filter applications with reconfigurable functionality (note: the reconfigurable functionality is highlighted within a red box. Red hyphens show the registers needed for storing primary and intermediate inputs-outputs)

single obfuscated hardware accelerator by designing a reconfigurable architecture. The proposed reconfigurable architecture can be configured to execute functionality of five different image-processing applications viz. image blurring, sharpening, vertical and horizontal embossment and Laplace edge detection. A specific image processing hardware accelerator design is enabled using a control bit pattern.

In order to design the re-configurable structurally obfuscated 3x3 filter hardware accelerator architecture, the scheduled obfuscated DFG (obtained in previous sub-section 7.2.2) is subjected to allocation, binding and datapath synthesis phases of HLS. In this process, the reconfigurable functionality is enabled through the Muxes acting as switches for different kernel coefficient inputs. These Muxes are controlled through a bit pattern "c2c1c0" which has five different modes to execute five different applications. The re-configurable structurally obfuscated 3x3 filter hardware accelerator architecture along with five modes of "c2c1c0" is shown in Fig. 7.4.

7.2.3. Generating secured five application specific 3x3 filter designs

Using eq. (7.7) and eq. (7.8), the mathematical expression of different 3×3 filters of specific applications can be deduced based on their kernel matrix. This sub-section presents the secured hardware accelerator designs of following 3x3 filter applications:

7.2.3.1. 3x3 Blur filter design

The kernel matrix of a 3×3 mean filter for blurring is given below:

$$F^{B} = 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}_{3\times3} \tag{7.9}$$

Based on the kernel coefficients, the equation of concurrent output pair "O_0 and O_1" for the 3×3 blur filter is derived as follows:

$$O_0 = [(a_{00} + a_{01} + a_{02} + a_{10} + a_{11} + a_{12} + a_{20} + a_{21} + a_{22}) \times (1/9)] \tag{7.10}$$

$$O_1 = [(a_{01} + a_{02} + a_{03} + a_{11} + a_{12} + a_{13} + a_{21} + a_{22}+a_{23}) \times (1/9)] \tag{7.11}$$

In order to generate the structurally obfuscated 3x3 Blur filter design, the above equations are first converted into corresponding DFG representing the unrolled version. Further it is subjected to THT based obfuscation followed by HLS process to generate the structurally obfuscated RTL design of Blur filter. The end to end image processing using a secured Blur filter design is shown in Fig. 7.5.

7.2.3.2. 3x3 sharpening filter design

The kernel matrix of a 3×3 sharpening filter is given below:

$$F^{S}=\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}_{3\times 3} \tag{7.12}$$

Based on the kernel coefficients, the equation of concurrent output pair “O$_0$

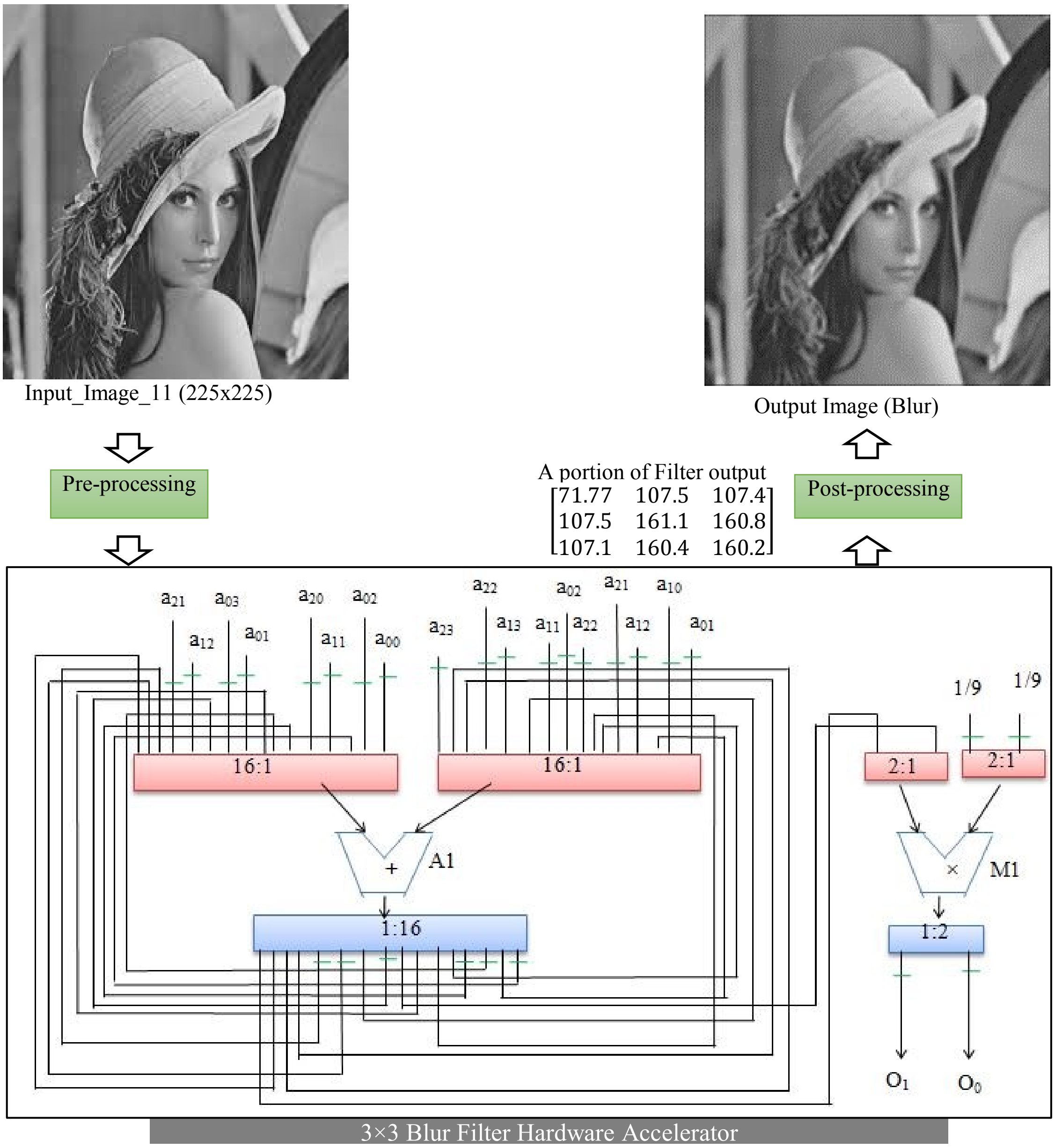


Fig. 7.5. End to end demonstration of image blurring application using 3x3 blur filter

and O$_1$” for the 3×3 sharpening filter is derived as follows:

$$O_0 = [(a_{00} + a_{01} + a_{02} + a_{10} + a_{12} + a_{20} + a_{21} + a_{22}) \times (-1)] + (a_{11} \times 9) \tag{7.13}$$

$$O_1 = [(a_{01} + a_{02} + a_{03} + a_{11} + a_{13} + a_{21} + a_{22} + a_{23}) \times (-1)] + (a_{12} \times 9) \tag{7.14}$$

In order to generate the structurally obfuscated 3x3 sharpening filter design,

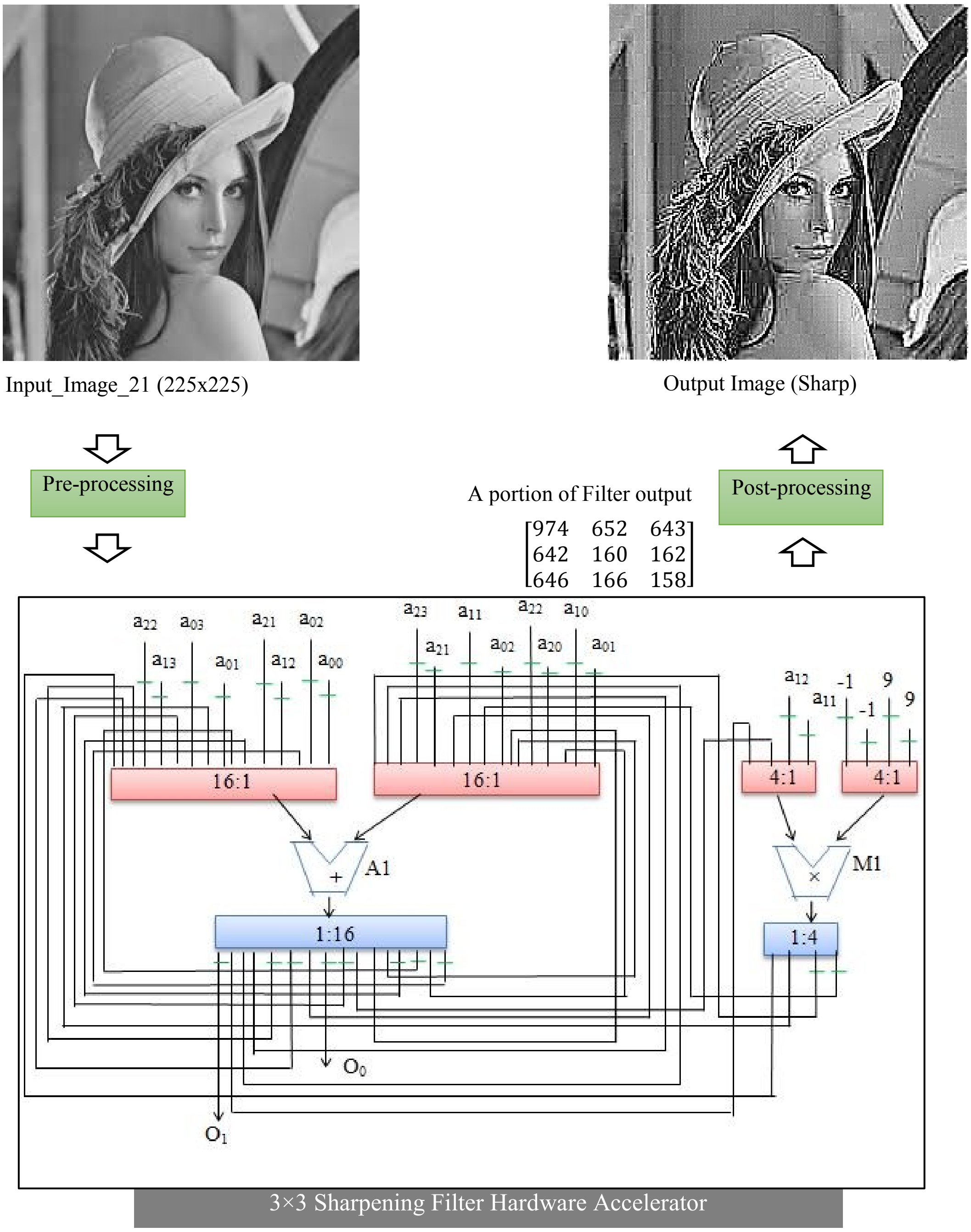


Fig. 7.6. End to end demonstration of image sharpening application using a 3x3 filter

the above equations are first converted into corresponding DFG representing the unrolled version. Further it is subjected to THT based obfuscation followed by HLS process to generate the structurally obfuscated RTL design of sharpening filter. The end to end image processing using secured sharpening filter design is shown in Fig. 7.6.

7.2.3.3. 3x3 vertical embossment filter design

The kernel matrix of a 3×3 vertical embossment filter is given below:

$$F^{VE} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}_{3\times 3} \tag{7.15}$$

Owing to non-symmetric nature of the kernel matrix of vertical embossment filter, a horizontal flip followed by a vertical flip are applied. This results into

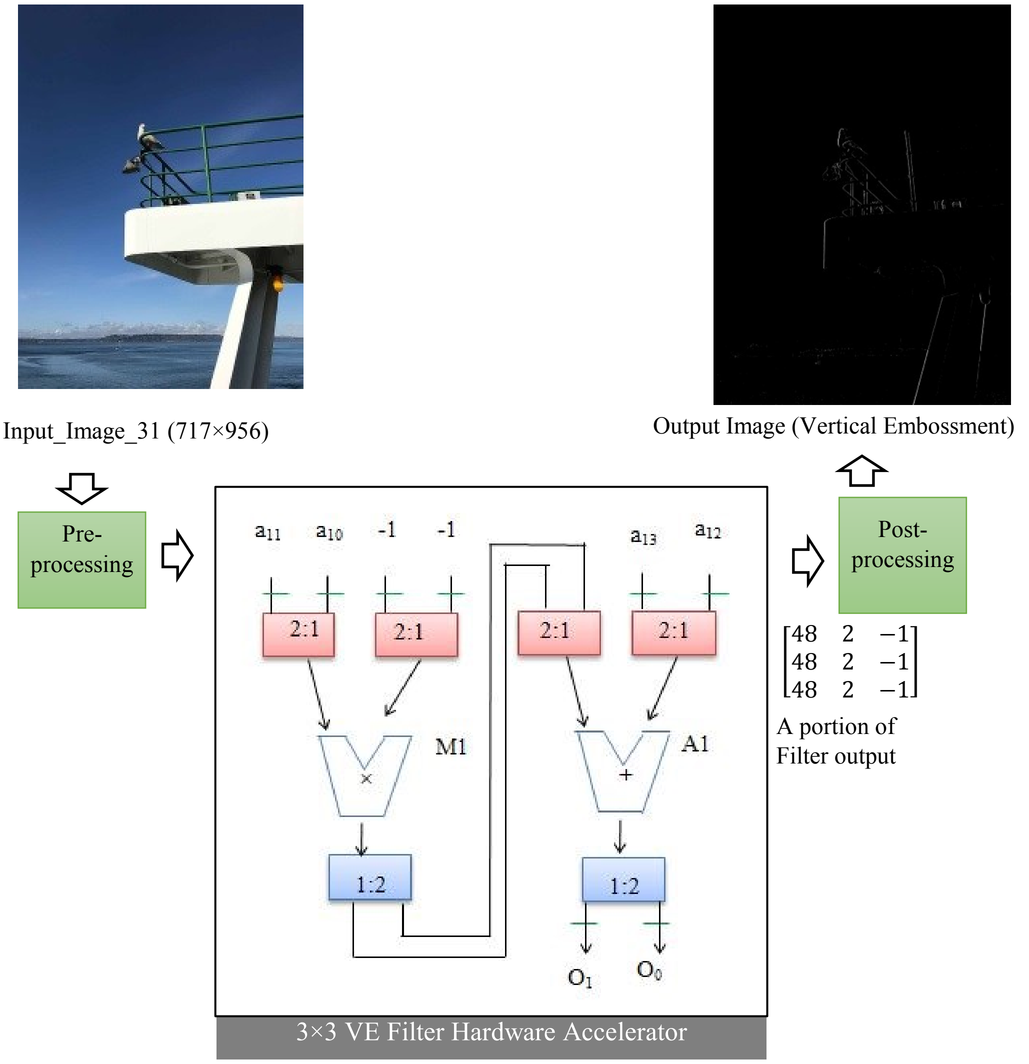


Fig. 7.7. End to end demonstration of vertical embossment application using 3x3 filter

the following kernel matrix of vertical embossment filter:

$$\mathrm{F}^{\mathrm{VE}}= \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}_{3\times3} \tag{7.16}$$

Based on the kernel coefficients, the equation of concurrent output pair "O_0 and O_1" for the 3×3 vertical embossment filter is derived as follows:

$$\mathrm{O}_0 = [(\mathrm{a}_{10} \times (-1))] + [(\mathrm{a}_{12})] \tag{7.17}$$

$$\mathrm{O}_1 = [(\mathrm{a}_{11} \times (-1))] + [(\mathrm{a}_{13})] \tag{7.18}$$

In order to generate the structurally obfuscated 3x3 vertical embossment filter design, the above equations are first converted into corresponding DFG. Further it is subjected to THT based obfuscation followed by HLS process to generate the structurally obfuscated RTL design of vertical embossment filter. The end to end image processing using secured vertical embossment filter design is shown in Fig. 7.7.

7.2.3.4. 3x3 horizontal embossment filter design

The kernel matrix of a 3×3 horizontal embossment filter is given below:

$$\mathrm{F}^{\mathrm{HE}}= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}_{3\times3} \tag{7.19}$$

Owing to non-symmetric nature of the kernel matrix of horizontal embossment filter, a horizontal flip followed by a vertical flip are applied. This results into the following kernel matrix of horizontal embossment filter:

$$\mathrm{F}^{\mathrm{HE}}= \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}_{3\times3} \tag{7.20}$$

Based on the kernel coefficients, the equation of concurrent output pair "O_0 and O_1" for the 3×3 horizontal embossment filter is derived as follows:

$$\mathrm{O}_0 = [(\mathrm{a}_{21})] + [(\mathrm{a}_{01} \times (-1))] \tag{7.21}$$

$$\mathrm{O}_1 = [(\mathrm{a}_{22})] + [(\mathrm{a}_{02} \times (-1))] \tag{7.22}$$

In order to generate the structurally obfuscated 3x3 horizontal embossment filter design, the above equations are first converted into corresponding DFG. Further it is subjected to THT based obfuscation followed by HLS process to generate the structurally obfuscated RTL design of horizontal embossment filter. The end to end image processing using secured horizontal embossment filter design is shown in Fig. 7.8.

7.2.3.5. 3x3 Laplace edge detection filter design

The kernel matrix of a 3×3 Laplace edge detection filter is given below:

$$F^{ED} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}_{3\times 3} \tag{7.23}$$

Based on the kernel coefficients, the equation of concurrent output pair "O_0 and O_1" for the 3×3 Laplace edge detection filter is derived as follows:

$$O_0 = [(a_{01} + a_{10} + a_{12} + a_{21}) \times (-1)] + (a_{11} \times 4) \tag{7.24}$$

$$O_1 = [(a_{02} + a_{11} + a_{13} + a_{22}) \times (-1)] + [(a_{12} \times 4)] \tag{7.25}$$

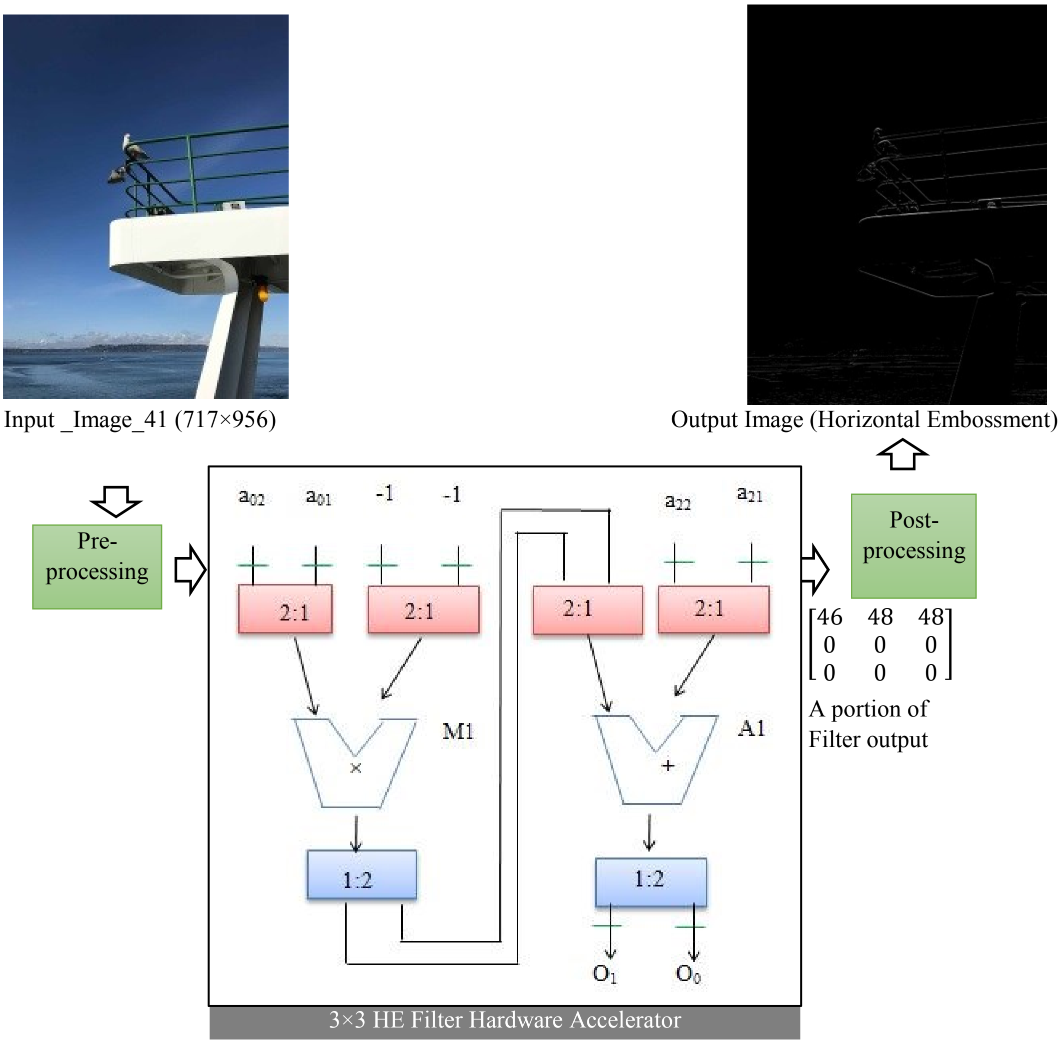

Input _Image_41 (717×956)
Output Image (Horizontal Embossment)
Pre-processing
a02
a01
-1
-1
a22
a21
2:1
2:1
2:1
2:1
M1
×
A1
+
1:2
1:2
O1
O0
Post-processing
46 48 48
0 0 0
0 0 0
A portion of Filter output
3×3 HE Filter Hardware Accelerator

Fig. 7.8. End to end demonstration of horizontal embossment application using 3×3 filter

In order to generate the structurally obfuscated 3x3 Laplace edge detection filter design, the above equations are first converted into corresponding DFG. Further it is subjected to THT based obfuscation followed by HLS process to generate the structurally obfuscated RTL design of Laplace edge detection filter. The end to end image processing using secured Laplace edge detection filter design is shown in Fig. 7.9.

7.3. Design Approach of a Secured 5x5 Filter Hardware Accelerator

Some image processing applications e.g. deep embossment uses 5×5 filter to produce filtered image. This section discusses the mathematic foundation of a 5x5 filter processing followed by generating structurally obfuscated filter

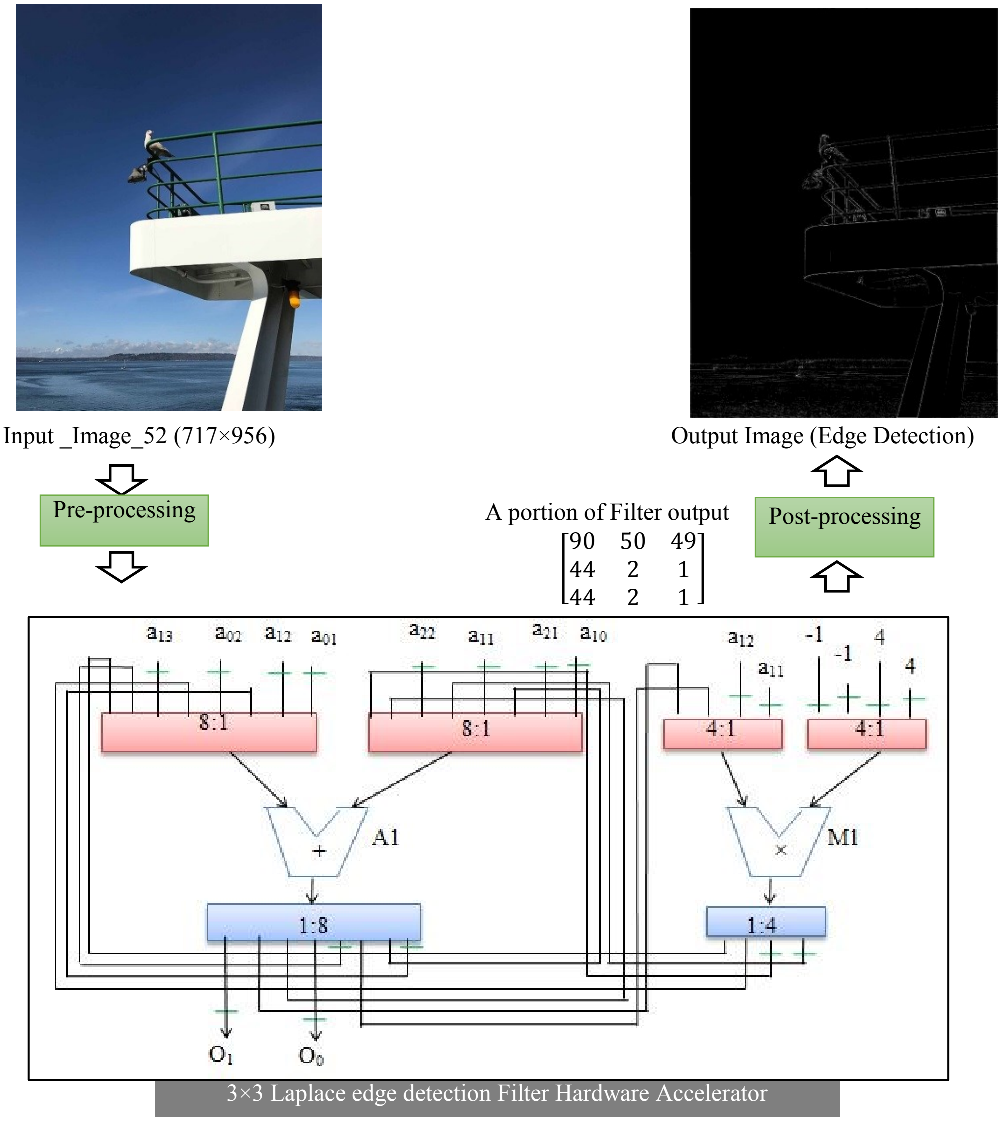

Input _Image_52 (717×956)
Output Image (Edge Detection)
Pre-processing
Post-processing
A portion of Filter output
90 50 49
44 2 1
44 2 1
a13 a02 a12 a01
a22 a11 a21 a10
a12 a11
-1 -1 4 4
8:1
8:1
4:1
4:1
A1
+
M1
×
1:8
1:4
O1
O0
3×3 Laplace edge detection Filter Hardware Accelerator

Fig. 7.9. End to end demonstration of edge detection using 3×3 edge detection filter

hardware accelerator.

7.3.1. Mathematical foundation of a 5x5 filter design

For the kernel of size 5×5, the filter coefficient matrix [F] is given below:

$$F = \begin{bmatrix} f_{00} & f_{01} & f_{02} & f_{03} & f_{04} \\ f_{10} & f_{11} & f_{12} & f_{13} & f_{14} \\ f_{20} & f_{21} & f_{22} & f_{23} & f_{24} \\ f_{30} & f_{31} & f_{32} & f_{33} & f_{34} \\ f_{40} & f_{41} & f_{42} & f_{43} & f_{44} \end{bmatrix}_{5\times 5} \qquad (7.26)$$

f_{pq} denote the kernel values, where p and q both vary from 0 to 4.

To execute 'same convolution' of 5×5 filter with the input image pixels, first the input matrix [A], given in (7.1), is padded with g number of zero-rows and zero-columns based on the padding rule given in eq. (7.3). Since f=5, therefor g is computed to be 2 using eq. (7.3). The enhanced input matrix, post padding, is shown below:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & X_{00} & X_{01} & \cdots & {}_{)(J-1)} & 0 & 0 \\ 0 & 0 & X_{10} & X_{11} & \dots & X_{1(J-1)} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & X_{(I-1)0} & X_{(I-1)1} & \cdots & {}_{\text{-}1)(J-1)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix}_{N\times M} \qquad (7.27)$$

Where, N×M is the dimension of enhanced input matrix which is given as (I+4)×(J+4) for executing the 'same convolution' using 5x5 filter. And the pixel values are represented by a_{ij} where i and j vary from 0 to N-1 and 0 to M-1 respectively.

The output pixels generated using convolution of updated input matrix [A] and 5x5 filter kernel [F] can be calculated using the 'for loop' given in eq. (7.5), where p and q take the values from 0 to 4 (i.e. lower value is 0 and upper value is 4) during calculating each output value O_w. And values of i and j vary from 0 to N-1 and M-1 respectively for computing the the entire output matrix [O]. For computing 1st output value O_0 using (7.5), lower and upper values of i and j are 0 and 4 respectively. Hence, O_0 is given as follows:

$$O_0 = \sum_{i=0,p=0}^{i=4,p=4} \left(\sum_{j=0,q=0}^{j=4,q=4} a_{ij} \times f_{pq} \right) \qquad (7.28)$$

Further, this equation is expanded as follows:

$$\begin{aligned} O_0 = &\left[\left(a_{00} \times f_{00}\right) + \left(a_{01} \times f_{01}\right) + \left(a_{02} \times f_{02}\right) + \left(a_{03} \times f_{03}\right) + \left(a_{04} \times f_{04}\right)\right] \\ &+\left[\left(a_{10} \times f_{10}\right) + \left(a_{11} \times f_{11}\right) + \left(a_{12} \times f_{12}\right) + \left(a_{13} \times f_{13}\right) + \left(a_{14} \times f_{14}\right)\right] \\ &+\left[\left(a_{20} \times f_{20}\right) + \left(a_{21} \times f_{21}\right) + \left(a_{22} \times f_{22}\right) + \left(a_{23} \times f_{23}\right) + \left(a_{24} \times f_{24}\right)\right] \\ &+\left[\left(a_{30} \times f_{30}\right) + \left(a_{31} \times f_{31}\right) + \left(a_{32} \times f_{32}\right) + \left(a_{33} \times f_{33}\right) + \left(a_{34} \times f_{34}\right)\right] \\ &+\left[\left(a_{40} \times f_{40}\right) + \left(a_{41} \times f_{41}\right) + \left(a_{42} \times f_{42}\right) + \left(a_{43} \times f_{43}\right) + \left(a_{44} \times f_{44}\right)\right] \end{aligned} \tag{7.29}$$

7.3.2. Generating structurally obfuscated 5x5 filter hardware accelerator

Using the eq. (7.29) as an algorithmic description of 5x5 filter application, the corresponding hardware accelerator can be designed in the form of RTL through the HLS process. Further in order to generate a secured 5x5 filter design, structural obfuscation mechanism is integrated with the HLS design

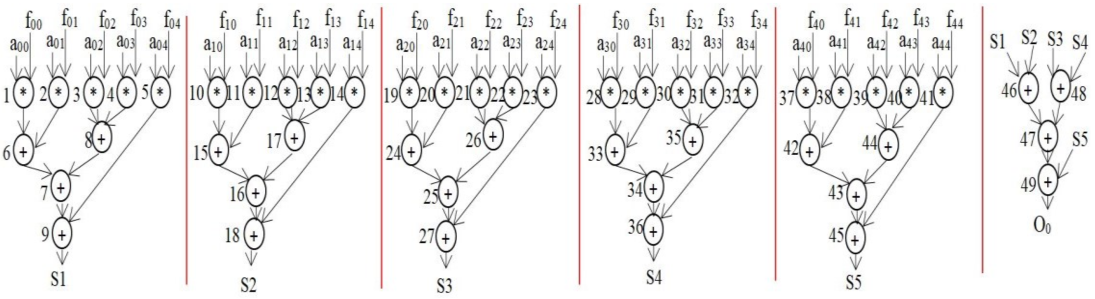


Fig. 7.10. Partitioned DFG of 5×5 filter application post performing THT based structural obfuscation

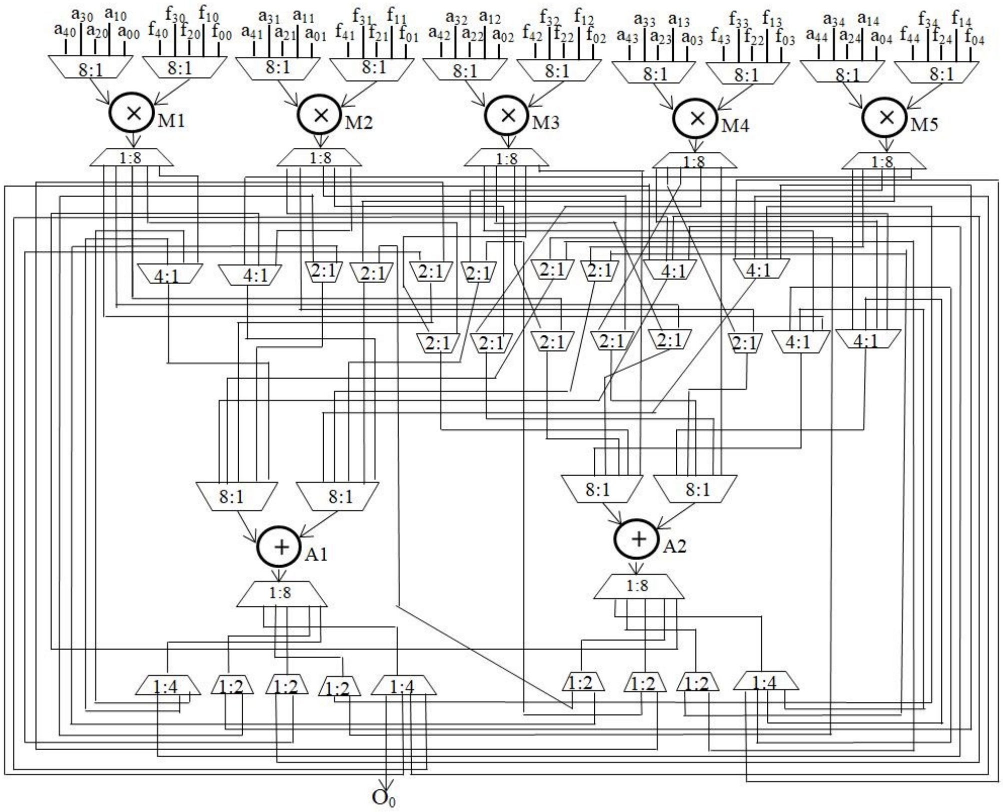


Fig. 7.11. Structurally obfuscated RTL datapath of 5×5 image filter hardware accelerator

process. Following two structural transformations are applied for obfuscation: (i) DFG partitioning (ii) tree height transformation. In the DFG partitioning based structural obfuscation technique, five cuts are made to create 6 partitions. Further, each partition of the DFG is applied with THT obfuscation. Thus obtained obfuscated partitioned DFG of 5x5 filter application is shown in Fig. 7.10. Further, FU constraints of 5M and 2A are applied to perform scheduling of obfuscated DFG. Finally, structurally obfuscated RTL datapath of 5x5 filter application is synthesized to generate the secured hardware accelerator as shown in Fig. 7. 11.

7.4. Summary

This chapter discussed a novel appproach of designing 3x3 and 5x5 filter hardware accelrators for image processing applications using HLS design process, to address the low power and high performance requiremrnt. Further, the threat of RE based hardware Trojan insertion attack was handled by employing the structural obfuscation during the HLS design process of image processing filters. The proposed structurally obfuscated 3x3 and 5x5 filter designs are resilient against the RE by an attacker. Furthermore, we also demonstrated the image processing applications for five specific structurally obfuscated filters.

Chapter 8

Techniques for Securing Functionally Obfuscated DSP Cores against Removal Attack

The IP core steganography and watermarking techniques, discussed in the previous chapters, act as detective control against IP piracy. However, these approaches cannot prevent IP piracy from happening. A functional obfuscation mechanism [36] is a preventive control against IP piracy, where the functionality is obfuscated by locking the design using some key gates or locking blocks. Thus the functional obfuscation technique produces a locked netlist which can only be activated by applying a correct key [37]. The adversary, being unaware of the correct key, cannot illegally use the IP core. For securing digital signal processing (DSP) cores, functional obfuscation has been performed by adding IP core locking blocks (ILBs) at the output of functional units (FUs) [36]. However, a functionally obfuscated design is susceptible to removal attack [39] where the attacker attempts to remove the ILBs in order to de-obfuscate it and acquire an unlocked netlist. Thus removal attack can defeat the goal of functional obfuscation and facilitate IP piracy.

This chapter discusses proposed SHA-512 based key generation hardware and anti-removal logic (ARL) based key generation unit to secure the ILBs used in functionally obfuscated DSP cores against the removal attack. The security is achieved by offering re-configurability to ILBs structures based on the output of proposed key generation hardware units. The outline of the chapter is as follows. The first section formulates the problem. The second section provides the overview of an ILB and its various features. The third section discusses the SHA-512 based ILB-keys generation hardware under the following sub-sections: overview, design of custom SHA-512 based key generation hardware, key based reconfiguration of ILB structure and advantages of using SHA-512 based ILBs-key generation hardware over AES hardware [53] to secure against the removal attack. Further, the fourth section discusses the ARL unit under the following sub-sections: overview, design of ARL unit based key generation hardware, and advantage of using ARL unit over AES and SHA-512 based logic to secure against removal attack. Finally, the fifth section summarizes the chapter.

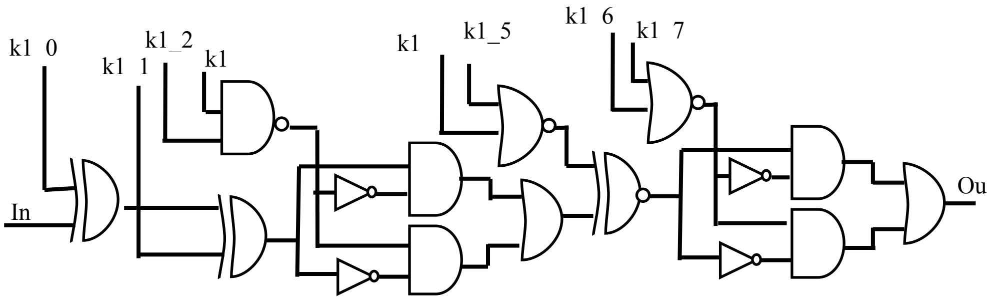


Fig. 8.1. A sample ILB structure requiring an 8-bit key 'k1' to activate

8.1. Problem Formulation

Given the data flow graph (DFG) representation of a target application, module library, resource constraints, along with the objective of securing IP cores against IP piracy, generate a removal attack resilient functionally obfuscated DSP IP core.

8.2. Overview of ILB and its Features

An ILB is interweaved structure of various logic gates such as XOR, XNOR, AND, NAND, and NOT gates. A sample ILB structure is depicted in Fig. 8.1. An ILB requires 8-bit long key to be activated. The ILBs are placed into the design at output bit of each FU such as multiplier and adder to enable logic locking or functional obfuscation of DSP cores. The various ILB structures are placed into the gate level design through a repetition pattern of a particular ILB. A random variable μ is used to achieve this repetition pattern, where μ is given as follows: $1 \leq \mu \leq T_{ILB}$. More explicitly, same ILBs is placed 'μ' times and post 'μ' repetitions, next ILB from total ILBs (T_{ILB}) is chosen and placed 'μ' times. The total ILBs T_{ILB} are determined as follows: (total number of FUs) × (# of output bits of a FU).

The ILBs exhibit following security properties:

(i) Multi-pairwise security: The ILBs are multi-pair wise secured because one bit of key cannot be sensitized to output without monitoring remaining key bits.

(ii) Protected against key gate isolation: Since the keys gates are associated in such a manner that one gate is linked with the key inputs of other gates; hence an individual key gate of ILBs cannot be isolated. Thus, the key gate isolation is prohibited to thwart the key sensitization attack.

(iii) Protected against run of key-gates: The keys gates have been intertwined in order to thwart the replacement of run of key gates with single key gate by an adversary, hence hindering key sensitization.

(iv) Non-mutable key gates: The robust composition of keys gates prevents against muting them without monitoring all the key bits hence hindering key sensitization.

(v) Thwarting IP piracy: The locked or functionally obfuscated netlist cannot be misused by an adversary to make illegal income, without knowing or deducing the correct key. Since the locking using ILBs involve a larger size key (making brute force almost infeasible) and it is also protected against key sensitization, therefore the correct key value cannot be recovered by the adversary.

8.3. SHA-512 based ILB-keys Generation Hardware to Secure against Removal Attack

8.3.1. Overview of SHA-512 based ILBs-key generation hardware

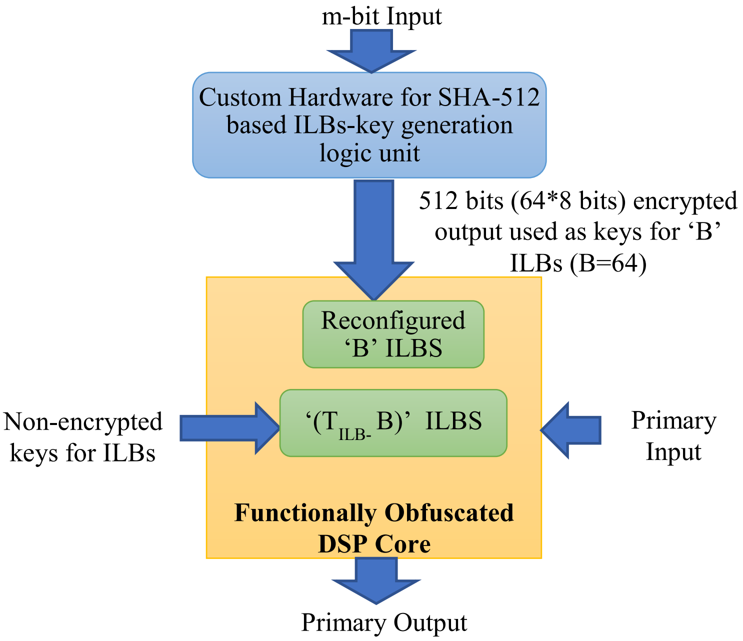

m-bit Input
Custom Hardware for SHA-512 based ILBs-key generation logic unit
512 bits (64*8 bits) encrypted output used as keys for ‘B’ ILBs (B=64)
Reconfigured ‘B’ ILBS
‘(T_ILB- B)’ ILBS
Non-encrypted keys for ILBs
Primary Input
Functionally Obfuscated DSP Core
Primary Output

Fig. 8.2. Overview of securing a functionally obfuscated DSP core against the removal attack using SHA-512 based ILBs-key generation hardware

The overview of employing security using SHA-512 based ILBs-key generation hardware against the removal attack on a functionally obfuscated design is shown in Fig. 8.2. As shown, the custom hardware of SHA-512 based ILBs-key generation logic takes arbitrary length (m bits) input to produce output of B*8 bits, where 'B' denotes the count of ILBs to be reconfigured and '8' denotes the size of key (in bits) per ILB. Since the overall length of the hash digest is 512-bit and one ILB requests an 8-bit key, therefore 'B' can have the maximum value of 64. More explicitly, upto 64 ILBs of the functionally obfuscated design can be reconfigured using the 512-bit hash digest generated from the SHA-512 based key generation hardware. And, the keys of remaining ILBs can be fed without encryption as shown in the Fig. 8.2. Obtaining ILB keys using SHA-512 based key generation logic offers strong structural reconfiguration of several ILBs simultaneously. This is because, an ILB structure is not fixed rather it is determined by the key generation logic and its arbitrary input of m-bit. Moreover, the architecture of custom SHA-512 based key generation hardware, its m-bit input and hash output are not known to the attacker. This thwarts the identification of any particular ILB in the obfuscated design by an attacker, thus securing against the removal attack.

The custom SHA-512 based key generation logic is shown in Fig. 8.3. The following two units participate in the logic: (i) SHA-512 custom logic (ii) ILB keys-extraction logic. An m-bit string (can be chosen arbitrarily) and initial hash buffer values are given as input to the SHA-512 custom unit and 512-bit hash digest is produced at the output. The initial values (standard) of eight

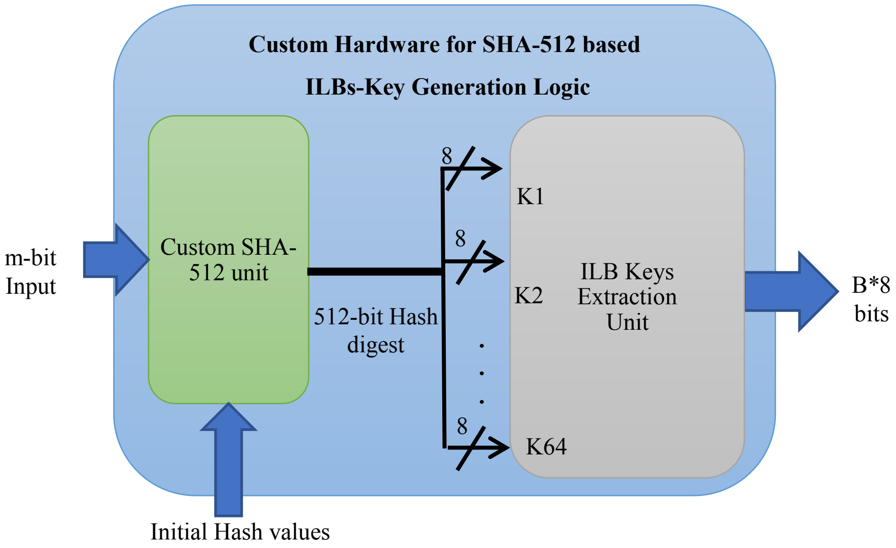


Fig. 8.3. Block diagram of custom hardware for SHA-512 based ILBs-key generation logic

hash buffers (a, b, c, d, e, f, g and h), each of size 64 bits, are given below:

a → "6A09E667F3BCC908", b → "BB67AE8584CAA73B",

c → "3C6EF372FE94F82B", d→ "A54FF53A5F1D36F1"

e → "510E527FADE682D1", f → "9B05688C2B3E6C1F"

g → "1F83D9ABFB41BD6B", h → "5BE0CD19137E2179"

The custom SHA-512 unit first translates the m-bit input into a chunk of size 1024-bit, followed by processing of 1024-bit chunk and eight hash buffer values to produce updated values of the hash buffers in each round. Instead of choosing fixed 80 rounds of traditional SHA-512, the number of rounds of execution is customized based on designer's choice. The final round produces the 512-bit digest contained in the eight hash buffers. Thereafter, the ILB keys-extraction unit divides the 512-bit hash digest into the 8-bits long bitstreams that function as encrypted keys for ILBs. Based on the count of ILBs to be reconfigured, the corresponding number of bitstreams can be obtained from the 512-bit digest. Hence by extracting B*8 bits using the key-extraction unit, 'B' number of ILBs of a functionally obfuscated DSP core can be structurally reconfigured.

8.3.2. Design of custom SHA-512 based key generation hardware

Figure 8.4 shows the internal steps of the custom SHA-512 based key generation logic. The process of generating ILB-keys is elaborated below using the following steps:

1. Appending padding bits: The input bitstream of random length (e.g. m bits) is padded with the following bit sequence "1000…0" to enhance the length upto 896-bit (considering the length of input bitstream is less than 896 bits).
2. Appending input bitstream length: The length of obtained 896-bit long chunk is further enhanced to a chunk of 1024-bit by padding the 128 bits of the length of initial input bitstream.
3. Word extraction: The 64-bit words are extracted to be exploited by the round function computation (RFC) logic as shown in the Fig. 8.4. The word is either taken out from the 1024-bit chunk or derived using 'word

computation logic' (same as standard SHA algorithm). For the first 16 rounds, 64-bit words (W0-W15) are taken out from the 1024-bit chunk. For the 17th round onwards, a 64-bit word is derived from the 'word computation logic'.

4. Constant extraction: The custom SHA-512 logic also exploits sixteen 64-bit constants (K0, k1,…, K15) which are all standard values. One constant is extracted from the set of constants for each round of RFC logic.
5. Hash buffer processing: The first round of RFC uses the initial hash buffer.

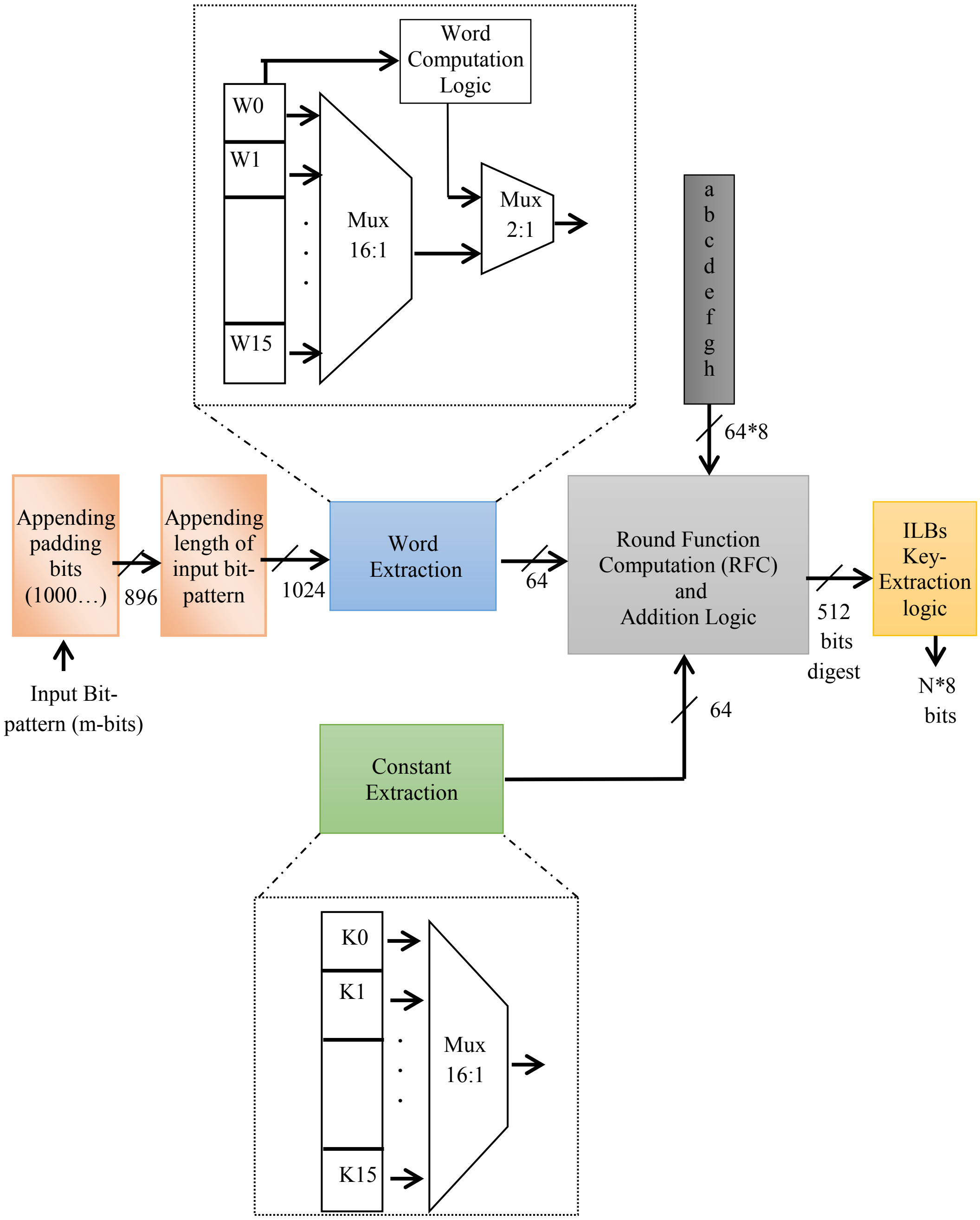

Word Computation Logic
W0
W1
W15
Mux 16:1
Mux 2:1
a
b
c
d
e
f
g
h
64*8
Appending padding bits (1000…)
896
Appending length of input bit-pattern
1024
Word Extraction
64
Round Function Computation (RFC) and Addition Logic
512 bits digest
ILBs Key-Extraction logic
N*8 bits
Input Bit-pattern (m-bits)
64
Constant Extraction
K0
K1
K15
Mux 16:1

Fig. 8.4. Block diagram of custom SHA-512 based key generation hardware

Further, the updated hash buffer values are generated in the subsequent rounds of RFC.

6. Round Function Computation (RFC): the processing of round function exploits the output from the previous 3 steps viz. word extraction, constant extraction and hash buffer. The RFC logic executes the following six elementary functions: MAJORITY function, SUMMATION/ ROTATION ‘a’, CHOOSE function, SUMMATION/ ROTATION ‘e’, MIX function-1, MIX function-2. The hardware of RFC alongwith the logic of each of six functions is shown in Fig. 8.5. The number of rounds (customized) of the

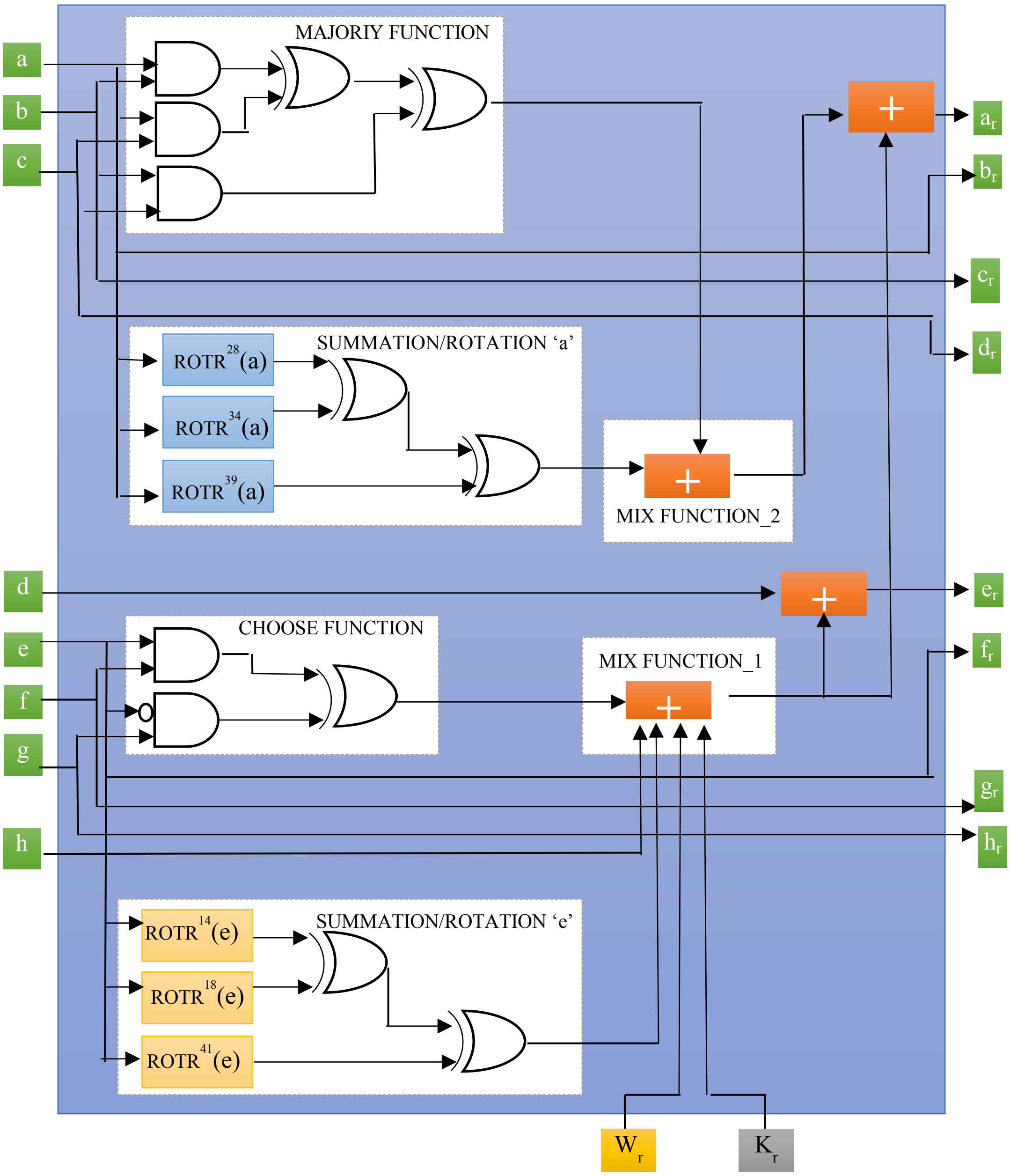


Fig. 8.5. Round Function Computation hardware of custom SHA-512, where a_r to h_r indicate updated hash buffer values, W_r and K_r indicate 64-bit word and constant respectively in the round ‘r’

RFC logic is decided by the designer. The hash buffer values obtained post final round indicate the 512-bit hash-digest.

7. ILBs keys extraction: The ILBs key extraction unit uses the 512-bit hash-digest to extract the B*8 bits that act as keys for 'B' number of ILBs of a functionally obfuscated design. Using the 512-bit digest, upto 64 ILBs can be structurally reconfigured by reordering their internal gates such that the functionality of ILBs is not affected and the output of ILBs is correct only on applying actual key bits.

The integration of the custom SHA-512 based ILBs-key generation hardware with a functionally obfuscated DSP core is performed by connecting its output to the key bits of ILBs. Thus generated the functionally obfuscated DSP design is resilient against the removal attack.

8.3.3. Key based reconfiguration of ILB structure

The structural reconfiguration of ILBs using the key bits generated from SHA-512 based key generation hardware is shown in Fig 8.6. Since an ILB needs 8-bit key to be activated, therefore an 8-bit block is extracted from the 512-bit output of the SHA-512 based key generation logic to reconfigure the ILB. To obtain the keys for 'B' ILBs, B*8 bits are extracted by the designer. For demonstration, a single ILB structure is configured based on the following 8-bit key value obtained from the ILB key selection unit: "10110000". Thereby, different 'B' number of ILBs can be structurally reconfigured based on the key bits by reordering the internal gates in an appropriate manner.

8.3.4. Advantage of using SHA-512 based ILBs-key generation hardware over AES hardware to secure against removal attack

Following cryptographic properties of SHA-512 based key generation hardware makes its suitable for producing keys for the ILBs: (a) collision resistance makes difficult to find two such inputs that produce same hash output (b) one-way random functionality ensures that the input message cannot be derived from the output hash (c) deterministic property means that the same hash value is always generated for a given input and (d) each bit of the digest is a function of every input bit. These security properties of hashing prevent an attacker to find actual input pattern from output bits. Further, SHA-

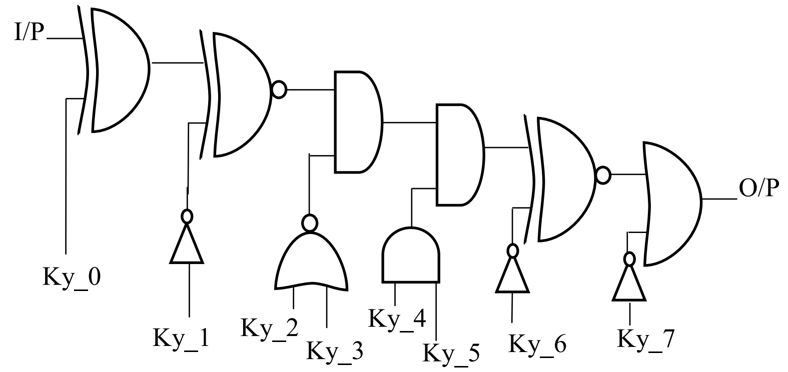


Fig. 8.6. Example of a configured ILBy based on the output "10110000" produced from ILB key selection/extraction logic

512 based ILBs-key generation logic offers the following advantages over AES based logic:

1. AES based key generation logic relies upon fixed secret key for generating the encrypted output bits that act as key bits for ILB reconfiguration. However the fixed secret key is vulnerable to standard side channel attacks, hence can be compromised by the attacker. On the contrary, the proposed 'SHA-512 based key generation logic' does not depend on fixed secret key to produce the key bits for ILBs reconfiguration, thereby rejecting the likelihood of side channel attack.
2. AES based logic has capability to provide key-bits to maximum 16 ILBs using its 128-bit output (since 16*8=128, where each ILB requires 8-bit key). This prevents a designer from reconfiguring more than 16 ILBs using AES128 for larger designs (containing several ILBs). This would demand more than one instance of AES in the obfuscated design to enable reconfiguration of more ILBs concurrently, hence resulting in excessive design and power overhead. On the contrary, the proposed 'SHA-512 based key generation logic' can generate keys for upto 64 ILBs using its 512-bit hashed output. This leads to lower design and power overhead compared to AES based logic.
3. Since the AES based logic offers reconfiguration of maximum16 ILBs only, hence provides less security against removal attack. On the contrary, the proposed SHA-512 based logic offers reconfiguration of maximum 64 ILBs in its single execution, thus employs higher security compared to

AES for securing functionally obfuscated DSP designs against the removal attack.

8.4. Anti-removal Logic (ARL) Unit to Secure against Removal Attack

8.4.1. Overview of lightweight ARL based ILBs-key generation hardware

As we discussed earlier, removal attack on ILBs can nullify the goal of functional obfuscation. The removal attack can be handled by making ILBs structure undetectable to an adversary. To make the ILBs undetectable, reconfigured ILBs can be used in place of standard ILB structures. The structural reconfiguration of ILBs can be performed by producing their keys from AES128 and SHA-512 based logic. However, SHA-512 and AES128 based ILBs-key generation hardware result in substantial design overhead and provide lesser security against removal attack in terms of number of input bits to be decoded to find the ILB key. Here, we discuss a custom lightweight and highly secure solution against the removal attack using an anti-removal (ARL)

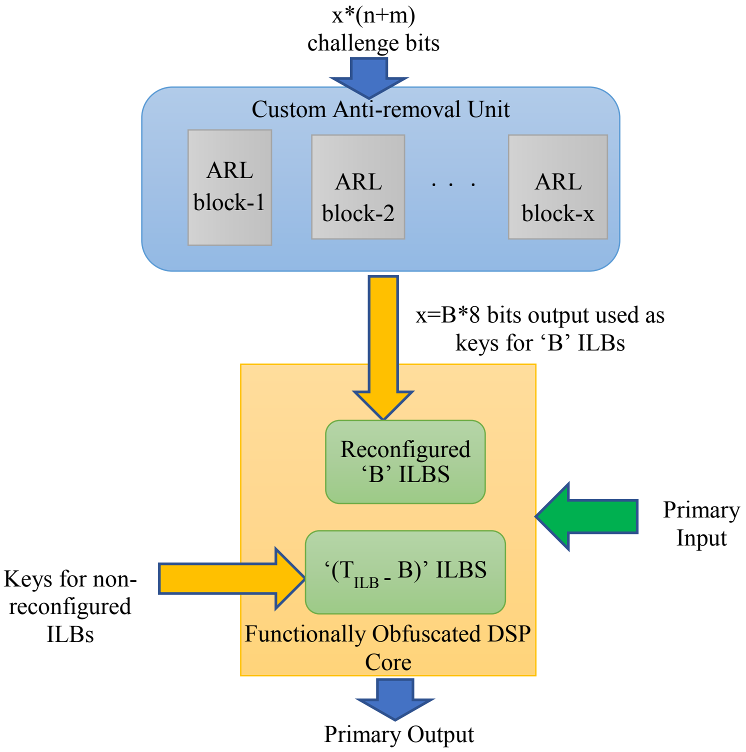


Fig. 8.7 Overview of using ARL unit to protect a functionally obfuscated DSP core against the removal attack

unit. The key-bits for ILBs of a functionally obfuscated design are produced using the ARL unit. Thus produced key-bits determine the structural configuration of ILBs to enable the security against removal attack.

The security of a functionally obfuscated design using ARL unit is highlighted in Fig. 8.7. As shown, an ARL unit is a composition of several ARL blocks. The number of ILBs that the designer has to reconfigure decides the number of ARL blocks used in an ARL unit. For example, the designer targets to reconfigure 'B' ILBs where each requires an 8-bit key to be activated. Since each ARL block produces only one bit of ILB key, hence total B*8 ARL blocks are required to be employed in the ARL unit. Because an adversary is not known with the custom ARL unit design, he/she cannot recover the keys of reconfigured ILBs. Hence the adversary is not able to deduce the ILB structures based on 8-bit keys. This makes the ILBs un-identifiable for the adversary, thereby hindering the removal attack. Additionally, the ARL unit itself remains protected against the removal attack as its architecture is customizable and also not recognizable to the attacker. Moreover, further camouflaging of the internal gates of ILBs and ARL unit in the entire design takes place after synthesis. This renders the identification of ILBs by the adversary highly challenging, thus improving the security against the removal attack.

8.4.2. Design of ARL based key generation hardware

Multiple ARL blocks are used to constitute an ARL unit, where each ARL block generates only a single key-bit for ILBs. For generating 'x' number of key-bits, 'x' ARL blocks are employed in an ARL unit, where x=B*8. Here, B indicates the count of ILBs to be reconfigured. Fig. 8.8 depicts the internal architecture of an ARL block. As depicted in the figure, an ARL block is a combination of multiple AND-OR networks which produce a single bit of output. The size and number of AND-OR networks is decided by the following two sets of the challenge inputs: S = {S0, S1, …, Sn} and C = {C0, C1, …, Cm}, where the length of challenge inputs 'S' and 'C' is given as 'n' and 'm' respectively. The parameter 'n' decides the size of AND-OR network and the parameter 'm' decides count of AND-OR networks in an ARL block. Total 2^m AND-OR networks are employed in an ARL block. Further, 'm' also decides the size of multiplexer exploited in the ARL block. The value of a key-bit produced from a selected AND-OR network is decided by the bit-pattern of challenge input set ‘S’. The challenge input set ‘C’ determines which AND-OR network will be selected. An AND-OR network architecture, which is flexible, is shown in Fig. 8.9.

The factors such as the count of ARL blocks, the size of challenge inputs ‘S’ and ‘C’ determine the structure of ARL blocks and the configuration of AND-OR networks, which in turn determines the architecture of an ARL unit. Since, these various factors of ARL architecture are tuned by the designer, therefore only s/he possesses the details of the custom ARL unit. Hence an adversary cannot know the exact architecture of ARL unit. This makes infeasible for an adversary to know the ILB keys, therefore the reconfigured ILBs using the keys generated from the ARL unit go undetected. Thereby, the removal attack on the ILBs is thwarted.

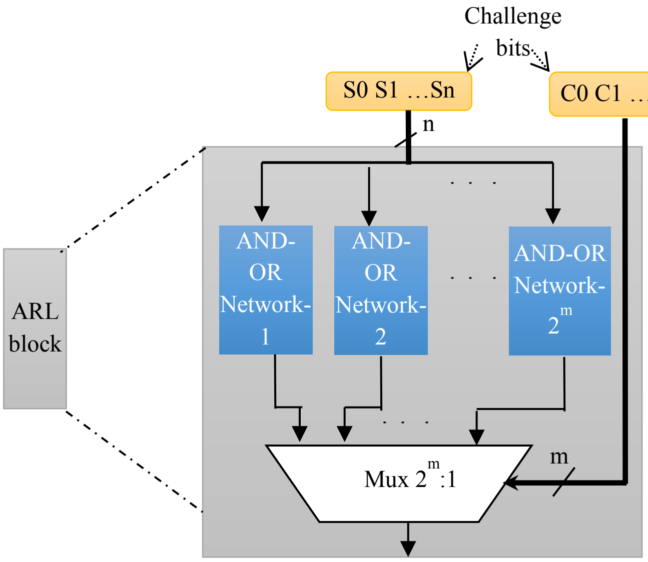


Fig. 8.8 Overview of ARL block

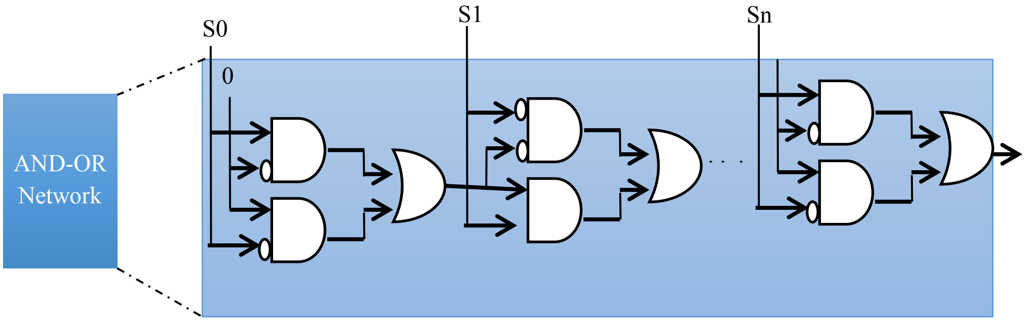


Fig. 8.9 Overview of an AND-OR network used in ARL block

8.4.3. Advantage of using lightweight ARL based ILBs-key generation hardware over AES and SHA-512 based logic to secure against removal attack

1. The ARL unit offers higher security in terms of number of challenge bits needed to decode the keys of reconfigured ILBs (from an attacker's standpoint). Generating the one bit of an ILB key requires total n + m bits of the challenge inputs, as discussed earlier. In order to reconfigure 64 ILBs, total 64*8=512 key-bits need to be generated. The 512 key-bits are generated from 512*(n+m) bits of challenge input of ARL unit. More explicitly, for n=3 and m=3, determining 512 key-bits require total 3072 bits of challenge input. On the other hand, four instances of AES128 will be required to generate 512 key-bits to reconfigure 64 ILBs. To decipher 512 bits, total 1024 bits (4×128 bits of fixed secret key + 4×128 bits of input) of challenge inputs will be required. And, in case of SHA-512 based logic, input of 1024-bit is required to know by the attacker to decipher the 512-bit output (acting as key-bits for ILBs). Thereby, an ARL unit offers higher security due to more number of challenge-bits required for cracking the same number of ILB keys and determining the ILB structures.
2. Producing key-bits for 'Z' more ILBs can be performed simply by adding Z*8 more ARL blocks in the ARL unit. However, producing key-bits for more ILBs using AES-128 or SHA-512 based logic demands one extra instance to be used with the functionally obfuscated design. This impacts area and power requirement. Hence, the ARL unit offers a lightweight solution to secure a functionally obfuscated design against removal attack.

8.5. Summary

The threat of removal attack is imperative to address to secure the functionally obfuscated IP cores or to preserve the goal of functional obfuscation. This chapter discussed SHA-512 and lightweight ARL unit based ILBs-key generation hardware to handle the removal attack on functionally obfuscated DSP cores. This chapter discussed the internal architecture of custom SHA-512 based key generation hardware and ARL unit. Further, we also discussed the advantages of using SHA-512 and lightweight ARL unit based ILBs-key generation hardware compared to an AES based solution.

Chapter 9

Robust Logic Locking Technique for Preventing IP Piracy

The IP piracy can be prevented by locking the functionality of the design through some key gates which are activated using correct keys only. This kind of technique of obscuring the design functionality is referred to as logic locking [36]. This chapter discusses a proposed robust logic locking technique for securing IP cores for DSP applications, by placing DSP locking cells (DLCs) at selected locations in the design. The proposed novel structure of DLC uses sequential elements such as flip-flops (FFs) alongwith combinational elements such as logic gates. The robustness of the proposed DSP locking cell structure lies in the fact that the probability of recovering the correct key even in exhaustive trials is rendered considerably lesser than one.

The outline of the chapter is as follows. The first section formulates the problem. The second section discusses the proposed logic locking technique for securing DSP cores against IP piracy under the following sub-sections: overview, proposed DSP locking cells (DLCs) structure, security assessment of DLCs, insertion of DLCs in the design, metrics used to evaluate security of the logic locking technique. The third section discusses the different attack scenarios addressed by the proposed technique. Finally, the fourth section summarizes the chapter.

9.1. Problem Formulation

Given the data flow graph (DFG) representation of a target application, module library, resource constraints, along with the objective of securing IP cores against IP piracy, generate a locked (functionally obfuscated) netlist of the IP core.

9.2. Logic Locking Technique for Securing DSP Cores against IP Piracy

This section discusses the proposed logic locking technique for securing DSP cores under the following subsections:

9.2.1. Overview

The overview of logic locking technique for securing DSP IP cores is depicted in Fig. 9.1. The proposed technique takes following inputs: DFG representing the target DSP application, module library, resource constraints, designer specified tuning variable 'ω' and keys for the DLCs. The logic locking technique integrated with the design flow generates a locked netlist at the output. The following three steps are executed to accomplish the logic locking process as shown in Fig. 9.1: (i) design a RTL datapath of target DSP application using high level synthesis (HLS) process (ii) perform locking of RTL datapath through insertion of reconfigured DLCs based on encoded value of 'ω' which determines the place for DLCs insertion as per its encoding rules (iii) perform synthesis of locked gate-level netlist. The keys for DLCs to produce reconfigured DLC structures are obtained from AES-128 output.

To perform the locking of the functionality of the DSP core, we proposed a

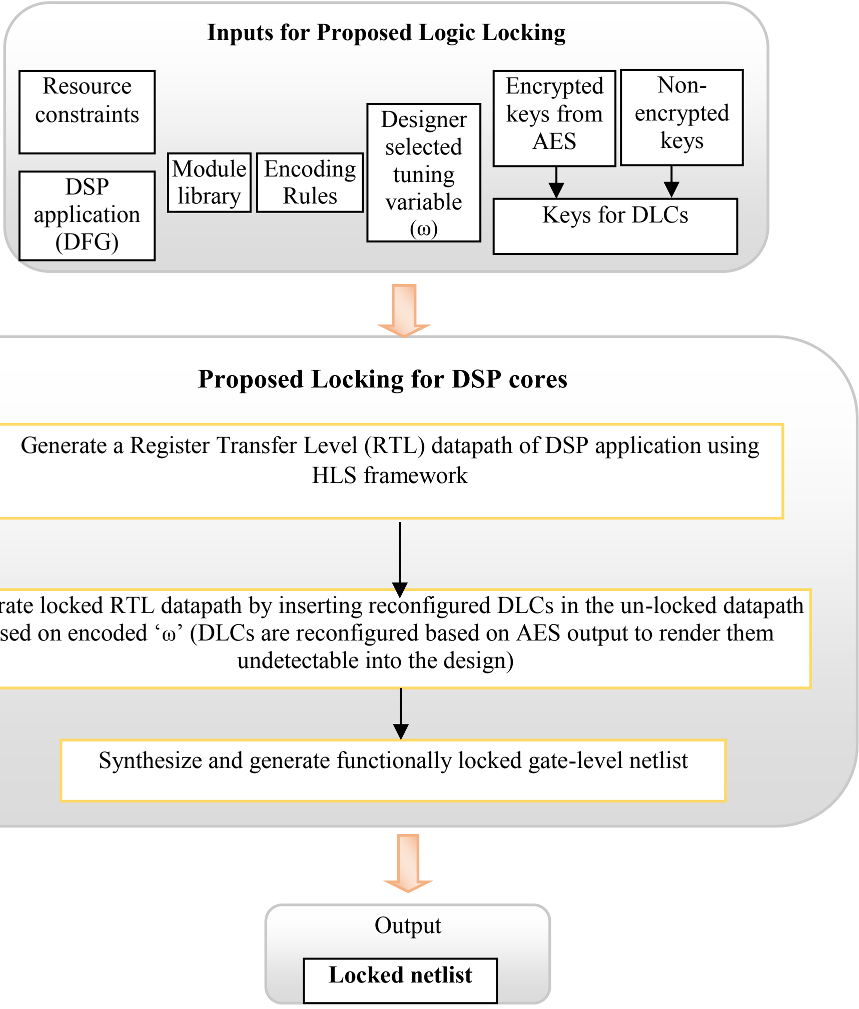

Inputs for Proposed Logic Locking
Resource constraints
DSP application (DFG)
Module library
Encoding Rules
Designer selected tuning variable (ω)
Encrypted keys from AES
Non-encrypted keys
Keys for DLCs
Proposed Locking for DSP cores
Generate a Register Transfer Level (RTL) datapath of DSP application using HLS framework
Generate locked RTL datapath by inserting reconfigured DLCs in the un-locked datapath based on encoded 'ω' (DLCs are reconfigured based on AES output to render them undetectable into the design)
Synthesize and generate functionally locked gate-level netlist
Output
Locked netlist

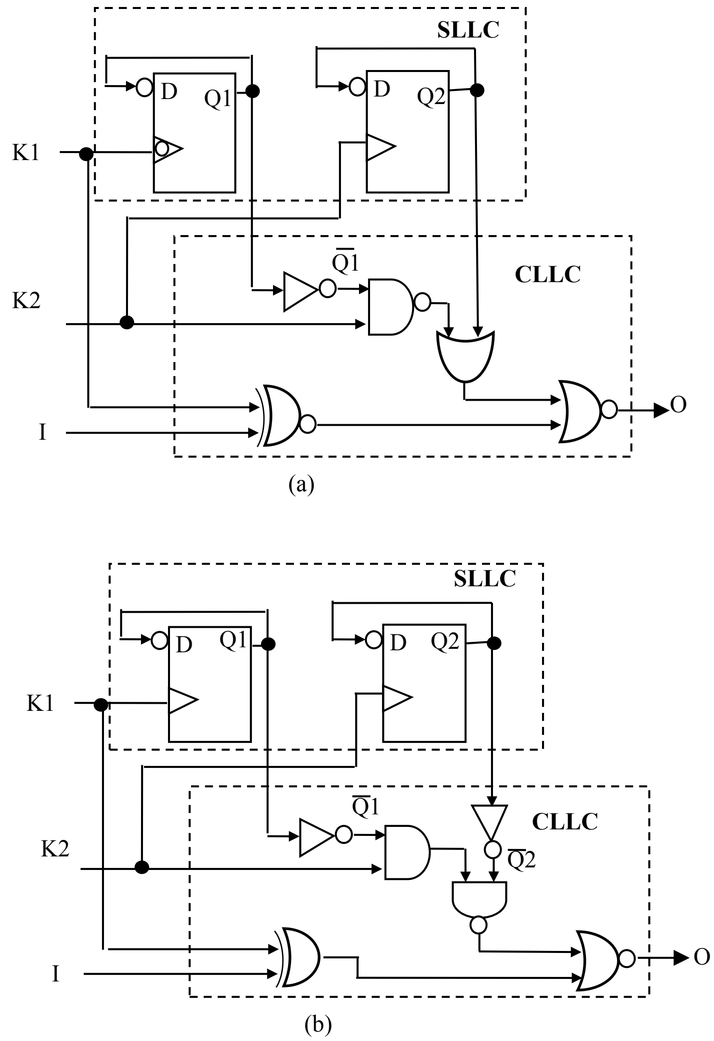
Fig. 9.1. Overview of proposed logic locking methodology for DSP IP cores

robust and reconfigurable structure of DLC. This reconfiguration of DLCs is accomplished using AES128 output. From the security perspective, the proposed logic locking approach achieves the probability of deducing correct key in all possible trials (attempting exhaustive key combinations) extremely lesser compared to the probability of deducing the same in exhaustive trials using existing logic locking techniques [36], [37]. In other words, in the proposed technique, the correct key is only possible to be obtained by an attacker if he/she attempts it at first combination. If the attacker fails to apply correct key combination at first trial, during providing various key combinations, then he/she cannot deduce the correct key in the leftover trials. On the other hand, in the existing logic locking approaches, the probability of deducing the correct key using exhaustive trials is 1. Thus, the proposed structure of DLCs ensures higher security than the existing logic locking technique [36] for DSP cores.

9.2.2. Proposed DSP locking cells (DLCs) structure and its security properties

The proposed DSP locking cell structure is constituted using following two logic locking sub-cells (i) sequential logic locking cell (SLLC) (ii) combinational logic locking cell (CLLC) as shown in Fig. 9.2. A SLLC comprises of two D flip-flops. We refer these flip-flops as key-FFs in this chapter, as their functioning is driven by the key bits. Each key-FF acts in toggling mode. The initial state of each key-FF is required to set to ‘0’. This setting of key-FFs can be achieved using power on reset. Further, a CLLC comprises of various logic gates such as OR, NOR, EX-OR, EX-NOR, AND, NAND and NOT gate. Different arrangements of various logic gates in CLLC and variations of edge triggering (such as positive and negative) of the key-FFs in SLLC can produce various reconfigured functional structures of DLCs.

The proposed DLCs exhibit various security properties such as (i) pair-wise security, protection against key-gate isolation, protection against run of key-gates, protection against muting of key gates to hinder the sensitization of key bits at output (ii) involving the outputs of key-FFs with key-gates makes the hybrid structure of proposed DLC more robust (iii) deducing the valid key using all possible combinations of key is infeasible, given that the valid key



(a)

(b)

Fig. 9.2. Proposed sample reconfigured DLC structures (where, "K1K2" is a two-bit key)

combination is not provided at the first trial (iv) since both type of cells viz. D-FFs and logic gates participate in the logic locking, therefore detection of only key-gates or only key-FFs is not enough for an attacker to perform the attacks. All the key-gates and key-FFs of a DLC are required to be identified by the attacker. This renders the attacker's effort tremendously challenging due to camouflaging of key-FFs and key-gates in the many similar resources of the gate level netlist of the DSP core (v) the proposed DLC structure requires lesser number of key-bits (only two per DLC structure), therefore keys for more DLCs can be obtained from an AES instance.

9.2.3. Security assessment of DLCs

Using the proposed DLCs, obtaining the correct key is only possible if the attacker attempts it at first trial. The reason of not finding the correct key in

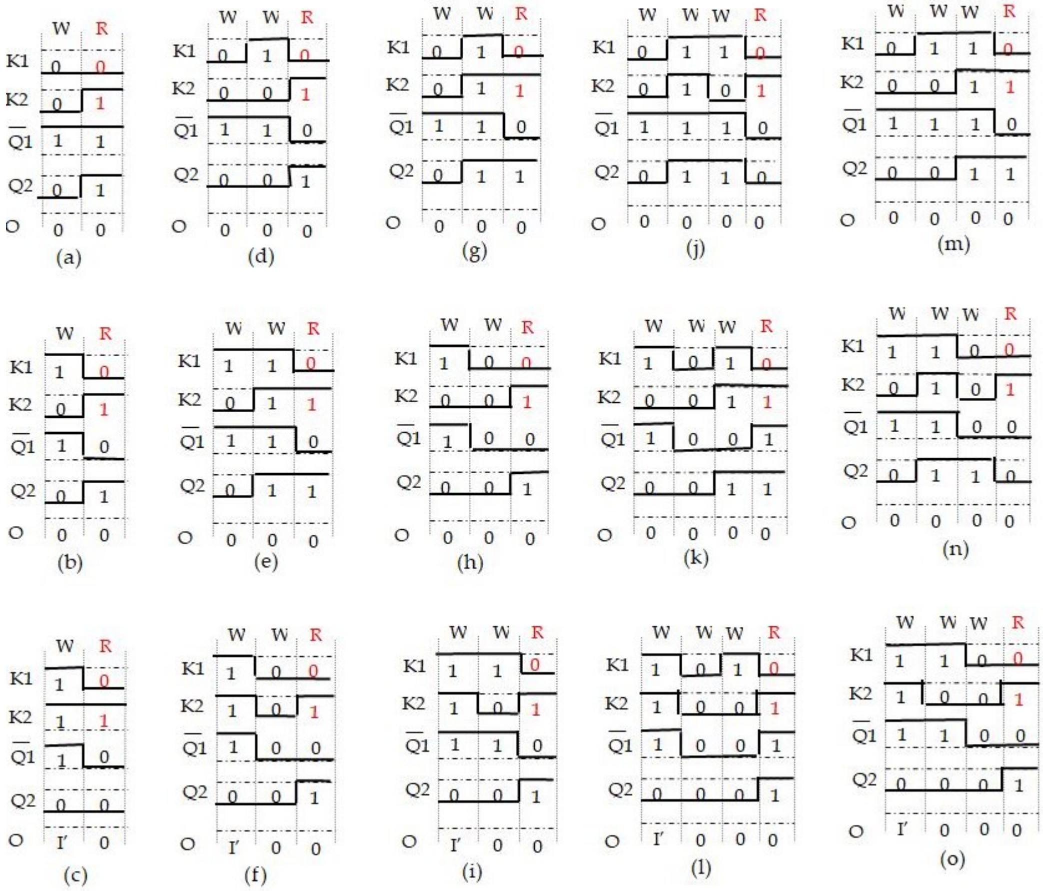


Fig. 9.3. Waveforms showing that correct output is not obtained unless the correct key is applied only in the first trial (n*ote- R: right key combination, W: wrong key combination. At W, output (O) is either complement of input (**I'**) or 0 and at R, O is always 0)*

exhaustive combinations except applying it at the first trial is explained in this sub-section with the help of proposed DLC and timing diagram shown in Fig. 9.2(a) and 9.3 respectively:

The proposed DLC structure leverages the toggling nature of FF elements. A FF can work in toggling mode on applying a certain input alongwith a clock signal. In the proposed DLCs, organization of logic gates and key-FFs is such that the unique feature of toggling of FFs can be exploited for robust security of logic locking against the brute force attack. To generate a correct output through proposed DLCs, the output of each key-FF must stay at 0 while applying the correct key. As shown in Fig. 9.2, the key bits are connected with the clock input of the key-FFs. Therefore during exhaustive combinations/trials of key value, variation in key-bits would lead to flipping of the flip-flop output. This guarantees that the output of a FF does not stay at 0 upon applying the correct key after a wrong key combination applied in the first trial. Hence, the attacker cannot obtain the correct output on exhaustive key combinations unless the valid key is provided only in the first trial. This idea is further explained below with the help of the proposed DLC structure, given Fig. 9.2(a), for different scenarios:

Note: Since each DLC requires two-bit key to activate therefore exhaustive trials in deducing the correct key of a DLC are four.

1. *Scenario of applying valid key in the first trial*: If the initial setting of each key-FF output is 0 and the correct key "01" is provided to the DLC (Fig. 9.2(a)) in the first trial, then the DLC functions correctly. In this case, the path of DLC from input (I) to output (O) bit is unlocked and valid output is generated. Thus, the correct output can be produced using DLC on providing valid key in the first trial. This could be achieved because the flip-flops output "Q1Q2" are "00" during the application of correct key. Therefore, the probability of obtaining valid output on providing valid key in the first trial is 1.
2. *Scenario of applying valid key in the second trial*: Let's suppose an invalid key is applied in the first trial and the valid key "01" is applied in the second trial. This will make the transition in either of the key bits (K1 or K2) as depicted in the waveforms shown in Fig. 9.3(a), (b) and (c). This transition on a key-bit (which is the clock input of flip-flop) will result in flipping (toggling) on output of respective key-FF and thus "Q1Q2" will no longer remain at "00". For example, if invalid key "00" is provided in the first trial then providing valid key "01" in the second trial results in transition on key-bit 'K2' (as shown in Fig. 9.3(a)) which toggles the output Q2. This toggling in the output of key-FF results in incorrect output of the DLC on feeding correct key in the second trial. Therefore, the probability of obtaining valid output on providing the valid key in the second trial is 0.
3. *Scenario of applying valid key in the third trial*: If valid key is fed in the third trial post two wrong trials, then the valid output is not obtained from the DLC (Fig. 9.2(a)) as depicted in the waveforms shown in Fig. 9.3(d), (e), (f), (g), (h) and (i). The underlying reason is that the applying valid key after two wrong trials results in the transition on key-bits. This

transition toggles either Q1 or Q2 or both, hence “Q1Q2” is not maintained at “00”. Therefore, the probability of obtaining valid output on providing valid key in the third trial is 0.

4. *Scenario of applying valid key in the fourth trial*: Similarly, applying valid key after three wrong trials makes the output of the DLC invalid as depicted in waveforms shown in Fig. 9.3 (j), (k), (l), (m), (n) and (o). Therefore, the probability of obtaining valid output on providing valid key in the fourth trial is also 0.

Although it does not match with our intuition that the probability of obtaining valid output by feeding valid key is not always 1, however it has been achieved for the proposed DLC structure as shown in Fig. 9.3. Therefore, the valid key cannot be deduced through the proposed DLCs in exhaustive trials except first and this makes the proposed logic locking more robust.

9.2.4. Insertion of DLCs in the design

Figure 9.4 shows the insertion technique of proposed DLCs in the DSP core design. Before the DLCs insertion process, the structure of DLCs and locations for their insertion are determined. To determine the locations of the DLC insertion, the designer specified tuning variable ‘ω’ is used which varies from 0 to 3 and conform to the following encoding rules:

- ω =0: select even output bits of functional units (FUs) for DLCs insertion
- ω =1; select odd output bits of FUs for DLCs insertion
- ω =2; select prime output bits of FUs for DLCs insertion
- ω =3; select all output bits of FUs for DLCs insertion

The total number of DLCs (T_{DLC}) to be placed in a DSP design is determined using the following equation:

$$T_{DLC} = (\text{total \# of FUs}) \times (\text{\# of output bits per FU chosen based on } \omega) \qquad (9.1)$$

For example, if a design contains three FUs each of size 32-bit then total output bits of FUs are 3×32=96. Therefore, on choosing ω =3, total 96 DLCs are placed in the design. Post determining the number of DLCS and their location of insertion based on ω, the DLCs are inserted at selected locations in the RTL datapath of the DSP design. It is noteworthy that the proposed DLC structure is not predefined rather it is reconfigurable based on its key value. As

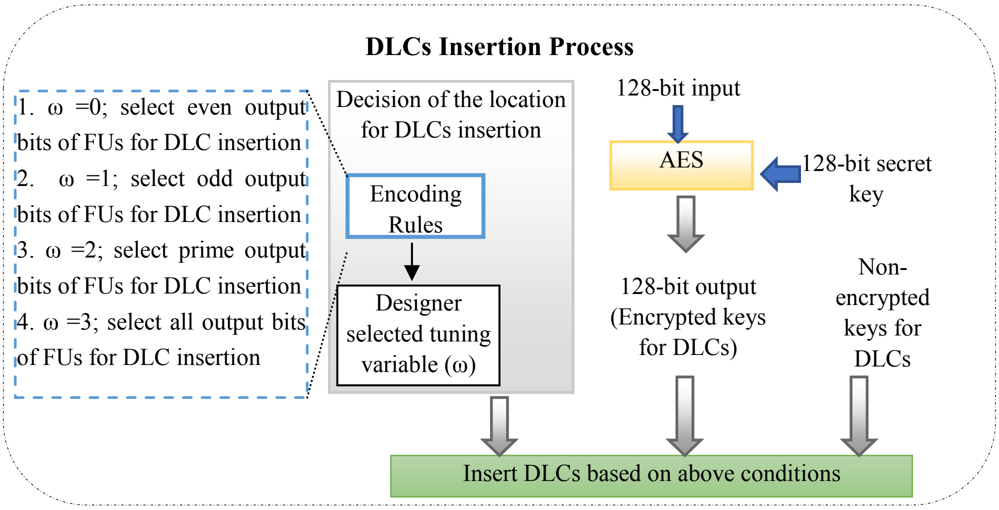

DLCs Insertion Process
1. ω =0; select even output bits of FUs for DLC insertion
2. ω =1; select odd output bits of FUs for DLC insertion
3. ω =2; select prime output bits of FUs for DLC insertion
4. ω =3; select all output bits of FUs for DLC insertion
Decision of the location for DLCs insertion
Encoding Rules
Designer selected tuning variable (ω)
128-bit input
AES
128-bit secret key
128-bit output (Encrypted keys for DLCs)
Non-encrypted keys for DLCs
Insert DLCs based on above conditions

Fig. 9.4. The process of DLCs insertion into RTL datapath of DSP cores

shown in Fig. 9.4, a subset of DLCs obtains the keys from AES128 output in order to generate the reconfigured structure of respective DLCs. Since a DLC is activated using a two-bit key, therefore 128-bit AES output can be used to reconfigure upto 64 DLCs. This structural reconfiguration of DLCs disables an attacker in identifying them in the design, hence hindering the removal attack.

9.2.5. Metrics used to evaluate security of the logic locking technique

Following metrics are used to evaluate the security achieved using the proposed logic locking technique:

1. Probability of obtaining valid key in exhaustive trials (P_v)

The P_v is the metric for assessing security of the proposed logic locking in compared to the existing logic locking techniques [36], [37]. The P_v metric indicates the probability of deducing the correct key by trying all the possible key combinations. The equation of P_v is formulated as follows:

$$P_v = \left(P_{appl}^{vk_{(1)}}\right).\left(P_{obt}^{vo_{(1)}}\right) + \left(P_{appl}^{vk_{(2)}}\right).\left(P_{obt}^{vo_{(2)}}\right) + \cdots + \left(P_{appl}^{vk_{(2^{k_b})}}\right).\left(P_{obt}^{vo_{(2^{k_b})}}\right) \qquad (9.2)$$

Where, k_b and 2^{k_b} represent the total number of key-bits and the exhaustive key-combinations or trials respectively.

$\left(P_{appl}^{vk_{(1)}}\right)$=probability of applying valid key at 1st trial.

$\left(P_{appl}^{vk_{(2)}}\right)$= probability of applying valid key at 2nd trial.

$\left(P_{appl}^{vk_{(2^{k_b})}}\right)$=probability of applying valid key at $(2^{k_b})^{\text{th}}$ trial.

$\left(P_{obt}^{vo_{(1)}}\right)$= probability of obtaining valid output on applying valid key at 1st trial.

$(P_{obt}^{vo_{(2)}})$= probability of obtaining valid output on applying valid key at 2ndtrial.

$\left(P_{obt}^{vo_{(2^{k_b})}}\right)$=probability of obtaining valid output on applying valid key at $(2^{k_b})^{\text{th}}$ trial.

It seems to be likely that the valid output can be obtained by applying valid key at any trial (with probability=1). However, this fact is challenged using the proposed DLC based logic locking technique where an attacker cannot find valid output even on applying valid key unless it is fed only in the first trial. More explicitly, this fact is explained for the following two cases:

(i) **In case of existing techniques of logic locking [36], [37]**: Valid output can be obtained by applying valid key at any trial with probability 1. In this case, the trial number in which the correct key is applied does not matter. The probability of applying valid key at any trial can be formulated as $1/2^{K_b}$ as the total possible trials are 2^{K_b} and the favourable trial is only one in which valid key is fed. Thus, the probability of obtaining valid key in exhaustive trials (i.e. P_v) is derived as follows using (9.2):

$$P_v = \frac{1}{2^{K_b}}.1 + \frac{1}{2^{K_b}}.1 + \frac{1}{2^{K_b}}.1 + \cdots 2^{K_b} times$$

$$P_v = 2^{K_b}\frac{1}{2^{K_b}}$$

$$P_v = 1 \tag{9.3}$$

The above eq. (9.3) shows that the probability of obtaining valid key in exhaustive trials is 1 in case of existing logic locking techniques.

(ii) **In case of proposed DLC based logic locking**: The probability that the valid output can be obtained on feeding valid key at any trial is not always 1. Only the probability of obtaining valid output on feeding the valid key in the first trial is 1. In case of other trials, this probability is 0 due to the structural nature of the proposed DLC. Thus, the probability of obtaining valid key in exhaustive trials is derived as follows using (9.2):

$$P_v = \frac{1}{2^{K_b}}.1 + \frac{1}{2^{K_b}}.0 + \cdots + \frac{1}{2^{K_b}}.0$$

$$P_v = 1/2^{K_b} \tag{9.4}$$

In other words, if the valid key is only fed at the first trial, only then DLC is activated and the attacker can deduce the correct key. Thereby, security (strength of logic locking) of the logic locking technique in terms of the probability of obtaining valid key in exhaustive trials is achieved to be $1/2^{K_b}$ instead of 1.

2. Encryption strength

Encryption strength (En^{s}) metric measure the percent of DLC key bits encrypted using an AES128 output. This metric is given as follows:

$$\mathrm{En}^{\mathrm{s}} = \frac{\mathrm{N}_{\mathrm{op}}^{\mathrm{AES}}}{\mathrm{N}_{\mathrm{kbits}}^{\mathrm{total}}} \tag{9.5}$$

Where, $\mathrm{N}_{\mathrm{op}}^{\mathrm{AES}}$ indicates the number of encypted output bits generated from AES128 and $\mathrm{N}_{\mathrm{kbits}}^{\mathrm{total}}$ indicates the total number of DLC key bits in a design.

9.3. Addressing different Attacks Scenarios

To launch attacks, an attacker is assumed to have access of the following: (i) locked gate-level netlist or a layout/GDS-II file (ii) sophisticated tools for reverse engineering to obtain locked gate-level netlist (iii) functional IC of the locked design. Using the abovementioned facilities, the attacker can try to unlock the design through various attacks such as: key-sensitization based attacks [37], SAT attack [40], removal attack [39]. Post obtaining the unlocked netlist, the design becomes susceptible to IP piracy attack. Various attacks are addressed using the proposed logic locking technique as follows:

1. **Addressing key-sensitization based attacks**: A key-bit can be sensitized at primary output by feeding a suitable input pattern to primary input. In the proposed DLC, one key bit obstructs the path of sensitization of the

other key bit and the complex organization of key-gates and key-FFs renders the logic locking resilient against the key-sensitization attack based on the following: (i) isolated key-bits (ii) run of key-gates (iii) mutable key-gates.

2. **Addressing brute force attack:** Performing brute-force attack almost infeasible because of the proposed DLCs based logic locking. This is due to the fact that the probability of obtaining valid key even in exhaustive trials is $1/2^{K_b}$ (highly lesser than one).
3. **Addressing SAT attack**: The SAT attack works by rejecting wrong key combinations iteratively with the help of distinguishing input-output (DIO) pairs. Since the proposed logic locking is employed for DSP IP cores which contain several multiplication operations, hence the SAT attack is not feasible due to its non-scalability for multipliers. However, efficient SAT attack algorithms may be assumed to be developing for DSP cores in future. The proposed logic locking can be used as a proactive countermeasure against the potential SAT attack, which can work as follows. The keys for the proposed DLCs are extracted from the encrypted output of the custom- AES block which executes a one-way random function. This prevents an attacker from determining the AES inputs from its output.
4. **Addressing removal attack**: If the attacker is assumed to be aware of some templates of DLC structures and has access to the locked netlist, then s/he can attempt removal attack on DLCs. However, locked design netlist using DLCs cannot be subjected to removal attack due to the following reason. The inside structures of DLCs are reconfigured using the different combinational logic gates and positive/negative edge triggered D-FFs, according to the output of AES128 hardware. Thereby multiple DLC structures are possible, depending on AES output, which are all unknown to the attacker. Hence, the attacker cannot detect DLCs fully in the locked design because of his/her unawareness of the DLCs structure, encrypted output of AES and its corresponding secret key. This prevents an attacker from matching the reconfigured DLC structures with the available templates and thus thwarts his/her attempts to remove DLCs.

5. **Addressing IP piracy attack**: The security against IP piracy using logic locking cannot be realized unless the logic locking is strong enough. The strength of logic locking is challenged by some key based attacks and removal attack etc. However, security against these attacks can be ensured as discussed earlier in this section. Further, the proposed logic locking prevents the IP piracy as the attacker cannot misuse the IP without unlocking it using the correct key. Since the probability of finding valid key in the exhaustive trials is very less due to the proposed logic locking, hence impeding an attacker from deducing the valid key value.

9.4. Summary

The logic locking technique has been proposed as a preventive measure against IP piracy. This chapter discussed the proposed DLC based logic locking technique which is resilient against various attacks such as key sensitization based attack, removal attack, SAT attack and brute force attack. Further, the robust structure of proposed DLC makes it nearly infeasible to obtain valid key even in the exhaustive trials. Various security properties discussed in the chapter makes the proposed approach highly robust to prevent the IP piracy attack.

Chapter 10

Experimental Results and Analysis

The experimental results and analyses of the proposed techniques of hardware/IP core security for DSP and multimedia applications are presented in this chapter. The results have been calculated for various DSP and multimedia benchmarks [8], [58], [59].

10.1 Results and analysis: Hardware steganography techniques for securing IP cores against piracy

The experimental results of the proposed hardware steganography techniques discussed in chapter 3 are presented in this section. 15 nm open cell library [60] was used to calculate different design parameters such as area and delay. The following subsections present the results for entropy based steganography and key-driven hash chaining based steganography techniques respectively.

10.1.1 Evaluating entropy based steganography

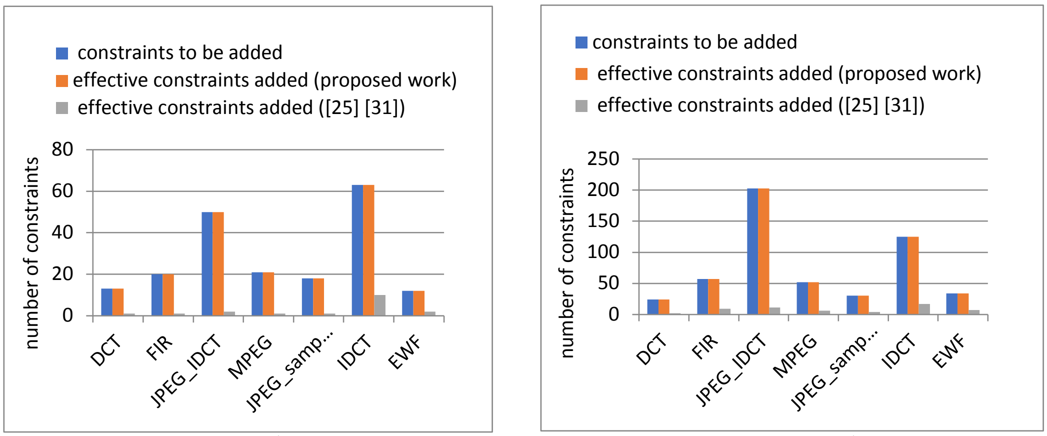
The proposed entropy based steganography technique has been implementation in java and run on processor with specifications "4 GB, DDR3 memory at 1.9 GHz". The implementation run time ranges between 2.41 to 7.01 seconds. Unlike signature-based IP core security techniques, the proposed technique is entirely signature free. The amount of vendor's secret information embedded can be totally controlled by the vendor/designer using an entropy threshold value. The security of proposed steganography technique and its impact on design cost are assessed as follows:

10.1.1.1 Security analysis

For analyzing the security achieved in terms of strength of IP ownership proof, the probability of coincidence (Pc) metric presented in eq. 3.2 in the section 3.2.5 of chapter 3 is used. Table 10.1 shows the Pc value for the increasing value of entropy threshold T^E. As shown, increasing T^E leads to increase in the number of stego-constraints 'f' embedded into the design hence resulting into decreasing value of Pc. The lower Pc value is desirable which represents

Table 10.1. Impact on Pc using proposed IP steganography for different entropy threshold (varying number of stego-constraints 'f')

Benchmark	# of storage variables	# of registers before steganography	Value of threshold entropy					
			$T^E = 4$		$T^E = 6$		$T^E = 8$	
			f	P_c	f	P_c	f	P_c
DCT	22	8	13	1.8E-01	24	4.1E-02	43	3.2E-03
FIR	30	8	20	6.9E-02	57	4.9E-04	57	4.9E-04
JPEG_IDCT	135	29	50	1.7E-01	203	8.1E-04	317	1.5E-05
MPEG	41	14	21	2.1E-01	52	2.1E-02	59	1.3E-02
JPEG_sample	44	12	18	2.1E-01	30	7.4E-02	72	1.9E-03
IDCT	49	10	63	1.3E-03	125	1.9E-06	125	1.9E-06
EWF	35	7	12	1.6E-01	34	5.3E-03	57	1.5E-04



(a) E^{th}=4

(b) E^{th}=6

Fig. 10.1. Estimation of the number of effective constraints added for both solutions (considering same number of constraints) at different entropy threshold

stronger proof of authorship. Hence, a large entropy threshold value can be chosen to achieve a stronger proof of authorship.

The comparison of proposed steganography approach with the related works [25], [31] in terms of the effective stego-constraints embedded in the design is shown in Fig. 10.1, for designer specified different entropy threshold values. As shown in Fig. 10.1, the proposed approach embeds more number of constraints effectively than related approaches [25], [31]. This is because, [25], [31] possess several constraints by default which is out of designer's control. So, effectively lesser constraints are inserted. However the proposed technique does not contain the constraints by default, thus offering full control over the number constraints embedded than [25], [31].

10.1.1.2 Design cost analysis

The metric used to compute the design cost of the proposed steganography in terms of area and delay is discussed in eq. (3.8) of section 3.4 of chapter 3.

Table 10.2. Design cost of the proposed approach in terms of area and latency

Benchmark	Proposed Solution	L_T(ps)	A_T(μm)	Cost
DCT	1A 4M	928	327.15	0.47
FIR	4A 4M	994	383.78	0.44
JPEG_IDCT	12A 12M	1988	1155.27	0.33
MPEG	3A 7M	795	596.11	0.36
JPEG_sample	6A 1M	1325	198.18	0.58
IDCT	4A 3M	1723	309.85	0.48
EWF	2A 1M	2716	118.75	0.66

Table 10.3. Impact on storage overhead on increasing threshold entropy

Benchmark	# of storage variables	# of registers before steganography	Value of threshold entropy resulting into 'q' more registers				
			q=0	q=1	q=2	q=3	q=4
DCT	22	8	2-6	7	NA	NA	NA
FIR	30	8	2-5	NA	NA	NA	NA
JPEG_IDCT	135	29	2-9	NA	NA	NA	NA
MPEG	41	14	2-7	NA	NA	NA	NA
JPEG_sample	44	12	2-8	9	NA	NA	NA
IDCT	49	10	2-5	NA	NA	NA	NA
EWF	35	7	2-7	8	NA	9,10	11

Table 10.4. Comparison of proposed approach with [25], [31] in terms of the # of registers for the same # of constraints (*Note: C - # of constraints added; P – proposed solution; R – related works; % RR – register reduction % obtained w.r.t. [25], [31])*

Benchmark	# of Registers required after embedding same number of constraints											
	$T^E = 4$				$T^E = 5$				$T^E = 6$			
	#C	P	R	**% RR**	# C	P	R	**% RR**	# C	P	R	**% RR**
DCT	13	8	9	**11 %**	18	8	10	**20%**	24	8	10	**20%**
FIR	20	8	9	**11 %**	57	8	10	**20%**	57	8	10	**20%**
JPEG_IDCT	50	29	29	**--**	124	29	30	**3.3%**	203	29	30	**3.3%**
MPEG	21	14	15	**6.6 %**	46	14	15	**6.6%**	52	14	15	**6.6%**
JPEG_sample	18	12	13	**7.6 %**	20	12	13	**7.6%**	30	12	13	**7.6%**
IDCT	63	10	11	**9.1 %**	125	10	18	**44.4%**	125	10	18	**44.4%**
EWF	12	7	8	**12.5%**	30	7	8	**12.5%**	34	7	8	**12.5%**

Table 10.2 presents the design cost of the proposed approach. Further, Table 10.3 presents the impact on register overhead with varying entropy threshold value. The variable 'q' denotes the number of extra registers needed to implant the stego-information. As presented in Table 10.3, the register overhead either increases marginally or remains same with varying entropy threshold value. This indicates the low overhead attribute of the proposed technique.

Further, the proposed approach is compared with [25], [31] in terms of registers count as shown in Table 10.4. As shown, the number of registers needed to embed the same number of constraint edges in case of proposed approach and [25], [31] is compared. Where # of C denotes the number of constraints edges to be inserted in both cases, P and R denote the number of registers needed in the proposed approach and [25], [31] respectively. The results in the table show that the proposed steganography incurs lesser storage overhead than [25], [31]. The reason is that the storage overhead in the signature-based protection techniques [25], [31] depends on the signature combinations and its encoded meaning. Additionally, foreseeing the relationship of signature combination with the storage overhead is also not feasible. On the other hand, the proposed steganography technique is signature free and hence independent of signature combination.

10.1.2 Evaluating key-driven hash chaining based steganography

The proposed key-driven hash chaining based steganography was implemented in C++ and run on a processor with specifications "4 GB, DDR3 memory". The following subsection presents the analysis on security and design cost of the proposed technique:

10.1.2.1 Security analysis

For analyzing the security achieved in terms of strength of IP ownership proof, the probability of coincidence (Pc) metric presented in eq. 3.3 in the section 3.3.4 of chapter 3 is used. Table 10.5 shows the Pc value for varying size of stego-constraints embedded in two different phases (register allocation and FU vendor allocation) of HLS design process. As shown, we achieved lesser Pc as the stego-constraints size is increased. Further, implanting stego-information into two distinct phases shows that the higher amount of digital evidence are hidden into the design. This strengthens the stego-mark.

Additionally, we analyze the security offered by the proposed technique in terms of key strength and an attacker's effort in determining the stego-constraints embedded into the IP core. The key strength in terms of maximum

Table 10.5. Variation in Pc for increasing size of stego-constraints using proposed key-driven hash-chaining approach (*note: f1 indicates #0s and f2 indicates #1s embedded effectively*)

Benchmarks	# of registers (G)	#constraint		Pc	#constraint		Pc	#constraint		Pc
		f1	f2		f1	f2		f1	f2	
DCT	8	13	10	9.9E-3	24	12	1.3E-3	43	12	1.0E-4
FIR	8	20	6	4.7E-2	57	23	1.1E-4	57	23	1.1E-4
JPEG_IDCT	29	203	109	3.8E-4	317	109	6.9E-6	355	109	1.8E-6
MPEG	14	21	16	9.6E-2	52	23	6.9E-3	59	23	4.1E-3
JPEG_sample	12	30	21	1.6E-3	72	31	6.7E-6	116	31	1.4E-7
EWF	7	34	18	2.0E-8	57	28	5.7E-13	86	28	6.5E-15

stego-key size (in bits) and attacker's effort in determining the key are given using eq. (3.4) and eq. (3.5) respectively discussed in section 3.3.4 of chapter 3. This represents the stronger security of proposed technique against determining the valid stego-key value by an attacker. The maximum key size and attacker's effort in terms of key security are reported in Table 10.6. Moreover, the attacker is also required to deduce the encoded bits used in each hash block for finding the stego-constraints embedded into the design. The attacker's effort of finding encoded bits, calculated using eq. (3.6) discussed in section 3.3.4, is shown in Table 10.6. Further, the attacker's total effort, calculated using eq. (3.7) discussed in section 3.3.4, is also reported in Table 10.6.

The security achieved in terms of maximum stego-key size is compared with the entropy based steganography technique. In contrast to the entropy based steganography technique, the proposed approach offers very large stego-key size (in bits). This is because the entropy-based steganography technique does not require stego-key; hence the key size remains zero.

10.1.2.2 Design cost analysis

The metric used to compute the design cost of the proposed hash-chaining based steganography has been discussed in eq. (3.8) of section 3.4 of chapter 3. Table 10.7 presents the design cost of the proposed approach. As shown, the design cost either remains same or increases by a nominal value with the growing size of constraints (#0s and #1s). The reason is that the design may require extra register in some cases to satisfy larger number of constraints.

Further, Table 10.8 analyzes the design cost overhead by comparing the baseline cost with the post embedding steganography cost. The table shows

Table 10.6. Security of proposed approach in terms of maximum key size and attacker's total effort

Benchmark	Maximum key size using (3.4)	Maximum attacker effort in terms of key security (using (3.5))	Maximum attacker effort in terms of finding encoded bits (using (3.6))	Total attacker effort (using (3.7))
DCT	491520	$>10^{147603}$	10^{2059}	$>10^{149662}$
FIR	192937984	$>10^{57939334}$	10^{2059}	$>10^{57941393}$
JPEG_IDCT	5.8153×10^{35}	$>10^{1.74\times10^\wedge35}$	10^{2059}	$>10^{1.74\times10^\wedge35}$
MPEG	7516192768	$>10^{2277634172}$	10^{2059}	$>10^{2277636231}$
JPEG_sample	283467841536	$>10^{85899345920}$	10^{2059}	$>10^{85899347979}$
EWF	584115552256	$>10^{177004712804}$	10^{2059}	$>10^{177004714863}$

Table 10.7. Impact of increasing size of stego-constraints on the design cost of proposed key driven hash chaining based steganography approach

Benchmarks	effective # of constraints embedded	#constraints		Design cost post phase1 & 2	effective # of constraints embedded	#constraints		Design cost post phase1 & 2
		0s	1s			0s	1s	
DCT	23	13	10	0.45357	26	24	12	0.45357
FIR	26	20	6	0.44465	80	57	23	0.44467
JPEG_IDCT	312	203	109	0.3258	426	317	109	0.3258
MPEG	37	21	16	0.37445	75	52	23	0.37479
JPEG_sample	51	30	21	0.47476	103	72	31	0.47783
EWF	52	34	18	0.6632	85	57	28	0.6655

Table 10.8. Comparison of design cost of proposed approach with respect to baseline

Benchmarks	Design Cost (Baseline)	effective # of constraints embedded	Design cost (proposed)	Cost overhead in %
DCT	0.453	23	0.453	0%
FIR	0.445	26	0.447	0%
JPEG_IDCT	0.325	312	0.326	0.3%
MPEG	0.374	37	0.374	0.2%
JPEG_sample	0.473	51	0.475	0.3%
EWF	0.661	52	0.663	0.3%

that the proposed steganography has nominal impact on the design cost due to incurring only a trivial cost overhead (≤ 0.3 %).

10.2 Results and analysis: Hologram based structural obfuscation to thwart reverse engineering based attacks

The experimental results of the proposed hologram based obfuscation technique discussed in chapter 4 are presented in this section. For experiments, we generated following five different hologram based obfuscated DSP designs, where each is an integration of two DSP cores. (i) IIR-FIR hologram based obfuscated design, (ii) 4 point DCT- 4 point IDCT, (iii) 8 point DCT- 8 point IDCT (iv) 8 point DCT - 4 point DCT and (v) 8 point DIT-FFT - 4 point DIT-FFT. Further, for comparative analysis, same DSP applications were also subjected to HLT based obfuscation technique [45]. A 15nm technology scale open cell library [60] was used to calculate the gate count and delay overhead of the designs of proposed technique and [45].

The gate count affected due to proposed obfuscation and [45] is presented in eq. (4.2) of section 4.2.3 of chapter 4. The comparison of affected gate count of proposed technique and [45] w.r.t. the baseline counterpart is presented in Table 10.9 and 10.10 respectively. Further, the comparison of security (in terms of %gate count affected) due to proposed obfuscation and [45] is presented in Table 10.11. Based on these results, following subsections present the analysis of obfuscated design area, delay and security.

10.2.1 Security analysis of proposed hologram based obfuscation

The security is analyzed using the strength of obfuscation metric which has been discussed using eq. (4.1) in section 4.2.3 of chapter 4. Table 10.11 presents the comparison of obfuscation strength (security) in terms of affected gate count (using eq. (4.1)) due to proposed hologram based obfuscation and [45]. Results show that we achieve more security due to larger # of gates affected compared to [45]. Further, as shown in Table 10.11, affected gate count for 4-point and 8-point DIT-FFT is 0% using [45] because of non-applicability of HLT based obfuscation. On the other hand, the proposed obfuscation offers considerable strength of obfuscation for 8 point DIT-FFT - 4 point DIT-FFT wrt baseline.

Table10.9 Comparison of affected gate count between proposed obfuscation and baseline

DSP kernels	Gate Count of baseline (un-obfuscated)	Gate Count of proposed obfuscation	Gate count reduction (through proposed obfuscation)	Affected gate count through proposed obfuscation
IIR+FIR	9248	6544	2704	29.2 %
8-pt DCT+8-pt IDCT	101632	55424	46208	45.5%
4-pt DCT+4-pt IDCT	26240	14400	11840	45.1%
8-pt DCT +4-pt DCT	63936	52352	11584	18.1%
8-pt DIT-FFT + 4-pt DIT-FFT	14016	9856	4160	29.7%

Table 10.10 Comparison of affected gate count between [45] and baseline

DSP kernels	Gate count of baseline (un-obfuscated)	Gate count of obfuscated design [45]	# of gates changed in terms of input connectivity	# of gates affected [45] (based on eq. (4.2)]
IIR	5968	5968	160	160
FIR	3280	3280	64	64
8-pt DCT	50816	50816	1792	1792
8-pt IDCT	50816	50816	1792	1792
4-pt DCT	13120	13120	256	256
4-pt IDCT	13120	13120	256	256
8-pt DIT-FFT	9344	9344	0	0
4-pt DIT-FFT	4672	4672	0	0

Table 10.11 Comparison of security of proposed obfuscation and [45]

DSP kernels	# of gates affected (proposed obfuscation)	# of gates affected ([45])	Affected gate count (proposed obfuscation)	Affected gate count ([45])
IIR+FIR	2704	224	29.2 %	2.4%
8-pt DCT+8-pt IDCT	49792	3584	45.5 %	3.5%
4-pt DCT+ 4-pt IDCT	9024	512	45.1%	2.0%
8-pt DCT +4-pt DCT	12096	2048	18.1%	3.2%
8-pt DIT-FFT + 4-pt DIT-FFT	4160	0	29.7%	0%

10.2.2 Design area analysis of proposed hologram based obfuscation

The Table 10.10 shows that the gate count of DSP cores due to HLT based obfuscation [45] remains same as the baseline. Whereas, the gate count of proposed hologram based obfuscated DSP cores are decreased considerably compared to its baseline as shown in Table 10.9. Therefore, the proposed hologram based obfuscation technique offers on average 33.5% savings in gate count (design area) than [45].

10.2.3 Delay analysis of proposed hologram based obfuscation

The tree height transformation (THT) [45] impacts the critical delay of design. Therefore, for the chosen DSP benchmarks, the THT technique increases the tree height by one control step where an operation is executed using the respective functional unit. Because of this additional control step, the overall delay using [45] is increased. However, in case of proposed hologram based obfuscation, the switching elements of hologram obfuscation (which are the 2x1 Muxes) lead to the additional propagation delay. Since the propagation delay of a 2x1 Mux is considerably lower than a FU, hence lesser delay overhead is incurred than [45]. Thus, the proposed approach is capable to provide higher strength of obfuscation at lesser delay overhead than [45].

10.3 Results and analysis: Double line of defense approach using integrated structural obfuscation and crypto-steganography to secure IP cores

The experimental results of the proposed structural obfuscation and crypto-steganography based double line of defense technique discussed in chapter 5 are presented in this section. 15 nm open cell library [60] was used to calculate different design parameters such as area and delay. The proposed approach has been implementation in C++ and run on processor with specifications “4 GB, DDR3 memory at 1.9 GHz”. The experimental results have been analyzed on JPEG compression processor and DFT processor. The security of proposed technique and its impact on design cost are assessed as follows:

10.3.1 Security analysis

Table 10.12 Comparison of JPEG compression hardware resources pre and post obfuscation

	Resource configuration	Structural changes due to proposed obfuscation
Non-obfuscated JPEG hardware	4+, 8*, 12(8:1) mux, 12(16:1) mux, 6 (1:8 demux), 6 (1:16 demux)	10064 gates
Structurally obfuscated JPEG hardware	3+, 3*, 10(32:1) mux, 2(16:1) mux, 5 (1:32 demux), 1 (1:16 demux)	

Table 10.13 Security analysis in terms of strength of obfuscation for 4-point DFT processor

Strength of obfuscation of proposed design w.r.t baseline (unsecured)			Strength of obfuscation of proposed design w.r.t baseline (%) = Total gates affected due to obfuscation/ total gates in baseline
Due to difference in gate count	# of gates modified	Total gates affected	
336	4000	4336	(4336/5760)*100 = 75.28%

The security achieved through structural obfuscation based first line of defense is analyzed in terms of strength of obfuscation and the security achieved through crypto-based steganography is analyzed in terms of probability of coincidence metric for both JPEG compression processor and DFT processor.

10.3.1.1 Strength of structural obfuscation analysis

The strength of obfuscation of proposed technique has been defined in section 5.2.4 of chapter 5. The significant amount of structural obfuscation is achieved in terms of the following: (a) alterations in the interconnectivity of high level components such as adders, and multipliers etc. (b) alterations in the total count of interconnect binding units such as Muxes and Demuxes (c) alterations in the total count of storage units such as registers. Because of the internal architecture concealment using the proposed obfuscation, the attacker fails to realize his/her ill intentions of inserting malicious logic (backdoor) or steal the design. For the JPEG compression hardware, the impact of structural obfuscation on architectural resources and gate count is presented in Table 10.12. Further, the strength of obfuscation analysis for DFT processor is shown in Table 10.13.

10.3.1.2 Probability of coincidence analysis of crypto-steganography

For analyzing the security achieved using proposed crypto-steganography in terms of strength of IP ownership proof, the probability of coincidence (Pc) metric presented in eq. 5.1 in the section 5.2.4 of chapter 5 is used. Table 10.14 shows the Pc value achieved for JPEG compression processor for varying size of stego-constraints embedded in two different phases (register allocation and FU vendor allocation) of HLS design process. As evident from

Table 10.14 Security analysis (in terms of Pc) of proposed crypto-based steganography approach on varying size of stego-constraints for different design solutions of JPEG compression processor

Design solution	Pc			
	# of constraint =100	# of constraint =200	# of constraint =300	# of constraint =400
3+, 3*	1.6245e-3	4.4e-6	2.4e-7	9.89e-8
3+, 5*	1.732e-2	3.39e-4	2.228e-5	9.497e-6
5+, 5*	6.329e-2	4.913e-3	5.907e-4	2.518e-4
7+, 9*	2.092e-1	4.61e-2	1.515e-2	6.87e-3
9+, 9*	2.552e-1	6.496e-2	2.146e-2	1.051e-2
11+, 11*	3.001e-1	8.816e-2	2.821e-2	1.166e-2

Table 10.15 Comparison of security of 4-point DFT in terms of Pc with the entropy based steganography approach (*Note: G=14, f1=14, f2=10, y=2)*

Pc (proposed crypto-based steganography)	Pc (entropy based steganography)
5.72E-2	3.54E-01

the table, the Pc value reduces with the increasing size of stego-constraints. The reason is that the effective # of 0s (f1) and the effective # of 1s (f2) embedded into the design increase with the increasing size of stego-constraints. Thus, we can achieve lower value of Pc by choosing the large size of stego-constraints to enhance the strength of steganography. In addition, the resource constraints (design solutions) used also affect the Pc metric as shown in Table 10.14 as the number of 0s and 1s of the stego-constraints vary as per the chosen design solution.

Further, the Pc value in case of DFT processor is presented and compared with the entropy based steganography approach in Table 10.15. The proposed crypto-based steganography results in lower Pc because of implanting stego-information in two distinct phases of HLS compared to implanting only in single phase in the entropy based steganography. Hence, the proposed technique implants more digital evidence which leads to stronger proof of ownership.

10.3.2 Design cost analysis

We used eq. (3.8) as the metric to compute design cost of the proposed approach. Table 10.16 presents the design cost of the proposed approach for JPEG compression processor. As evident from the table, the design cost post crypto-steganography marginally increases for different size of stego-constraints. The reason is that, post embedding stego-constraints, the design may undergo more FU allocation of vendor type-2 than that of type-1. As the

Table 10.16 Design cost analysis of proposed approach on varying size of stego-constraints for different design solutions of JPEG compression processor

Design solution	Pre-steganography cost	Design cost of JPEG compression processor pre and post crypto-steganography			
		# of constraint =100	# of constraint =200	# of constraint =300	# of constraint =400
3+, 3*	0.2167	0.2167	0.2169	0.2173	0.2173
3+, 5*	0.1917	0.1920	0.1924	0.1929	0.1929
5+, 5*	0.1713	0.1713	0.1713	0.1719	0.1719
7+, 9*	0.1718	0.1720	0.1725	0.1729	0.1729
9+, 9*	0.1752	0.1754	0.1757	0.1763	0.1763
11+, 11*	0.1785	0.1785	0.1789	0.1794	0.1794

Table 10.17 Comparison of design cost of DFT processor with baseline and entropy based steganography

Metric	Baseline design	Proposed obfuscated and crypto-steganography secured design		Entropy based steganography
		Post obfuscation	Post steganography	
Cost	0.4674	0.4674	0.4680	0.466

area and latency of FUs of vendor type-2 is marginally higher than the vendor type-1, therefore it results in marginal increase in the cost.

Further, Table 10.17 shows the comparison of design cost of baseline N-point DFT processor with that of proposed secured design and entropy driven steganography based secured design. As shown, the design cost of proposed crypto-steganography is nominally affected due to changes in the size of Muxes.

10.4 Results and analysis: Double line of defense approach using integrated multi-key based structural obfuscation and physical level watermarking to secure IP cores

The experimental results of the proposed multi-key driven structural obfuscation and physical level watermarking based double line of defense technique discussed in chapter 6 are presented in this section. 15 nm open cell library [60] was used to calculate the design cost. The proposed approach has been implementation in java and run on processor with specifications "4 GB, DDR3 memory at 1.9 GHz". The experimental results have been analyzed for various DSP benchmarks. The security of proposed technique and its impact on design cost are assessed as follows:

10.4.1 Security analysis

The security achieved through multi-key driven structural obfuscation based first line of defense is analyzed in terms of strength of obfuscation and total

Table 10.18 Strength of multi-key structural obfuscation in terms of gate count modified

DSP benchmarks	Total gates in baseline (non-obfuscated)	Total gates in proposed approach	Difference in gate count due to proposed obfuscation
FIR	688	8832	8144
IIR	4464	4128	336
ARF	7360	5888	1472
DCT	3680	4208	528
Differential Equation	2560	8704	6144

Table 10.19 Total key size for the proposed obfuscated watermarked design

DSP benchmarks	Key size (in bits) of structural obfuscation
FIR	20
IIR	7
ARF	9
DCT	7
Differential Equation	17

Table 10.20 Security of proposed watermarking using probability of coincidence (Pc)

DSP benchmarks	Pc
FIR	8.93E-26
IIR	1.81E-11
ARF	8.6E-16
DCT	8.5E-10
Differential Equation	6.0E-37

key size and the security achieved through physical level watermarking is analyzed in terms of probability of coincidence and tamper tolerance for various DSP benchmarks.

10.4.1.1 Strength of multi-key structural obfuscation analysis

The strength of obfuscation of proposed technique has been defined in section 6.2.4 of chapter 6. Table 10.18 shows the comparative study between non-obfuscated versus proposed key based structurally obfuscated design. The change in gate count shown in Table 10.18 happens due to applying various phases of proposed multi-key structural obfuscation. Moreover, alterations in the interconnectivity of various resources also incorporate huge variation in the structure of the design. This makes the structure unidentifiable for an attacker, thus hindering the malicious attempt of backdoor insertion.

10.4.1.2 Total key size of multi-key structural obfuscation

Incorporation of multiple keys in the proposed obfuscation makes the decoding of the structural obfuscation highly challenging from an attacker's perspective. Hence, the security against the malicious intent of RE and backdoor insertion is enhanced. The total key size of proposed obfuscation technique has been defined in eq. (6.1) in section 6.2.4 of chapter 6. And, Table 10.19 reports the total key size for various DSP benchmarks.

10.4.1.3 Probability of coincidence of physical level watermarking

Table 10.21 Tamper tolerance (T_S) and brute-force attack analysis of proposed obfuscated watermarked design

DSP benchmarks	T_S (total combinations representing signature space)	Probability of finding WM signature using brute-force attack= ($1/T_S$)
FIR	$4.9*10^6$	2.0E-07
IIR	$7.3*10^2$	1.4E-03
ARF	$5.9*10^4$	1.7E-05
DCT	$7.3*10^2$	1.4E-03
Differential Equation	$3.9*10^8$	2.6E-09

Table 10.22 Design cost analysis of proposed approach

DSP benchmarks	Design cost of baseline design	Design cost of proposed approach
FIR	1	0.326
IIR	0.517	0.491
ARF	0.431	0.412
DCT	0.483	0.482
Differential Equation	0.76	0.455

For analyzing the security achieved in terms of strength of IP ownership proof using proposed physical level watermarking, the probability of coincidence (Pc) metric presented in eq. 6.2 in the section 6.2.4 of chapter 6 is used. Table 10.20 presents the Pc of the proposed approach. As shown, considerably lower Pc has been obtained for the various DSP benchmarks.

10.4.1.4 Tamper tolerance analysis

For analyzing the security achieved against removal attack on watermark, the tamper tolerance metric (T_S) presented in eq. 6.3 in the section 6.2.4 of chapter 6 is used. Table 10.21 shows the higher tamper tolerance of the proposed technique for the designer chosen watermark strength. Higher value of T_S represents stronger security against removal attack.

10.4.2 Design cost analysis

We used eq. (3.8) as the metric to compute design cost of the proposed approach. The design area is computed using the area of the enveloping rectangle of the floorplan, whereas the execution latency is determined from the scheduled design. Table 10.22 compares the design cost of proposed double line of defense technique wrt the baseline design. As shown in the table, the proposed technique does not incur overhead due to applying security. The overall design cost is reduced either due to substantial reduction in latency or floorplan area.

10.5 Results and analysis: Secured hardware accelerator design approach for image processing filters

The experimental results of the proposed secured hardware accelerator design approach for image processing filters discussed in chapter 7 are presented in this section. 15 nm open cell library [60] was used to calculate the design cost. The experimental results have been analyzed for generic 3x3 and 5x5 filters and various application specific 3x3 filters. The security of proposed technique and its impact on design cost are assessed as follows:

10.5.1 Security analysis

Structural obfuscation technique is employed to achieve security of proposed filter hardware accelerators against the threat of RE and potential backdoor insertion. Applying obfuscation techniques such as loop unrolling, graph partitioning and THT incur considerable alteration in the RTL datapath, without affecting the functionality. A structurally obfuscated netlist obtained in the proposed technique is considerably altered than the original version in terms of alteration in the overall gate count and gates connectivity. This prevents an attacker from reverse engineering the obfuscated netlist to identify the original structure.

10.5.1.1 Security analysis of generic 3×3 and 5×5 filter hardware accelerators

Table 10.23 shows the comparison of the RTL modules in the datapath of 3×3 and 5×5 filter hardware accelerators pre and post employing structural obfuscation. Further, impact of structural obfuscation at gate level is measured in terms of number of gates affected due to employing obfuscation techniques. Table 10.24 compares the gate count of 3×3 and 5×5 filter hardware accelerators pre and post employing structural obfuscation. As shown in the table, substantial change in number of gates is achieved. This is due to the fact that the count of datapath modules such as Muxes, Demuxes, FUs and registers and the size of Muxes and Demuxes noticeably changes post obfuscation. This change in the datapath is also reflected at gate level post logic synthesis, resulting in an obfuscated netlist.

10.5.1.2 Security analysis of application specific 3×3 filters hardware accelerators

Security of five image processing applications specific 3×3 filter hardware accelerators, achieved using structural obfuscation, is analyzed in terms of number of gates affected. Table 10.25 shows the comparison of the gate count of blur, sharpening, vertical embossment, horizontal embossment and Laplace edge detection filter before and after structural obfuscation. The structural obfuscation incurs huge change in gate count as shown in Table 10.25.

10.5.2 Design cost analysis

We used eq. (3.8) as the metric to compute the design cost of the proposed approach. Table 10.26 presents the design cost and power of the different application specific hardware accelerator. The result shows that the design

Table 10.23 RTL components of image processing filters of size 3x3 and 5x5 (note: 'M', 'A' and 'Reg' denote multipliers, adders and registers respectively)

Benchmarks	RTL components	
	Baseline	Proposed
3×3 Filter	M=1, A=1, 8x1 mux=2, 1x8 dmux=1, 16x1 mux=2, 1x16 dmux=1, Reg=20, Latches=6	M=3, A=1, 8x1 mux=15, 1x8 dmux=3, 16x1 mux=2, 1x16 dmux=1, Reg=86, Latches=12
5×5 Filter	M=5, A=2, 8x1 mux=10, 1x8 dmux=5, 16x1 mux=4, 1x16 dmux=2, Reg=76, Latches=21	M=5, A=2, 2x1 mux=12, 1x2 dmux=6, 4x1 mux=6, 1x4 dmux=3, 8x1 mux=14, 1x8 dmux=7, Reg=78, Latches=21

Table 10.24 Gate count of proposed filter hardware accelerators

Benchmarks	# Gates	
	Proposed processor	Proposed obfuscated processor
3×3 Filter	7408	19312
5×5 Filter	23456	22560

Table 10.25 Security analysis of proposed application specific image processing filters

Benchmarks	Gate count		Gate difference
	Proposed processor	Proposed obfuscated processor	
Blur Filter	3152	6544	3392
Sharpening Filter	3488	7232	3744
Vertical embossment Filter	1120	1792	672
Horizontal embossment Filter	1120	1792	672
Laplace edge detection Filter	2288	4368	2080

Table 10.26 Power and design cost of proposed application specific 3x3 filter hardware accelerators

Benchmarks	Power (µW)	Design cost
Blur Filter	21.257	0.682
Sharpening Filter	23.306	0.685
EV Filter	10.012	0.758
EH Filter	10.012	0.758
Laplace edge detection Filter	17.159	0.728

cost and power dissipation of the proposed designs are low.

10.6 Results and analysis: Techniques for securing functionally obfuscated DSP cores against removal attack

The experimental results of the proposed techniques for securing functionally obfuscated DSP cores against removal attack discussed in chapter 8 are presented in this section. 15 nm open cell library [60] was used to calculate the design cost. The experimental results for the following proposed techniques for securing functionally obfuscated DSP cores against the removal attack are presented in this section: (i) SHA-512 based ILBs-key generation technique (i) ARL based ILBs-key generation technique. Further, the experimental results are compared with [36] to analysis the impact of proposed techniques. The experimental results have been analyzed for various DSP benchmarks. The security of proposed techniques and its impact on design cost are assessed as follows:

10.6.1 Results and analysis for SHA-512 based ILB-keys generation technique

This proposed technique has been implementation on a processor with specifications "on MD A8- 4500M APU with 4 4 GB DDR3 memory at 1.9 GHz".

10.6.1.1 Security analysis

Security of the proposed SHA-512 based technique is assessed in terms of the number of encrypted keys (to reconfigure the ILBs) generated for securing a functionally obfuscated design. The security comparison of proposed technique with [36] is accomplished for the same number of locking key-bits for the ILBs of functionally obfuscated DSP cores. As presented in Table 10.27, the proposed technique generates more number of encrypted key bits for the ILBs in a functionally obfuscated design than [36]. This results in larger number of ILBs structure reconfiguration using the proposed technique. The reason is that the output of proposed key generation logic using SHA-512 is 512-bit long compared to 128 bit output of AES based key generation approach for ILBs [36]. Even, two AES blocks [36] are capable to provide 256

Table 10.27 Comparison of security in terms of key-bits encrypted for the ILBs of obfuscated design of proposed approach and [36]

DSP Core Benchmarks	No. of encrypted key-bits for ILBs using proposed approach	No. of encrypted key-bits for ILBs using [36]	% increase in security
IIR	512	256	50 %
Mesa Horner	512	256	50 %
DWT	512	256	50 %
ARF	512	256	50 %
FIR	512	256	50 %
JPEG IDCT	512	384	25 %
Mesa Interpolate	512	384	25 %

Table 10.28 Gate count comparison of baseline design with [36] and proposed methodology

DSP Benchmark	Gate count of baseline	Gate count of [36] (Obfuscated design with AES)	Gate count of proposed methodology (Obfuscated design with SHA-512 based key encryption hardware)	% reduction in gate count of proposed work wrt [36]
IIR	7360	90144	62336	30.84
Mesa Horner	12544	93792	65984	29.65
DWT	13088	95360	67552	29.16
ARF	19328	102688	74880	27.08
FIR	19360	104320	76512	26.66
JPEG IDCT	70880	218192	150816	30.88
Mesa Interpolate	74656	206256	138880	32.67

Table 10.29 Power comparison of baseline design, [36] and proposed work

DSP Benchmark	Power of baseline (in µW)	Power of [36] (in µW)	Power of Proposed methodology (in µW)	% reduction in power of proposed work wrt [36]
IIR	58.906	721.465	498.904	30.85
Mesa Horner	100.396	750.662	528.101	29.62
DWT	104.750	763.211	540.651	29.16
ARF	154.691	821.861	599.300	27.08
FIR	154.947	834.922	612.362	26.66
JPEG IDCT	567.286	1746.294	1207.052	30.88
Mesa Interpolate	597.507	1650.764	1111.522	32.67

key bits when implemented for IIR, Mesa Horner, DWT, ARF, FIR as shown in Table 10.27. This is still 50% lesser secured than proposed SHA-512 based key generation logic. For larger DSP circuits such as JPEG IDCT, Mesa Interpolate etc., three AES instances can concurrently provide 384 key bits. However, the proposed one SHA-512 based logic can provide 512 key bits for ILBs reconfiguration. Average 43.75% enhancement in security is offered by the proposed approach than [36].

10.6.1.2 Area and power overhead analysis

The gate count of the proposed technique and [36] is compared w.r.t. baseline (non-obfuscated DSP designs) as shown in Table 10.28. As shown, the gate count overhead of proposed technique is significantly lower than [36]. This is because of using more than one instance of AES to enable concurrent reconfiguration of several ILBs in [36]. This results in gate count overhead using [36] than the proposed technique and baseline. Further, the power

Table 10.30 Comparison of design cost of proposed work with [36]

DSP Benchmark	Design cost of proposed work	Design cost of [36]	% reduction in cost
IIR	0.827	0.987	16.2
MESA HORNER	0.719	0.880	18.3
DWT	0.819	0.981	9.2
ARF	0.619	0.775	20.1
FIR	0.604	0.754	19.9
JPEG IDCT	0.419	0.619	32.4
MESA INTERPOLATE	0.319	0.519	38.5

comparison of baseline design with [36] and proposed technique is presented in Table 10.29. The power overhead of proposed technique in contrast to baseline design is lower than [36]. This is because, the design overhead due to AES block [36] becomes high for improving the security. This results in larger gate count and thus higher power requirement than proposed technique. This indicates that the proposed technique is a low power and high secure solution than [36] against removal attack.

10.6.1.3 Design cost analysis

In computation of the design cost, we included area and power as the design parameters, where both were given same weightage. Table 10.30 presents the design cost of obfuscated DSP cores. As evident from the table, the proposed technique offers significant reduction in the design cost than [36]. This reduction is achieved owing to integration of relatively lightweight SHA-512 based key encryption hardware compared to AES hardware used in [36] to achieve enough security.

10.6.2 Results and analysis for ARL based ILB-keys generation technique

The proposed ARL unit offers a lightweight, low power and more secure solution against removal attack than [36]. The experimental results were analyzed in terms of security and area (gate count) overhead.

10.6.2.1 Security analysis

The security is analyzed in terms of number of input (challenge) bits needed to produce the same number of key bits for ILBs reconfiguration using the proposed unit and [36]. The substantial improvement in security using proposed ARL unit compared to [36] is graphically shown in Fig. 10.2. The reason is as follows. In the results, upto 32 ILBs are reconfigured which require 256 bits generated from 2 instances of AES128 in case of [36]. The two instances of AES require 512 bits of input (2*(128+128)) to generate 256 bits for ILB keys. And for larger designs such as Mesa Interpolate and JPEG IDCT, minimum 48 ILBs are reconfigured (which require three AES instances). However to reconfigure the same number of ILBs using the proposed ARL unit, more number of challenge/input bits are required to generate the ILB keys. This provides more security against guessing correct key bits used for ILBs reconfiguration. An average 66.67% more input bits than [36] are required to be known by an attacker in order to deduce the correct key bits.

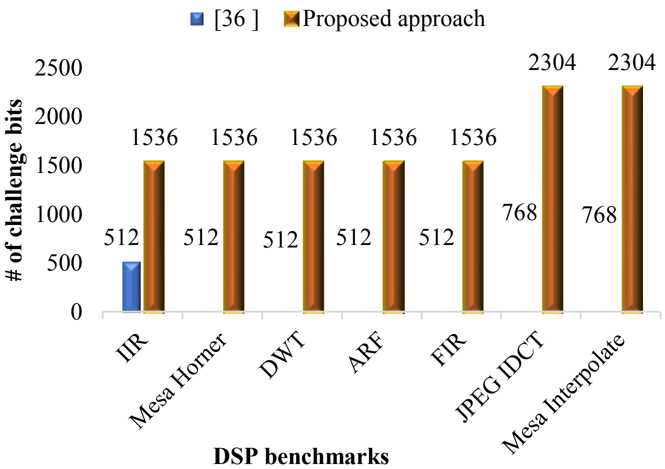


Fig. 10.2 Security comparison of proposed work with [36]

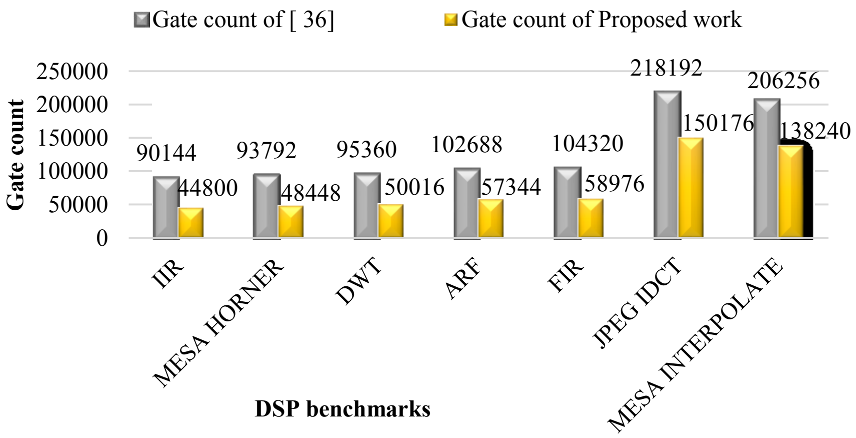


Fig. 10.3 Gate count comparison of proposed work with [36]

10.6.2.2 Area overhead analysis

The proposed ARL unit results in lesser area or gate count overhead than [36] as shown in Fig. 10.3. The reason is as follows. For generating key bits for the same number of ILBs, two or three instances of AES are required as discussed earlier in section 10.6.2.1. This leads to substantial gate count overhead than the proposed ARL unit. The proposed technique offers an average 42.57% reduction in gate count than [36].

10.7 Results and analysis: Robust logic locking technique for preventing IP piracy

The experimental results of the proposed robust logic locking technique for preventing IP piracy discussed in chapter 9 are presented in this section. The proposed approach has been implemented on a processor with specifications "2GB RAM and processor frequency of 2.4 GHz". The results are calculated for the DSP benchmarks of size 32-bit (i.e. size of input, output and FUs is 32-bit). The security of the proposed technique and its impact on design overhead are assessed as follows:

10.7.1 Security analysis

The security of proposed logic locking technique is analyzed in terms of the following:

10.7.1.1 Probability of finding valid key in exhaustive trials

This probability P_v is measured using eq. (9.2) discussed in section 9.2.5 of chapter 9. In case of the existing logic locking techniques [36], [37], the eq. (9.2) converges into eq. (9.3) and in case of the proposed technique it converges into eq. (9.4). The comparison of P_v of proposed technique with the existing technique [36], [37] is shown in Table 10.31. The table shows that we obtained the P_v very lesser than [36], [37] despite encoding lesser key bits. The higher security using the proposed logic locking is achieved because of very less probability of deducing the correct key in exhaustive trials. However, in case of existing approaches [36], [37], it is likely to find the correct key in exhaustive trials with probability 1 hence the attack time to find the key is projected to be finite. Since using the proposed technique, the correct key

Table 10.31 Comparison of the strength of the proposed logic locking with [36] [37] in terms of the probability of deducing correct key in exhaustive trials (P_v) using eq. (9.3) and (9.4)

DSP Core Benchmarks	# of key bits in locked design of [37]	# of key bits in locked design of [36]	# of key bits in locked design of proposed work	P_v using exhaustive (2^{K_b}) trials in [36] [37]	P_{CK} using exhaustive (2^{K_b}) trials in proposed work	# of times security enhancement in proposed approach w.r.t. [36] [37]
IIR	384	768	192	1	1.6E-58	6.3E+57
Mesa Horner	384	768	192	1	1.6E-58	6.3E+57
DWT	384	512	128	1	2.9E-39	3.4E+38
ARF	448	1024	256	1	8.6E-78	1.2E+77
FIR	576	1280	320	1	4.7E-97	2.1E+96
JPEG IDCT	1728	5376	1344	1	2.6E-405	3.8E+404

Table 10.32 Attack time comparison of the proposed logic locking with [36] [37] (note: In proposed logic locking, correct key cannot be deduced in exhaustive trials except first)

DSP Benchmark	Estimated attack time through proposed work	Estimated attack time through [37] (in years)	Estimated attack time through [36] (in years)
IIR	∞	1.2E+99	4.9E+214
Mesa Horner	∞	1.2E+99	4.9E+214
DWT	∞	1.2E+99	4.2E+137
ARF	∞	2.3E+118	5.7E+291
FIR	∞	7.8E+156	6.6E+368
JPEG IDCT	∞	4.8E+503	6.9E+1601

Table 10.33 Comparison of the encryption strength of the proposed logic locking with [36] [37]

DSP Benchmarks	# of key bits in the design of [37]	% of key bits encrypted of [37] (using (9.5))	# of key bits in the design of [36]	% of key bits encrypted of [36] (using (9.5))	# of key bits in the design using proposed	% of key bits encrypted of proposed (using (9.5))
IIR	384	33.3%	768	16.7%	192	66.7%
Mesa Horner	384	33.3%	768	16.7%	192	66.7%
DWT	384	33.3%	512	25.0%	128	100%
ARF	448	28.6%	1024	12.5%	256	50.0%
FIR	576	22.2%	1280	10.0%	320	40.0%
JPEG IDCT	1728	7.4%	5376	2.4%	1344	9.5%

cannot be deduced in the exhaustive trials except first, hence the attack time to find the valid key is projected to be infinite (indicating that the time taken will be much higher than that of obtaining the key using brute-force). Table 10.32 shows the attack time comparison of proposed technique with [36], [37] (supposing that 1 billion (10^9) keys can be fed per second.

10.7.1.2 Encryption strength analysis

The encryption strength En^s of the proposed logic locking is measured using eq. (9.5) discussed in section 9.2.5 of chapter 9. Table 10.33 shows the comparison of encryption strength of the proposed technique with the existing approaches [36], [37]. As shown, the proposed technique encrypts higher percentage of key-bits w.r.t. [36], [37] using one AES. The reason is that the

Table 10.34 Percent reduction in the resource count using proposed work w.r.t. [36] [37]

DSP Benchmarks	Proposed work					
	% reduction in NAND gates w.r.t. [37]	% increase in D-FFs w.r.t. [37]	Overall % reduction in resource count w.r.t. [37]	% reduction in NAND gates w.r.t. [36]	% increase in D-FFs w.r.t. [36]	Overall % reduction in resource count w.r.t. [36]
IIR	40.7%	3.7%	37.0%	7.4%	3.7%	3.7%
Mesa Horner	32.7%	3.7%	29.0%	6.5%	3.7%	2.8%
DWT	27.3%	2.5%	24.8%	4.2%	2.5%	1.7%
ARF	24.0%	4.7%	19.3%	7.1%	4.7%	2.4%
FIR	33.4%	5.9%	27.5%	8.5%	5.9%	2.6%
JPEG IDCT	30.4%	19.1%	11.3%	13.9%	19.1%	0%

proposed technique requires lesser key bits than the existing techniques, while deploying the same number of ILBs in the design.

10.7.2 Design area analysis

Design area of the proposed and existing techniques [36], [37] is assessed in terms of resource count which is calculated as the count of total required NAND gates and D-FFs. Table 10.34 presents the % reduction in NAND gates and % increase in D-FFs using proposed technique compared to existing techniques [36], [37]. Furthermore, the Table 10.34 presents the overall % reduction in resource count achieved through proposed approach. The overall % reduction in the resource count is computed in terms of an algebraic summation of '% decrease in total NAND gates' and '% increase in total D-FFs'. Table 10.34 shows that the proposed technique achieves on average 25.8% reduction in the resource count with respect to [37] and on average 1.9% reduction with respect to [36]. This indicates that the proposed technique incurs lesser resource count and hence lesser area than existing techniques [36], [37].

Chapter 11

Conclusion and Future work

11.1 Conclusion

The DSP and multimedia applications are prevailing in the modern consumer electronics. Therefore, the application specific IP cores or hardware accelerator of DSP and multimedia applications are important part of modern SoC designs. However, different parties or houses involved in the SoC design process are situated offshore/globally. This renders the DSP and multimedia IP cores vulnerable to various kinds of hardware security threats such as IP piracy and hardware Trojan insertion. These threats may pose substantial impact on system, services, and users and as well as on IP designer itself. This thesis presented novel techniques for generating secure IP cores or hardware accelerators to counter the aforementioned hardware threats. The following objectives were accomplished:

- Proposed an IP core steganography based hardware security solution to address the threat of IP piracy by implanting the vendor’s stego-mark into the design during register allocation phase of HLS process. The vendor’s stego-mark or steganography information to be implanted into the design is monitored using a controlling parameter called entropy threshold. The implanted stego-information incurred negligible design overhead and produced lower design cost compared to signature-based IP core protection techniques.
- Proposed an IP core steganography based hardware security solution that generates a robust stego-mark using a large size stego-key and a chain of SHA-512 hash blocks. Thus generated stego-constraints cannot be easily regenerated or back tracked by an adversary to compromise the stego-mark and misuse it to escape counterfeit detection. Thus the proposed approach is capable to offer higher security against the IP piracy threat and outperforms the related works.
- Proposed hologram based structural obfuscation technique that hinders reverse engineering by camouflaging the functionality of one design

into another, hence preventing against Trojan insertion attack. Likewise a security image hologram, where two distinct images are camouflaged and displayed at different viewing angles, the proposed work merges two distinct DSP cores such that each one functions at a particular bit pattern. Therefore, proposed obfuscation has been called as hologram based obfuscation which obfuscates two DSP kernels simultaneously by camouflaging functionality of one DSP kernel into another.

- Proposed an integrated structural obfuscation and crypto-steganography based double line of defense mechanism to address both IP piracy and hardware Trojan insertion threats simultaneously. The proposed approach has been applied on JPEG compression processor and DFT processor.
- Proposed an integrated multi-key based structural obfuscation and physical level watermarking based double line of defense mechanism to address both IP piracy and hardware Trojan insertion threats simultaneously. The structural obfuscation is performed using key-driven partition and key-driven folding knob based transformations combined with key-driven loop unrolling, key-driven ROE and key-driven THT, and the proposed physical-level watermarking is performed through early floorplanning of obfuscated DSP circuit.
- Proposed a HLS driven technique to generate secured hardware accelerator designs for image processing filters. We addressed the problem of stringent performance and low power requirement of image processing applications by designing dedicated hardware for image processing filters. We also handled the threat of Trojan insertion through RE by designing structurally obfuscated versions of filters hardware.
- Proposed SHA-512 based key generation hardware and ARL based key generation unit to secure the IP core locking blocks (ILBs) used in functionally obfuscated DSP cores against the removal attack. The security is achieved by offering re-configurability to ILBs structures

based on the output of proposed key generation hardware units. Proposed logics provide higher security than related work.

- Proposed logic locking of DSP circuits using highly robust DSP locking cells (DLCs) comprising of combinational gates and flip-flops. Proposed DLCs are reconfigurable based on AES128 output. The DLCs are inserted in register transfer level (RTL) datapath based on designer selected value of ‘ω’. Proposed DLCs are capable of hindering the probability of obtaining correct key (<<1) even through exhaustive trials, thus rendering the brute force attack ineffective. The proposed approach achieved higher security at lower design overhead than the existing logic locking techniques.

11.2 Future work

In the conventional hardware security techniques for IP cores authentication, designer’s secret information is not uniquely associated with the designer’s identity. Therefore in future, we target to offer unique/non-conflicting authentication to IP cores. To do so, we aim at exploiting the vendor’s unique information based on physiological or behavioral biometric traits to offer robust detective control on IP piracy.

REFERENCES

[1] Sengupta A. (2020). Secured hardware accelerators for DSP and image processing applications. *The Institute of Engineering and Technology (IET) Book*, e-ISBN: 9781839533075.

[2] Sengupta A. (2020). Frontiers in securing IP cores - Forensic detective control and obfuscation techniques. *The Institute of Engineering and Technology (IET) Book*, ISBN-10: 1-83953-031-6, ISBN-13: 978-1-83953-031-9.

[3] Colombier B. (2017). Methods for protecting intellectual property of IP cores designers. *Micro and nanotechnologies/Microelectronics, Université de Lyon*, NNT : 2017LYSES038.

[4] Colombier, B, & Bossuet, L. (2014). Survey of hardware protection of design data for integrated circuits and intellectual properties. *IET Computers &amp; Digital Techniques*, 8, (6), p. 274-287.

[5] Saha D. & Sur-Kolay S. (2011). SoC: A real platform for IP reuse, IP infringement, and IP protection. *CAD for Gigascale SoC Design and Verification Solutions*, vol. 2011, doi:10.1155/2011/731957.

[6] Schneiderman R. (2010). DSPs evolving in consumer electronics applications. *IEEE Signal Process. Mag.*, vol. 27(3), pp. 6–10.

[7] Mahdiany H. R., Hormati A. and Fakhraie S. M. (2001). A hardware accelerator for DSP system design. in *Proc. ICM*, pp. 141-144.

[8] Sengupta A., Mohanty S. P (2019). IP core protection and hardware-assisted security for consumer electronics. *The Institute of Engineering and Technology (IET) Book,* ISBN: 978-1-78561-799-7, e-ISBN: 978-1-78561-800-0.

[9] Sengupta, A. (2016). Cognizance on Intellectual Property: A High-Level Perspective. *IEEE Consumer Electronics Magazine*, *5*(3), 126-128.

[10] Chakraborty R. S., Bhunia S. (2009). HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28 (10), pp. 1493-1502.

[11] Sengupta, A. (2016). Intellectual property cores: Protection designs for CE products. *IEEE Consumer Electronics Magazine*, *5*(1), 83-88.

[12] Plaza S. M., Markov I. L. (2015). Solving the third-shift problem in IC Piracy with test-aware logic locking. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*., vol. 34(6), pp. 961-971.

[13] Castillo E., Meyer-Baese U., Garcia A., Parilla L., Lloris A. (2007). IPP@HDL: Efficient intellectual property protection scheme for IP cores. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,* vol. 15, no. 5, pp. 578–590.

[14] Gorman, C. (2012). Counterfeit Chips on the Rise. *IEEE Spectrum*. 49. 16-17. 10.1109/MSPEC.2012.6203952.

[15] Guin U., Huang K., DiMase D., Carulli J. M., Tehranipoor M. and Makris Y. (2014). Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain. *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207-1228.

[16] Mitra S., Wong H.P. and Wong S. (2015). The Trojan-proof chip. *IEEE Spectrum*, vol. 52, no. 2, pp. 46-51.

[17] Tehranipoor, M., & Koushanfar, F. (2010). A survey of hardware Trojan taxonomy and detection. *IEEE design & test of computers*, *27*(1).

[18] Bhunia, S., Hsiao, M. S., Banga, M., & Narasimhan, S. (2014). Hardware Trojan attacks: threat analysis and countermeasures. *Proceedings of the IEEE*, *102*(8), 1229-1247.

[19] McFarland, M. C., Parker, A. C., & Camposano, R. (1990). The high-level synthesis of digital systems. *Proceedings of the IEEE*, *78*(2), 301-318.

[20] McFarland, M. C., Parker, A. C., & Camposano, R. (1988). Tutorial on high-level synthesis. In *Proceedings of the 25th ACM/IEEE Design Automation Conference* (pp. 330-336). IEEE Computer Society Press.

[21] Coussy, P., & Morawiec, A. (Eds.). (2008). *High-level synthesis: from algorithm to digital circuit*. Springer Science & Business Media.

[22] Gajski, D. D., Dutt, N. D., Wu, A. C., & Lin, S. Y. (2012). *High—Level Synthesis: Introduction to Chip and System Design*. Springer Science & Business Media.

[23] Rajendran, J., Zhang, H., Sinanoglu, O., & Karri, R. (2013). High-level synthesis for security and trust. In *On-Line Testing Symposium (IOLTS), 2013 IEEE 19th International,* pp. 232-233.

[24] Mishra, V. K., & Sengupta, A. (2014). MO-PSE: Adaptive multi-objective particle swarm optimization based design space exploration in architectural synthesis for application specific processor design. *Advances in Engineering Software*, *67*, 111-124.

[25] Sengupta A., Bhadauria S. (2016). Exploring low cost optimal watermark for reusable IP cores during high level synthesis. *IEEE Access*, vol. 4, pp. 2198-2215.

[26] Hong I. and Potkonjak M. (1999). Behavioral synthesis techniques for intellectual property security. in *Proc. DAC*, pp. 849–854.

[27] Kirovski D., Hwang Y., Potkonjak M. and Cong J. (2006). Protecting combinational logic synthesis solutions. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25(12), pp. 2687-2696.

[28] Cui A., Chang C. H. and Tahar S. (2008). IP watermarking using incremental technology mapping at logic synthesis level. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27(9), pp. 1565-1570.

[29] Cui A. and Chang C. (2007). Watermarking for IP protection through template substitution at logic synthesis level. in *Proc. ISCAS*, New Orleans, LA, pp. 3687-3690.

[30] Koushanfar, F., Hong, I., & Potkonjak, M. (2005). Behavioral synthesis techniques for intellectual property protection. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, *10*(3), 523-545.

[31] Sengupta A., Roy D. (2017). Antipiracy-Aware IP Chipset Design for CE Devices: A Robust Watermarking Approach. *IEEE Consumer Electronics Mag*, vol. 6(2), pp. 118-124.

[32] Le Gal B., Bossuet L. (2012). Automatic low-cost IP watermarking technique based on output mark insertions. *Design Autom. Embedded Syst.*, vol. 16(2), pp. 71-92.

[33] Sengupta A., Roy D. (2018). Multi-Phase Watermark for IP Core Protection. *Proc. 36th IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1-3.

[34] Sengupta A., Roy D., Mohanty S. P. (2018). Triple-Phase Watermarking for Reusable IP Core Protection During Architecture Synthesis. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 37(4), pp. 742-755.

[35] Roy J. A., Koushanfar F., and Markov I. L. (2008). EPIC: Ending piracy of integrated circuits. in *Proc. Design Autom. Test Europe*, pp. 1069–1074.

[36] Sengupta A., Kachave D. and Roy D. (2019). Low cost functional obfuscation of reusable IP cores used in CE hardware through robust locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38(4), pp. 604-616.

[37] Yasin, M., Rajendran, J. J., Sinanoglu, O., & Karri, R. (2016). On improving the security of logic locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *35*(9), 1411-1424.

[38] Zhang, J. (2016). A Practical Logic Obfuscation Technique for Hardware Security. *IEEE Trans. VLSI Syst.*, *24*(3), 1193-1197.

[39] Yasin M., Mazumdar B., Sinanoglu O., and Rajendran J. (2017). Removal attacks on logic locking and camouflaging techniques. *IEEE Transactions on Emerging Topics in Computing*, doi: 10.1109/TETC.2017.2740364.

[40] Subramanyan P., Ray S., and Malik S. (2015). Evaluating the security of logic encryption algorithms. in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 137-143.

[41] Xie Y. and Srivastava A. (2019). Anti-SAT: Mitigating SAT Attack on Logic Locking. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38(2), pp. 199-207.

[42] Torrance, R., & James, D. (2009). The state-of-the-art in IC reverse engineering. In *Cryptographic Hardware and Embedded Systems-CHES 2009* (pp. 363-381). Springer, Berlin, Heidelberg.

[43] Chakraborty R. S., Narasimhan S. and Bhunia S. (2009). Hardware Trojan: Threats and emerging solutions. *IEEE International High Level Design Validation and Test Workshop*, pp. 166-171.

[44] Lao Y. and Parhi K. K. (2015). Obfuscating DSP Circuits via High-Level Transformations. *IEEE Trans. on Very Large Scale Integr. (VLSI) Syst.*, vol. 23(5), pp. 819–830.

[45] Sengupta A., Roy D., Mohanty S.P., and Corcoran P. (2017). DSP Design Protection in CE through Algorithmic Transformation based

Structural Obfuscation. *IEEE Transactions on Consumer Electronics*, vol. 63(4), pp. 467 – 476.

[46] Sengupta A. and Roy D. (2017). Protecting an intellectual property core during architectural synthesis using high-level transformation based obfuscation. *IET Electronics Letters*, vol: 53(13), pp. 849 – 851.

[47] Sengupta A., Roy D., Mohanty S.P., and Corcoran P. (2018). Low-cost obfuscated JPEG CODEC IP core for secure CE hardware. I*EEE Trans. Consum. Electron*., vol. 64(3), pp. 365–374.

[48] Sengupta A., Mohanty S. P., Pescador F., Corcoran P. (2018). Multi-Phase Obfuscation of Fault Secured DSP Designs with Enhanced Security Feature. *IEEE Transactions on Consumer Electronics*, vol. 64(3), pp: 356-364.

[49] Sengupta, A. (2015). Exploration of kc-cycle transient fault-secured datapath and loop unrolling factor for control data flow graphs during high-level synthesis. *Electronics Letters*, *51*(7), 562-564.

[50] Benda L., Mudry P., Ijspeert A. (2008). Hardware acceleration for image processing. *Technical report EPFL*, [online] Available: http://biorob2.epfl.ch/pages/studproj90/birg67936/rapport.pdf.

[51] Dutta H., Hannig F., Teich J., Heigl B., and Hornegger H. (2006). A design methodology for hardware acceleration of adaptive filter algorithms in image processing. in *Proc. ASAP*, Steamboat Springs, CO, USA, pp. 331–340,

[52] Ortega-Cisneros S. et al. (2014). Real time hardware accelerator for image filtering. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, CIARP*, pp. 80-87.

[53] Sengupta A., Mohanty S. P. (2019). Advanced encryption standard (AES) and its hardware watermarking for ownership protection. *Book: IP Core Protection and Hardware-Assisted Security for Consumer Electronics.* e-ISBN: 9781785618000, pp. 317-335.

[54] Obukhov A., Kharlamov A. (2008). Discrete Cosine Transform for 8x8 Blocks with CUDA. *Nvidia whitepaper document*.

[55] Matrix Technologies, Hologram Features (2016). [Online]. Available: http://www.matrixtechnologies.in/hologram-features.html, last accessed on May 2019.

[56] Kachave D., Sengupta A. (2016). Integrating physical level design and high level synthesis for simultaneous multi-cycle transient and multiple transient fault resiliency of application specific datapath processors. *Microelectronics Reliability*, vol. 60, pp. 141-152.

[57] Sait, S. M., & Youssef, H. (1999). *VLSI physical design automation: theory and practice* (Vol. 6). World Scientific Publishing Company.

[58] Express benchmark suite, University of California San Diego, 2016, https://www.ece.ucsb.edu/EXPRESS/benchmark/.

[59] Mohanty S. P., Ranganathan N., Kougianos E., Patra, P. (2008). Low-power highlevel synthesis for nanoscale CMOS circuits. Springer Science & Business Media.

[60] 15 nm open cell library. [Online]. Available: https://si2.org/open-cell-library/, last accessed on Jan. 2020.