ALTERNATIVE PARADIGMS OF HARDWARE SECURITY FOR ADDRESSING THREATS OF IP PIRACY AND TROJAN DURING HIGH LEVEL SYNTHESIS

Ph.D. Thesis

By
ADITYA ANSHUL

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
DECEMBER 2024

ALTERNATIVE PARADIGMS OF HARDWARE SECURITY FOR ADDRESSING THREATS OF IP PIRACY AND TROJAN DURING HIGH LEVEL SYNTHESIS

A THESIS

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

by
ADITYA ANSHUL

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
DECEMBER 2024

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **ALTERNATIVE PARADIGMS OF HARDWARE SECURITY FOR ADDRESSING THREATS OF IP PIRACY AND TROJAN DURING HIGH LEVEL SYNTHESIS** 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 AUGUST, 2021 to NOVEMBER 2024 under the supervision of Prof. ANIRBAN SENGUPTA, 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
(ADITYA ANSHUL)

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
(Prof. ANIRBAN SENGUPTA)

ACKNOWLEDGEMENTS

I would like to extend my heartfelt appreciation to everyone who contributed to making this journey both enriching and memorable. First and foremost, I am deeply grateful to my supervisor, **Prof. Anirban Sengupta**, for granting me the privilege to work under his guidance. His unwavering support, dedication, and belief in my potential have been invaluable throughout this process. His mentorship not only deepened my understanding of research but also instilled in me a strong sense of commitment and perseverance. Through his direction, I have grown both professionally and personally, learning the significance of work ethics, intellectual curiosity, and the patience required to achieve meaningful academic progress.

I also owe a mention to **Prof. Abhishek Srivastava** and **Prof. Ankhi Roy** for their valuable feedback on my research work throughout these years.

I am also grateful to **Dr. Ranveer Singh**, Head of the Department of Computer Science & Engineering, for extending all necessary support to me. My sincere acknowledgement and respect to **Prof. Suhas Joshi**, Director, Indian Institute of Technology Indore, for providing me the opportunity to explore my research capabilities at the Indian Institute of Technology Indore.

I wish to thank all the faculty members, my seniors Dr. Mahendra Rathor, Dr. Rahul Chaurasia, and lab mates Vishal Chaurasia, Nabendu Bhui, and friends for their continuous support.

Further, I wish to express my deepest gratitude to my wife (Soumya) and parents (Sanjay and Poonam Tiwari), for their strong belief in me and for their continuous support all the way. I also wish to express my gratitude to my brothers for being the driving force of my career and for emotional support throughout the PhD work. I am thankful to all for being with me every single moment to keep me motivated to work for the past years.

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

Adittya Anshul

DEDICATED TO MY WIFE AND PARENTS

ABSTRACT

The use of hardware intellectual property (IP) cores has become a key design approach in modern electronics, particularly in electronics, computing, and multimedia systems. This popularity stems from the ability of IP cores to enhance performance and efficiency by accelerating application processes. In addition, growing design complexities, shorter product lifecycles, and increasing pressure to bring products to market faster (time-to-market factor) have driven the adoption of reusable IP cores in modern system-on-chip (SoC) designs. These factors, coupled with the data and computation intensive nature of many modern applications, have made IP cores an essential component in these designs. However, the growing reliance on third-party IP vendors and a globally distributed design supply chain has introduced significant security risks. As IP cores are often sourced from multiple vendors to expedite the design process, the involvement of external parties create vulnerabilities that can be exploited by malicious actors. An untrusted design house, particularly in offshore locations, might steal or tamper with the IP core design, leading to piracy or malicious alterations, compromising the integrity and safety of the final product. On the other hand, it is also essential for the SoC integrator to demarcate between authentic and pirated products before integration into final SoC design to ensure the safety and security of end consumers.

Applications that rely on multimedia, digital signal processing (DSP), image processing, healthcare, and machine learning have become more prevalent, making IP cores even more essential in sectors like healthcare, military, robotics, artificial intelligence, etc. These cores enable critical functions in advanced systems, making them a fundamental part of the technological infrastructure. As SoC designers increasingly integrate reusable IP cores sourced from various global vendors, the complexity of the IP supply chain amplifies security concerns. The most significant threats include IP piracy, IP counterfeiting, unauthorized claim of IP ownership, and hardware Trojans insertion. Given these risks, securing IP core has become a critical concern. Designer must implement comprehensive security measures to ensure that the hardware remains trustworthy throughout its lifecycle. Further, it is also

essential to incorporate low-cost design solutions to generate an optimal secure hardware IP core.

For complex, data-heavy applications such as image processing, DSP, multimedia, healthcare, and machine learning, securing reusable hardware IP cores requires specialized approaches. One solution that has gained traction is the use of high-level synthesis (HLS) framework. This framework provides a means to incorporate security mechanisms into IP core designs at higher abstraction of design levels, while maintaining design flexibility and reducing complexity. HLS allows designers to integrate robust security features with minimal impact on the overall design cost, making it an attractive solution for securing IP core/design. One of the key features of HLS is the design space exploration (DSE) framework, which facilitates the generation of low-cost secure design solutions. This thesis provides several alternative paradigm for securing hardware IPs against IP piracy and hardware Trojan during HLS. Towards the security of IP cores, this thesis contributes the following: (a) low-cost multiphase encryption and low-cost crypto-chain signature base security approaches against IP piracy and false IP ownership claim, (b) designing enhanced security framework for hardware IPs using IP seller's protein molecular biometrics and facial biometric-based encryption key, (c) exploiting statistical hardware watermarking technique using encrypted dispersion matrix and eigen decomposition framework, (d) securing GLRT cascade hardware IP design framework for ECG detector (e) designing voice biometric-based hardware watermarking framework, and (f) designing HLS-based low-cost (optimal) functional trojan-resistant hardware IP design framework.

LIST OF PUBLICATIONS (32)

PEER-REVIEWED JOURNALS (13):

1. A. Sengupta, R. Chaurasia and **A. Anshul**, "Robust Security of Hardware Accelerators Using Protein Molecular Biometric Signature and Facial Biometric Encryption Key," ***IEEE Transactions on Very Large Scale Integration (VLSI) Systems***, vol. 31, no. 6, pp. 826-839, June 2023, doi: 10.1109/TVLSI.2023.3265559, (**Impact Factor**: 2.8).

2. M. Rathor, **A. Anshul**, Anirban Sengupta, "Securing Reusable IP Cores using Voice Biometric based Watermark", ***IEEE Transactions on Dependable and Secure Computing***, vol. 21, no. 4, pp. 2735-2749, July-Aug. 2024, doi: 10.1109/TDSC.2023.3315780, (**Impact Factor**: 7.0).

3. M. Rathor, A. Sengupta, R. Chaurasia and **A. Anshul**, "Exploring Handwritten Signature Image Features for Hardware Security," ***IEEE Transactions on Dependable and Secure Computing***, vol. 20, no. 5, pp. 3687-3698, 1 Sept.-Oct. 2023, doi: 10.1109/TDSC.2022.3218506, (**Impact Factor**: 7.0).

4. A. Sengupta, **A. Anshul**, Secure hardware IP of GLRT cascade using color interval graph based embedded fingerprint for ECG detector. ***Nature Scientific Reports*****,** 14, 13250 (2024), doi: 10.1038/s41598-024-63533-7, (**Impact Factor**: 3.8).

5. A. Sengupta, **A. Anshul**, V. Chourasia and N. Kumar, "M-HLS: Malevolent High-Level Synthesis for Watermarked Hardware IPs," ***IEEE Embedded Systems Letters***, Accepted, June 2024, doi: 10.1109/LES.2024.3416422, (**Impact factor**: 1.7).

6. A. Sengupta and **A. Anshul**, "Watermarking Hardware IPs Using Design Parameter Driven Encrypted Dispersion Matrix With Eigen Decomposition Based Security Framework," ***IEEE Access***, vol. 12, pp. 47494-47507, 2024, doi: 10.1109/ACCESS.2024.3382202, (**Impact Factor**; 3.4).

7. A. Sengupta and **A. Anshul**, "A Survey of High Level Synthesis based Hardware (IP) Watermarking Approaches," ***IEEE Design & Test***, Accepted, 2024, doi: 10.1109/MDAT.2024.3435056, (**Impact factor**: 1.9).

8. **A. Anshul** and A. Sengupta, "A Survey of High Level Synthesis Based Hardware Security Approaches for Reusable IP Cores," ***IEEE Circuits and Systems Magazine***, vol. 23, no. 4, pp. 44-62, Fourthquarter 2023, doi: 10.1109/MCAS.2023.3325607, (**Impact Factor**: 5.6).

9. R. Chaurasia, **A. Anshul**, A. Sengupta and S. Gupta, "Palmprint Biometric Versus Encrypted Hash Based Digital Signature for Securing DSP Cores Used in CE Systems," ***IEEE Consumer Electronics Magazine***, vol. 11, no. 5, pp. 73-80, 1 Sept. 2022, doi: 10.1109/MCE.2022.3153276, (**Impact Factor**: 3.7*)*.

10. **A. Anshul**, A. Sengupta, PSO based exploration of multi-phase encryption based secured image processing filter hardware IP core datapath during high level synthesis, ***Elsevier Expert Systems with Applications***, Volume 223, 2023, 119927, doi: 10.1016/j.eswa.2023.119927, (**Impact Factor**: 7.5).

11. **A. Anshul**, A. Sengupta "Exploration of Optimal Crypto-Chain Signature Embedded Secure JPEG-CODEC Hardware IP during High Level Synthesis", ***Elsevier Journal***

on Microprocessors and Microsystems, Volume 102, October 2023, 104916, doi: 10.1016/j.micpro.2023.104916, (**Impact factor:** 1.9).

12. A. Sengupta, **A. Anshul**, R. Chaurasia "Exploration of Optimal Functional Trojan-Resistant Hardware Intellectual Property (IP) Core Designs during High Level Synthesis", ***Elsevier Journal on Microprocessors and Microsystems***, Volume 103, November 2023, 104973, doi.org/10.1016/j.micpro.2023.104973, (**Impact Factor**: 1.9).

13. M. Rathor, **A. Anshul**, K Bharath, R. Chaurasia, A. Sengupta, Quadruple phase watermarking during high level synthesis for securing reusable hardware intellectual property cores, ***Elsevier Computers and Electrical Engineering***, Volume 105, 2023, 108476, doi.org/10.1016/j.compeleceng.2022.108476, (**Impact factor**: 4.0).

BOOK CHAPTERS (10):

14. A. Sengupta, **A. Anshul** "HLS Based Fingerprinting"**,** ***IET Book "High-Level Synthesis based Methodologies for Hardware Security, Trust and IP Protection"***, 2024, Chapter DOI**:** 10.1049/PBCS084E_ch7.

15. A. Sengupta, **A. Anshul** "High Level Synthesis based Watermarking using Crypto-Chain Signature Framework"**,** ***IET Book "High-Level Synthesis based Methodologies for Hardware Security, Trust and IP Protection"***, 2024, Chapter DOI**:** 10.1049/PBCS084E_ch6.

16. A. Sengupta, **A. Anshul** "High Level Synthesis based Watermarking using Multi-modal Biometric"**,** ***IET Book "High-Level Synthesis based Methodologies for Hardware Security, Trust and IP Protection"***, 2024, Chapter DOI**:** 10.1049/PBCS084E_ch5.

17. A. Sengupta, **A. Anshul** "HLS based Mathematical Watermarks for Hardware Security and Trust"**,** ***IET Book "High-Level Synthesis based Methodologies for Hardware Security, Trust and IP Protection"***, 2024, Chapter DOI**:** 10.1049/PBCS084E_ch4.

18. A. Sengupta, **A. Anshul** "High-Level Synthesis based Watermarking using Protein Molecular Biometric with Facial Biometric Encryption"**,** ***IET Book "High-Level Synthesis based Methodologies for Hardware Security, Trust and IP Protection"***, 2024, Chapter DOI**:** 10.1049/PBCS084E_ch2.

19. A. Sengupta, **A. Anshul** "Palmprint Biometrics Vs. Fingerprint Biometrics Vs. Digital Signature using Encrypted Hash: Qualitative and Quantitative Comparison for Security of DSP coprocessors"**,** ***IET Book "Physical Biometrics for Hardware Security of DSP and Machine Learning Coprocessors"***, 2023, Chapter DOI**:** 10.1049/PBCS080E_ch6.

20. A. Sengupta, **A. Anshul** "Secured Design Flow using Palmprint Biometrics, Steganography and PSO for DSP coprocessors"**,** ***IET Book "Physical Biometrics for Hardware Security of DSP and Machine Learning Coprocessors"***, 2023, Chapter DOI: 10.1049/PBCS080E_ch7.

21. A. Sengupta, **A. Anshul** "Taxonomy of Hardware Security Methodologies: IP Core Protection and Obfuscation"**,** ***IET Book "Physical Biometrics for Hardware Security of DSP and Machine Learning Coprocessors"***, 2023, Chapter DOI: 10.1049/PBCS080E_ch9.

22. **A Anshul**, R. Chaurasia, A. Sengupta "Securing Hardware Coprocessors against Piracy using Biometrics for Secured IoT systems"**,** ***IET Book "Artificial Intelligence for Biometrics and Cybersecurity***", 2023, Chapter DOI: 10.1049/PBSE020E_ch8.

23. **A. Anshul**, A. Sengupta "Role of Consumer Technology and Connected Electronic Devices on SCM: A Discussion on its Usages, Impact, and Challenges"**,** ***UTHM Book "Evolution of Information, Communication and Computing System",*** 4(1), 1-11, 2023.

PEER- REVIEWED CONFERENCE PUBLICATIONS (9):

24. A. Sengupta, R. Chaurasia and **A. Anshul**, "Hardware Security of Digital Image Filter IP Cores against Piracy using IP Seller's Fingerprint Encrypted Amino Acid Biometric Sample," ***Proceedings of 8th IEEE Asian Hardware Oriented Security and Trust Symposium (AsianHOST)***, Tianjin, China, 2023, pp. 1-6, doi: 10.1109/AsianHOST59942.2023.10409476.

25. A. Sengupta, **A. Anshul**, S. Thakur and C. Kothari, "Fusing IP vendor Palmprint Biometric with Encoded Hash for Hardware IP Core Protection of Image Processing Filters," ***Proceedings of 35th IEEE International Conference on Microelectronics (ICM)***, Abu Dhabi, United Arab Emirates, 2023, pp. 218-221, doi: 10.1109/ICM60448.2023.10378937.

26. A. Sengupta, **A. Anshul**, C. Kothari and S. Thakur, "Secured and Optimized Hardware Accelerators using Key-Controlled Encoded Hash Slices and Firefly Algorithm based Exploration," ***Proceedings of 35th IEEE International Conference on Microelectronics (ICM)***, Abu Dhabi, United Arab Emirates, 2023, pp. 149-152, doi: 10.1109/ICM60448.2023.10378911 .

27. A. Sengupta and **A. Anshul**, "Key-Driven Multi-Layered Structural Obfuscation of IP cores using Reconfigurable Obfuscator based Network Challenge and Switch Control Logic," ***Proceedings of 9th IEEE International Symposium on Smart Electronic Systems (iSES)***, Ahmedabad, India, 2023, pp. 141-146, doi: 10.1109/iSES58672.2023.00038.

28. **A. Anshul** and A. Sengupta, "Low-Cost Hardware Security of Laplace Edge Detection and Embossment Filter Using HLS Based Encryption and PSO," ***Proceedings of 9th IEEE International Symposium on Smart Electronic Systems (iSES)***, Ahmedabad, India, 2023, pp. 135-140, doi: 10.1109/iSES58672.2023.00037.

29. **A. Anshul**, K. Bharath and A. Sengupta, "Designing Low Cost Secured DSP Core using Steganography and PSO for CE systems," ***Proceedings of 8th IEEE International Symposium on Smart Electronic Systems (iSES)***, Warangal, India, 2022, pp. 95-100, doi: 10.1109/iSES54909.2022.00030.

30. **A. Anshul** and A. Sengupta, "IP Core Protection of Image Processing Filters with Multi-Level Encryption and Covert Steganographic Security Constraints," ***Proceedings of 8th IEEE International Symposium on Smart Electronic Systems (iSES)***, Warangal, India, 2022, pp. 83-88, doi: 10.1109/iSES54909.2022.00028.

31. A. Sengupta, V. Chourasia, **A. Anshul** "HLS Scheduling Driven Encoded Watermarking for Secure Convolutional Layer IP Design in CNN", ***Proceedings of 11th IEEE International Conference on Consumer Electronics (ICCE-TW)***, Taichung, Taiwan, 2024, pp. 587-588, doi: 10.1109/ICCE-Taiwan62264.2024.10674266.

32. A. Sengupta, V. Chourasia, **A. Anshul**, N. Kumar "Robust Watermarking of Loop Unrolled Convolution Layer IP Design for CNN using 4-variable Encoded Register Allocation", ***Proceedings of 11th IEEE International Conference on Consumer Electronics (ICCE-TW)***, Taichung, Taiwan, 2024, pp. 589-590, doi: 10.1109/ICCE-Taiwan62264.2024.10674385.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

$S_i/Y_i/P_i$	Resource architecture for i^{th} particle position
$S_i^+/Y_i^+/P_i^+$	New updated resource architecture of i^{th} particle
$S_{GB}/Y_{GB}/P_{GB}$	Global best particle position
$S_{LBi}/Y_{LBi}/P_{LBi}$	Local best particle position for i^{th} particle
$S_{max}/Y_{max}/P_{max}$	Maximum resource architecture for a particular benchmark application
A_{ix}	Velocity of i^{th} particle in x^{th} dimension
A_{ix}^+	New updated velocity of i^{th} particle in x^{th} dimension
A_{ix}^{max}	Maximum velocity of i^{th} particle in x^{th} dimension
I	Maximum iteration count
C_f^{LBi}	Local best fitness cost of i^{th} particle
n_m	Total control step required by multiplier in scheduled data flow graph of DSP application
d_m	Delay of one multiplier
A_{Con}	IP vendor specified area constraints
T	Termination criterion
X	Total dimensions (hardware types)
$SDFG^{TMR}$	Trojan resistant scheduled dataflow graph
U_{OG}	Original unit in the $SDFG^{TMR}$
ω	Inertia weight
b_1 *and* b_2	Acceleration coefficients used in PSO for making a balance between social and cognitive factor
r_1 *and* r_2	Random number used for providing stochasticity
A_{TMR}	Total area of the TMR design
T_{TMR}	Total execution time of TMR design
A_{MAX}	Area of the design with maximum resources
T_{MAX}	Maximum execution time of design with minimum resources
W_1 *and* W_2	Weights assigned to area and time in design cost function
n	Total number of particles in the population
C_f^i	Fitness cost of i^{th} particle
n_a	Total control step required by adder in scheduled data flow graph of DSP application
d_a	Delay of one adder
T_{Con}	IP vendor specified latency constraints

V_i	Velocity of i^{th} particle
PSO-DSE	Particle swarm optimization driven design space exploration
TMR	Triple modular redundancy
U_{DP}	Duplicate unit (copy) of original unit in $SDFG^{TMR}$
U_{TP}	Triplicate unit (copy) of original unit in $SDFG^{TMR}$
PC/C_i	Probability of coincidence
$TT/T_i/Z_t/T_O$	Tamper tolerance
E	Entropy
DC	Design cost
P_T	Total duration of voice pitch
p_e and p_s	End time and start time, respectively, on the voice pitch variation curve
Δp and Δi	Step size
I_T	Total duration of voice intensity
I_e and I_s	End time and start time of voice intensity, respectively
$s(n)$	Input to the filtering unit
H	Linear combination matrix of the representative function
VD_i	i^{th} Vendor

ACRONYMS

HLS	High Level Synthesis
IP	Intellectual Property
DSP	Digital Signal Processor
CE	Consumer Electronics
IC	Integrated Circuits
ASIC	Application Specific Integrated Circuit
SoC	System on Chip
RTL	Register Transfer Level
VLSI	Very Large Scale Integration
GDS	Graphic Database System
DFG	Data Flow Graph
CDFG	Control Data Flow Graph
SDFG	Scheduled Data Flow Graph
CS	Control Step
CIG	Colored Interval Graph
3PIP	3rd party Intellectual Property
ASIC	Application Specific Integrated Circuit
PSO	Particle Swarm Optimization
DSE	Design Space Exploration
FU	Functional Unit
FFA	Firefly Algorithm
SHA	Secure Hash Algorithm
DCT	Discrete Cosine Transform
IDCT	Inverse Discrete Cosine Transform
DWT	Discrete Wavelet Transform
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
TMR	Triple Modular Redundant

DMR	Double Modular Redundant
ACO	Ant Colony Optimization
BFO	Bacterial Foraging Algorithm
GA	Genetic Algorithm
AI	Artificial Intelligence
ML	Machine Learning
FPGA	Field programmable gate array
THT	Tree Height Transformation
VHDL	Very High Speed Integrated Circuit Hardware Description Language
RE	Reverse Engineering
GLRT	Generalized Likelihood Ratio Test
ECG	Electrocardiogram
MRI	Magnetic Resonance Imaging
CT	Computed Tomography
BF	Blur Filter
SF	Sharpening Filter
LED	Laplace Edge Detection
VE/HE	Vertical/Horizontal Embossment
WFB	Wavelet Filter Bank

Chapter 1

Introduction

1.1. Hardware Systems: ASIC/IP Core

We are fortunate to be part of a generation that benefits from the advancements in smart technology, all made possible by the relentless efforts of scientists and researchers. In this modern era, electronic/computing systems have been instrumental in realizing the vision of making technology smarter, more efficient, and accessible to everyone. Today, there is an increasing demand for systems and devices that offer rapid and cost-effective processing, whether for applications or hardware. As a result, various consumer electronics and computing devices, such as smartphones, smartwatches, tablets, digital cameras, computers, and audio headsets, have become integral to our daily lives, fulfilling both our needs and desires. Apart from consumer electronics and multimedia systems, these computing devices are also used in healthcare, military operations, and other real-world scenarios. Electronics and computing devices in smart healthcare enable remote monitoring, precise diagnostics, and personalized treatments, while in advanced military operations, they enhance communication, surveillance, and strategic decision-making [1] - [4].

Moreover, the need for application-specific computing is rapidly increasing in the current technological landscape. Unlike general-purpose systems that handle a wide range of functions, application-specific computing systems offer tailored performance, optimized power consumption, and enhanced efficiency for specific tasks. In the modern era, these systems provide faster processing, reduced energy use, and lower costs by focusing on specialized applications like image processing or data encryption, thereby improving overall performance. As they perform several complex data and computation-intensive tasks like image processing, audio-video processing, and more, they are designed as application-specific integrated circuits (ASIC), also known as hardware accelerators. The base of these devices is a system-on-chip (SoC), which integrates various components, including functional blocks, memory units, memory controllers, and peripherals. Rather than designing a SoC

entirely from the ground up, manufacturer (SoC integrator/designer) often opt to acquire various modules or cores from third-party intellectual property (IP) vendors/sellers. This approach, known as the core-based design paradigm, has revolutionized the way electronic systems are developed, enabling faster production cycles and reducing costs while maintaining high performance and functionality. Generally, the complete design process, starting from the procurement of different IP cores to assembly and the release of the final product, involves multiple entities, locations, and design houses. This is because some places can afford lower technical costs while some cheap labor, besides time to market factor [5] - [11].

As discussed above, ASICs/hardware accelerators play a critical role in enhancing performance and efficiency for data-intensive tasks by speeding up the underlying processes. This is achieved by offloading specific computing tasks to specialized hardware components, known as hardware accelerator or IP core. An IP core is a reusable block of Boolean logic/functions, register transfer level (RTL) design, or gate-level design organization representing the designer's intellectual property. Some examples of hardware accelerators/IP cores include cryptographic IP cores for performing specialized cryptographic operations, image processing or digital signal processing (DSP) IP cores used for performing image processing applications (such as blurring, sharpening, etc.), compression-decompression of images, biometric recognition tasks like facial, fingerprint, and palmprint detection. Similarly, artificial intelligence (AI) cores/machine learning (ML) cores are responsible for the execution of complex AI/ML applications. In consumer electronics and computing systems, data-intensive applications such as audio-video processing and image compression-decompression are effectively handled by these IP cores, offering high efficiency at reduced design costs. Fig. 1.1 highlights the examples and applications of different data and computation-intensive hardware IPs used in several consumer electronics and multimedia systems. IP cores execute complex algorithms like fast Fourier transform (FFT), finite impulse response (FIR) filtering, discrete cosine transformation (DCT), image compression and decompression (JPEG-CODEC), etc., which are fundamental in multimedia, machine learning, and digital signal processing applications. Due to the

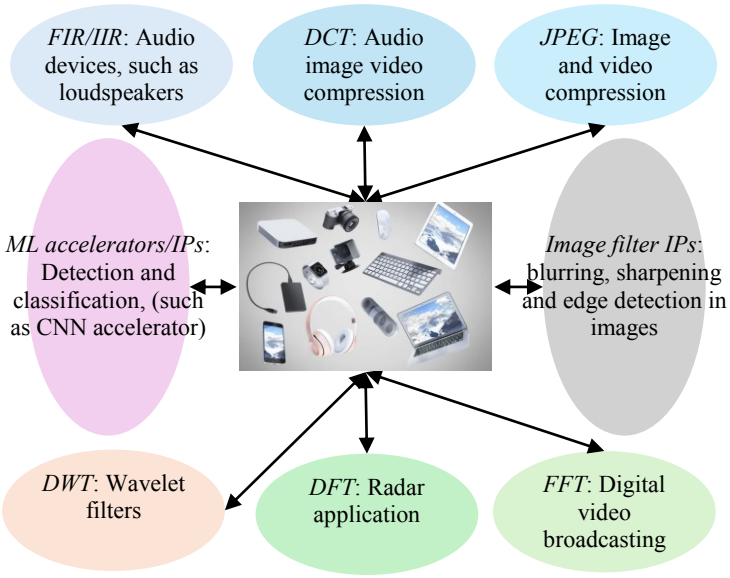


Fig. 1.1 Examples and applications of different data and computation intensive hardware IPs used in consumer electronics and multimedia systems

complexities of design, cost, and time-to-market pressure, these application frameworks are developed as dedicated reusable IP cores. This approach reduces costs and accelerates design turnaround times. Consequently, SoC designers incorporate reusable IP cores from various vendors, which are mass-produced, thoroughly tested, and verified by companies around the globe, ensuring a robust and diverse IP supply chain [12] - [14].

Understanding the design and development process of such complex systems is crucial for researchers and users. The design cycle of these systems consists of multiple phases and involves various entities. These phases can be categorized based on design complexity, cost, and flexibility, making it essential to comprehend each stage thoroughly. Additionally, the entities involved in the design process—such as third-party IP vendors, system integrators (SoC), and foundries (fabrication houses)—play distinct roles in the integrated circuit (IC) design chain. Their involvement helps lower design costs, reduces complexity, and shortens development time. However, as the design passes through various entities in the global design supply chain process, it also necessitates robust security measures to protect designs from potential threats and ensure their safe use by end consumers. The participation of diverse entities, including offshore design houses, raises concerns about trustworthiness. Unreliable entities could engage in malicious activities, such as IP piracy or fraudulent claims of IP ownership. There is also the risk of

covert malicious logic being implanted by a rouge entity in a counterfeited product. Thus, protecting the IP rights of IP vendors/sellers is vital, along with ensuring the safety of end-consumers. Given the significant role of multimedia, DSP, and ML IP cores in consumer electronics, IoT devices, smart healthcare, and mission-critical tasks, their security cannot be compromised. Incorporating a pirated IP version into the SoCs of these applications could jeopardize user safety and system integrity [15] - [26].

This chapter provides an overview of the foundational elements upon which the proposed hardware security techniques are developed. The second section outlines the different abstraction levels of the ASIC/IP core design process. The third section delves into the high-level synthesis (HLS) process, highlighting its significance in creating low-cost, secure reusable hardware IPs and some examples of data intensive DSP applications. The fourth section examines the various hardware threats and attacks in the ASIC/IP core design process. Finally, the fifth section details the overall structure and organization of the thesis.

1.2. Abstraction Levels in ASIC Design Process

To effectively manage the complexity of designing a hardware IP core, it is essential to begin at a higher abstraction level within the integrated circuit (IC) design process. This approach is advantageous because higher abstraction levels simplify the design process, provide greater flexibility, and make it easier to integrate cost-efficient (low-cost) architectures and robust security mechanisms compared to lower-level design abstractions. The design abstraction hierarchy generally consists of the following levels: (i) system level, (ii) algorithmic/behavioral level, (iii) register transfer level (RTL), (iv) logic/gate/netlist level, and (v) physical/layout/transistor level. Fig. 1.2 illustrates the different abstraction level used in VLSI/digital ICs design process.

The highest level in this hierarchy is the system level, where the design or application is characterized based on input, output, and transfer functions. At this level, key parameters such as functionality, size, speed, and power requirements are also taken into account. The next level is the algorithmic or behavioral level, where the design is represented in terms of its behavior. At

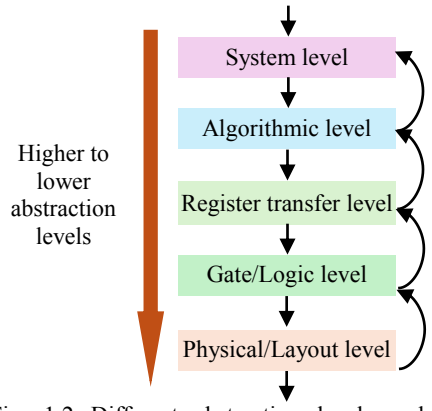


Fig. 1.2 Different abstraction level used in VLSI/digital ICs design process

this stage, designers often use control data flow graphs (CDFG)/data flow graphs (DFG) to depict the system's behavior. The input transfer/mathematical function is converted into these graphical forms to serve as an intermediate representation, enabling further transformation into subsequent design levels.

The algorithmic representation of the design is then converted into the register transfer level (RTL) using HLS tool. RTL describes the interconnection between different units such as arithmetic and logic unit (ALU), control unit, storage hardware. This RTL design, referred to as "*soft IP*," is usually available in the form of synthesizable code, such as schematic design files *(.bdf)* or hardware description language files *(.vhd/.vhdl)*. One significant benefit of designing at higher abstraction levels is that integrating security mechanism is less complex and ensures robust security across all subsequent lower-level designs. The "*soft IP*" core provides a flexible foundation for chip designers, allowing them to modify and optimize design parameters to meet specific requirements. This flexibility is crucial for achieving a balance between performance, power efficiency, and security, thereby making the design process both efficient and adaptable to various technological needs.

The gate level, also known as the netlist level, represents the next stage of design abstraction in the IP core design process. At this level, the RTL design is converted into a gate-level design using logic or RTL synthesis. The gate level defines the design in terms of the interconnections between various logic cells and represents the output of the synthesis process at the logic level. The resulting design, known as the gate-level netlist, is referred to as a "*firm IP core*." Unlike the *soft IP core*, a *firm IP core* is dependent on specific technology and is less flexible for modification. Both RTL and gate-level

netlist designs support further post-synthesis processes, such as placement, routing, and deployment on reconfigurable platforms like FPGAs.

Following the gate level is the layout/physical/transistor level, which represents an even lower level of abstraction. At this stage, the gate-level design is transformed into a layout-level design through a process called layout synthesis. The resulting design, known as a "*hard IP core*," is typically presented in a fixed layout format, such as the graphic data system (GDS) or layout editor format (LEF). Unlike *soft IP cores*, *hard IP cores* are not modifiable by chip designers or system integrators. One significant drawback of hard IP designs is their lack of portability; they cannot be used in different foundries for which they were not originally designed. This limitation arises because the layout design is specific to the foundry's process technology and design rules, preventing its reuse across different manufacturing environments. Given the greater flexibility and portability of soft IP cores, they are often preferred over hard IP cores. Soft IP cores can be modified to suit specific functional requirements and can be reused across various platforms and foundries. However, this flexibility comes with a trade-off: soft IP cores are more vulnerable to intellectual property (IP) protection risks because they can be easily modified by system integrators. In contrast, hard IP cores, being unmodifiable, are more secure against such risks. Therefore, IP cores are generally designed and marketed in one of three forms: (a) *soft IP cores*, (b) *firm IP cores*, or (c) *hard IP cores*, depending on the level of flexibility, modifiability, and security required.

In addition to their classification based on abstraction levels, IP cores are also categorized into two types based on their design sizes and computational capabilities: micro-IPs and macro-IPs. Micro-IPs are smaller logic blocks, such as individual logic gates, combinational circuits, and sequential circuits (like registers and memory). In contrast, macro-IPs represent larger, more complex logic designs and include components such as central processing units (CPUs), digital signal processors (DSPs), and application-specific cores. Examples of macro-IPs include cores designed for specific tasks like image processing is joint photographer expert group (JPEG-CODEC), video processing is moving picture expert group (MPEG), and digital filtering (finite

impulse response (FIR) and infinite impulse response (IIR) filters). These macro-IPs are particularly useful for applications that require intensive computations, such as audio processing, image compression and decompression, digital data filtering, etc. The choice between micro-IPs and macro-IPs largely depends on the specific requirements of the application, including the level of computational complexity and the design size constraints. Overall, the categorization and abstraction levels provide a framework for selecting the appropriate IP core type, balancing flexibility, security, performance, and application needs.

1.3. Introduction to HLS

In the IC design chain process/cycle, synthesis is a critical process that involves transforming a design from one form to another to facilitate verification and analysis. Given the increasing complexity, design cost, and time constraints, it is vital for designers to start at a less complex and more flexible level. The choice of synthesis level depends on the required information for analysis and representation. Synthesis processes are generally categorized into three levels: (a) high-level synthesis (HLS), (b) logic synthesis, and (c) physical synthesis, corresponding from the highest to the lowest level of abstraction. Among these, HLS offers the most flexibility and the least complexity, making it a preferred starting point for many designers [83], [87].

HLS converts a behavioral description of the design—typically a mathematical equation representing the input-output relationship of a data-intensive algorithm—into RTL design. This conversion involves several phases: transformation, scheduling, binding, and the final datapath and controller synthesis. The overview of HLS design flow is highlighted in Fig. 1.3. Transfer/mathematical function of input application, resource constraints, and module library (containing details of area, power, latency corresponding to used functional units, such as adders and multipliers) are the primary inputs of HLS process. HLS begins with the transformation phase. In the transformation phase, the mathematical or behavioral description of the design is represented as a control data flow graph (CDFG). The CDFG is a structural model that captures the input-output relationships and data flow of the design.

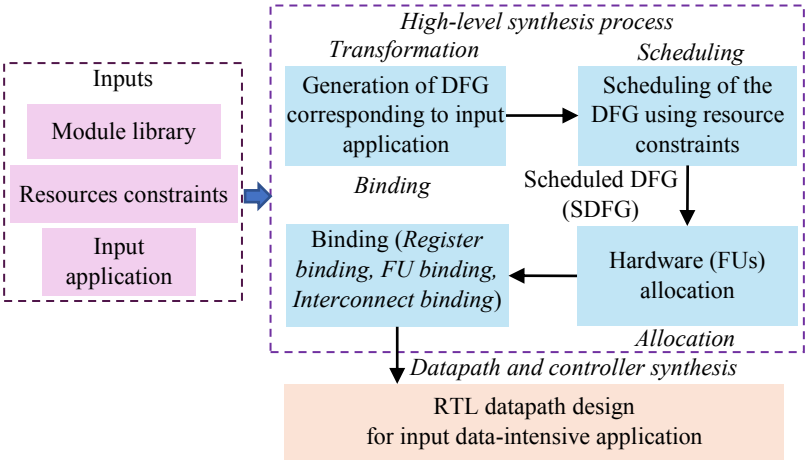


Fig. 1.3 Overview of HLS design flow

For example, a CDFG for a FIR digital filter is depicted in Fig. 4 (a), where *X[n], X[n-1], X[n-2], X[n-3]* denote primary inputs, *Z[0], Z[1], Z[2], Z[3]* denote input coefficients and '+' and '*' denote corresponding multiplications and additions operations.

The scheduling phase, one of the most critical stages in HLS, transforms the DFG into a scheduled version based on input resource constraints (number of adders and multipliers) and scheduling algorithms. For instance, the CDFG/DFG of an FIR filter (shown in Fig. 1.4 (a)) can be scheduled with different resource constraints, such as one multiplier and one adder (shown in Fig. 1.4 (b)) or two multipliers and one adder (shown in Fig. 1.4 (c)). The scheduling algorithm used is LIST scheduling, which prioritizes operations that do not depend on others for execution and schedules them to maximize resource utilization (based on data dependency and input resource constraints). Conflicts are resolved by giving priority to operations higher on the list. Depending on the chosen resource constraints, the scheduled design may have different execution times. For example, a design with one multiplier and one adder may take more control steps (six CS, C0-C5) than one with two multipliers and one adder, but it uses fewer resources. However, using more resources can show lesser control steps (five CS, C0-C4), while resulting in a larger design area. Therefore, it is crucial for designers to select resource constraints that balance minimizing both design latency and area. This selection process can also be automated through the design space exploration (DSE) capabilities of the HLS framework.

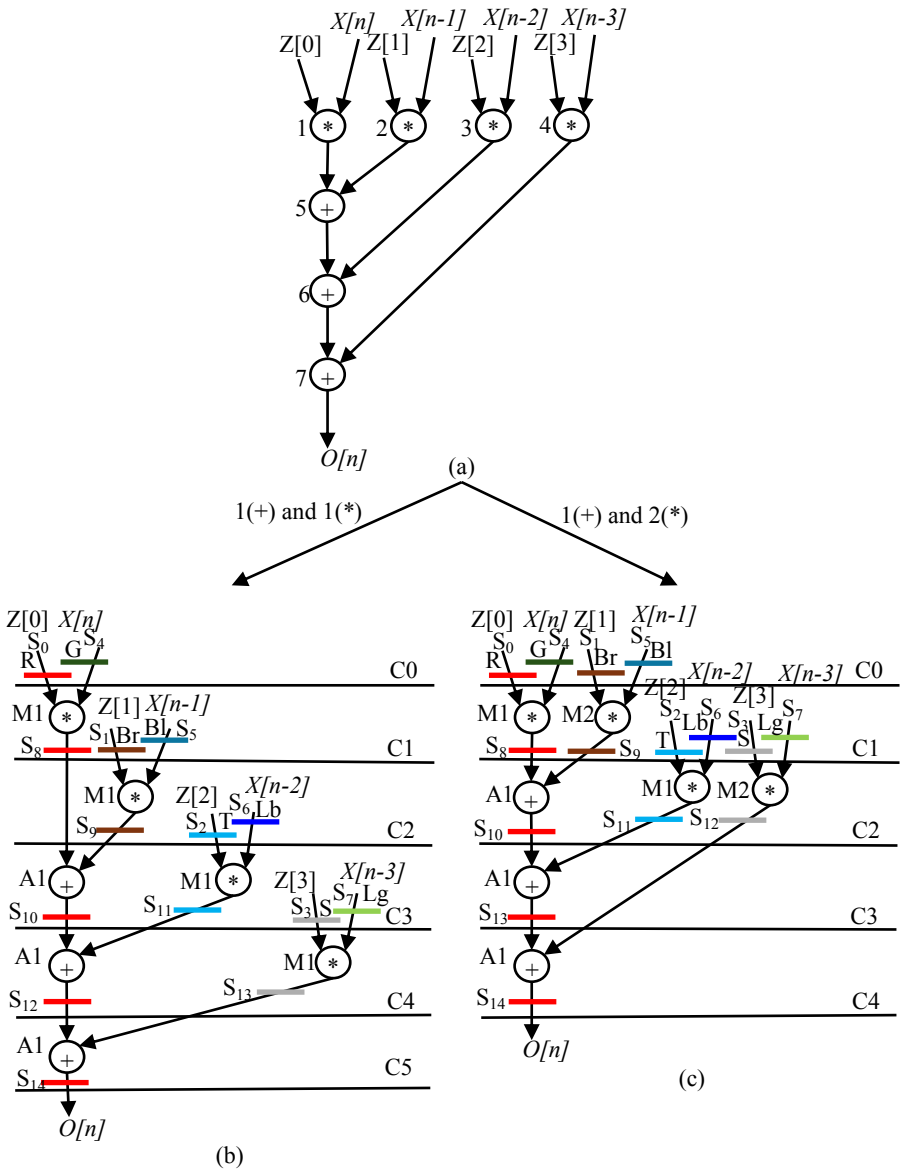


Fig. 1.4 (a). Control data flow graph (CDFG) of FIR filter, (b) Scheduled data flow graph (SDFG) of FIR filter scheduled with one multiplier and one adder, and (c) Scheduled data flow graph (SDFG) of FIR filter scheduled with two multipliers and one adder

The next phase, hardware allocation, involves assigning hardware resources (such as adders and multipliers) to the operations and storage registers for the design's storage variables (used for storing input, intermediate and output values). This allocation considers constraints such as design latency, design area, and power consumption. More hardware resources reduce latency/delay due to parallel execution but increase the design area. Conversely, minimal hardware resources reduce area but may increase latency due to serial execution. Subsequently, the binding phase follows, where specific operations are assigned to particular instances of functional units, and storage variables are mapped to registers. For example, in an FIR filter design, multipliers and adders are allocated to specific operations, and storage variables are assigned

to specific registers, which are depicted using different colors. Figures 4 (b) and (c) show scheduled data flow graph (SDFG) of FIR filter scheduled with one multiplier and one adder and SDFG of FIR filter scheduled with two multipliers and one adder, where $S_0 - S_{14}$ are the storage variables allocated to eight different resisters (*R-Lg*).

After the scheduling, allocation, and binding phases, the datapath and controller synthesis phase is performed. This phase constructs the RTL datapath using the allocated functional units, registers, latches, and other components such as multiplexers and demultiplexers, as determined during binding. The controller is designed to generate control signals for different units of the datapath based on the scheduled operations. Thus, HLS transforms the behavioral description of a data-intensive application into an RTL design, also known as a *soft IP core*. Once the RTL design is obtained, it can be further transformed into a gate-level or netlist design through logic synthesis. The gate-level design represents a more detailed and complex circuit than the RTL design. Subsequently, this gate-level design is converted into a layout design using physical synthesis, which is then sent to foundries for chip fabrication. Designing an IP from a lower level of abstraction is generally not preferred due to the higher complexity involved.

Security is another critical aspect of the synthesis process, particularly when designing IP cores. Various security mechanisms, such as hardware watermarking, steganographic constraints, and digital signatures, can be integrated into the design during the HLS phase. These approaches embed covert watermarking/security constraints into the design to detect unauthorized use or piracy of IP cores, ensuring that only legitimate IP versions are integrated into SoC systems. To further enhance security, the design can undergo high-level transformations, a process known as structural obfuscation. Structural obfuscation alters the design's structure without affecting its functionality, making it difficult for adversaries to interpret the design’s function or interconnectivity. Common high-level transformations include loop unrolling, tree height transformation, and redundant operation elimination [92]. These transformations prevent adversaries from reverse-engineering the design by obscuring its functionality and architecture [20]. Implementing

security mechanisms at lower abstraction levels is challenging due to the complexity and the unavailability of IPs at these levels, such as gate-level netlists. In contrast, many DSP and multimedia applications are available in their algorithmic descriptions and can be automatically synthesized into RTL designs using commercial or non-commercial tools. This capability allows designers to integrate security mechanisms with the computer-aided design (CAD) tools of HLS to create secure IP versions for data-intensive applications.

In summary, the synthesis process in IC design is an essential step that involves converting designs from higher to lower levels of abstraction, starting with HLS, which provides greater flexibility and ease of integration for security mechanisms. By beginning at a higher abstraction level, designers can manage complexity more effectively, optimize design parameters, and ensure robust security, ultimately producing secure, efficient, and adaptable IP cores for various applications.

Leveraging high-level synthesis for low-cost IP core design [32], [91]: HLS plays a vital role in achieving low-cost IP core designs by allowing the exploration of various resource constraints, which impact design latency and area. Optimal resource selection during scheduling is crucial for designers and integrating design space exploration (DSE) within HLS enables the identification of cost-effective architectural solutions that meet area and latency requirements. When secret hardware security constraints are embedded, they can increase design cost, making the exploration of low-cost resource options even more important. By incorporating security measures at the HLS stage, security is inherently propagated to lower design levels, ensuring protection across firm and hard IPs as synthesis progresses.

Examples and importance of DSP applications: Further, DSP co-processors utilize various algorithms to perform specific tasks related to digital signal processing. Commonly employed DSP algorithms include the Haar wavelet transform (HWT), fast Fourier transform (FFT), discrete cosine transform (DCT), discrete wavelet transform (DWT), inverse discrete cosine transform (IDCT), and discrete Fourier transform (DFT). Each of these algorithms serves distinct purposes in processing signals and images. The DCT is primarily used

to convert an image from its spatial domain to the frequency domain and forms the core algorithm for image compression and decompression in JPEG-CODEC co-processors. DFT and FFT, on the other hand, transform a discrete signal from the time domain to the frequency domain, which is essential for signal analysis in many applications. The HWT facilitates the transformation of a signal's waveform from the time domain to the time-frequency domain, making it effective for both lossy and lossless compression of signals and images. DWT plays a crucial role in denoising real signals by decomposing them into finer frequency and coarser time resolutions across different sub-bands. DWT serves as the foundational algorithm for image compression in JPEG2000 standards. Additionally, digital filters like FIR and IIR filters are vital in modern electronics, finding applications in telecommunications, speech processing, and attenuation removal for specific frequency bands. Fig. 1.1 highlights the application of several DSP applications in the real-world scenario.

Next, machine learning IP cores are also integrated into DSP co-processors to handle tasks related to AI and machine learning algorithms, such as pattern recognition, classification, and predictive analytics. Meanwhile, multimedia processors rely on multimedia algorithms, such as those used in the JPEG compression-decompression and MPEG standards. The JPEG algorithm is widely employed for image compression. It operates by first converting an input image from the spatial domain to the frequency domain and then applies quantization, which involves discarding less important frequency components, resulting in a compressed image. This method is extensively used in fields like medical imaging and digital photography, where storage efficiency and transmission speed are crucial.

In summary, DSP co-processors leverage a variety of signal processing algorithms and digital filters to manage tasks that range from image and signal compression to machine learning, contributing to a broad spectrum of applications in telecommunications, multimedia processing, and beyond.

Next, to generate an application-specific processor for data-intensive tasks, the synthesis process begins with the algorithmic or behavioral description of the application as input [12], [91]. This description can take various forms, such

as C/C++ code, a transfer function, or a mathematical equation representing the input-output relationship. For instance, the algorithmic description of a FIR filter can be expressed as a mathematical function [81]:

$$O[n] = \sum_{k=0}^{N} Z[k] * X[n-k] \quad (1.1)$$

Where, N represents the order of the FIR filter. Further, the mathematical equation based on the order of FIR filter, can be represented as follows:

$$O[n] = Z[0] * X[n] + Z[1] * X[n-1] + Z[2] * X[n-2] + \cdots + Z[N] * X[n-N] \quad (1.2)$$

Where, $X[n]$ to $O[n]$ denote the current input and output, and X[n-1], X[n-2] denote the previous input values and, $Z[0], Z[1], \ldots, Z[N]$ denote the FIR's input coefficients. This transfer function or mathematical equation is converted into its corresponding CDFG.

1.4. Hardware Threats and Attacks in the ASIC/IP Core Design Flow

As discussed above, the semiconductor design and manufacturing process is a complex global network involving various offshore entities, such as third-party intellectual property (3PIP) vendors, system integrators, and foundry/fabrication houses. This network is designed to accelerate the design process, reduce cost, and shorten the time to market. Within this framework, IP cores are often provided by multiple IP vendors, based on specific design requirements. These IPs are then supplied to a SoC integrator for integration into an SoC design or sent directly to foundry houses for fabrication as standalone integrated circuits (ICs). Once integrated at the SoC integrator, the design is passed to the foundry houses for fabrication. This sequential, unidirectional flow—from IP vendor to SoC integrator and then to foundry house—illustrates the asymmetric nature of the business model. Multiple IP vendors may provide different IP designs, and multiple foundries may handle the fabrication, which introduces various vulnerabilities to hardware security threats within the IC design supply chain process [15]-[26], [27]-[36], [53], [54].

The involvement of multiple entities across different stages of the semiconductor supply chain exposes it to significant risks, including the infiltration of counterfeit components. Historical cases underscore the severity of these vulnerabilities. For instance, in 2007 and 2008, the United States Customs and Border Protection (CBP), in collaboration with European Union Customs, conducted joint operations to enforce intellectual property rights. These operations resulted in the seizure of hundreds of thousands of counterfeit ICs and computer network components. However, these seizures likely represented only a fraction of the counterfeit products that entered the market during that period.

In 2010, a notable incident involved VisionTech, a company whose owner and administrative manager were charged with trafficking counterfeit semiconductors. They were responsible for importing thousands of shipments of counterfeit components into the United States, primarily targeting the U.S. Navy and defense contractors. This conspiracy, which spanned nearly five years, highlighted the potential for rogue brokers to severely compromise national security and jeopardize countless lives. VisionTech's actions were estimated to have caused substantial damage to 21 semiconductor companies by supplying counterfeit components, demonstrating the far-reaching impacts of counterfeit products within the supply chain.

The financial implications of counterfeit components are staggering. In 2012, a report by the market research firm iHS iSuppli estimated that counterfeit products/elements resulted in multibillion-dollar losses to the global electronics supply chain. In 2016, the European Union (EU) and Dutch customs conducted an operation targeting semiconductor imports from Hong Kong and China, seizing over one million counterfeit devices in just a few weeks. The 2018 report by the world semiconductor council (WSC) further emphasized the critical nature of the issue, noting that counterfeit components significantly undermine both security and economic stability. Pirated parts not only compromise the reliability of customer applications but also cost semiconductor companies billions of dollars annually in efforts to ensure the authenticity and reliability of their products. Despite the efforts to combat counterfeiting, accurately assessing the full impact of counterfeit

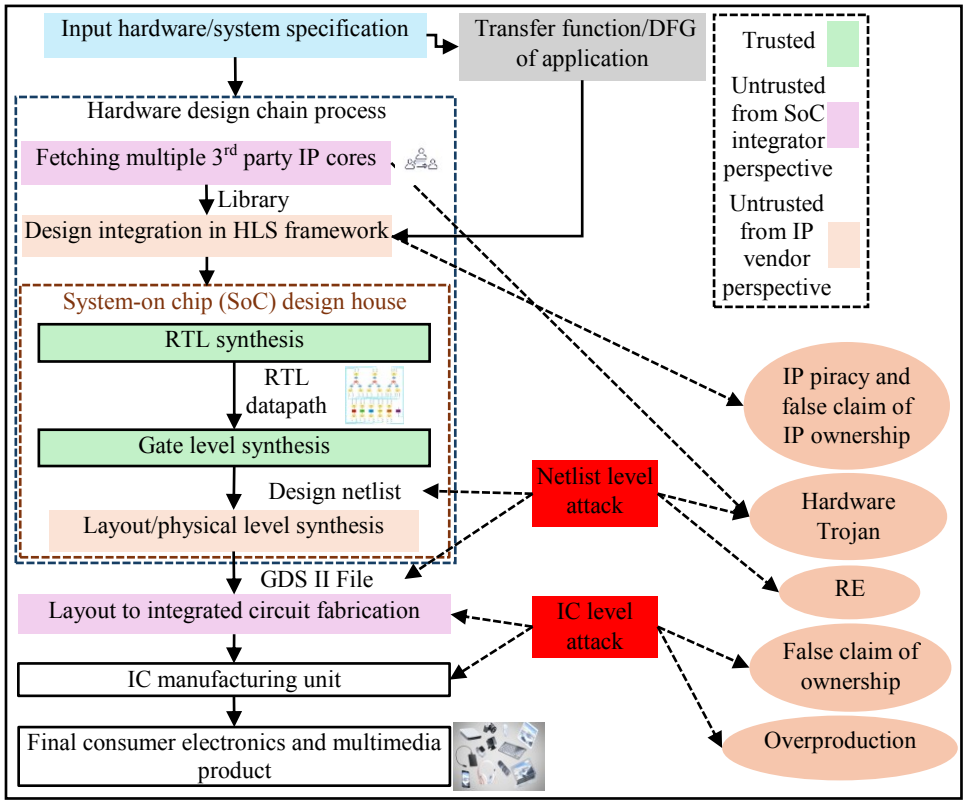


Fig. 1.5 Different hardware threats and attacks in the hardware IC design flow process

semiconductors remains challenging. Nonetheless, the available data and reports make it clear that counterfeit components pose a serious threat to the integrity of the global IC supply chain. This raises profound concerns about the trustworthiness of the supply chain, as the presence of counterfeit components can lead to severe financial losses, security vulnerabilities, and risks to human safety. The complexity of the semiconductor supply chain, combined with the involvement of multiple global entities, underscores the urgent need for enhanced security measures and stricter enforcement of intellectual property rights to protect against the proliferation of counterfeit components [20], [89], [90]. Fig. 1.5 shows the different hardware threats and attacks in the hardware IC design flow process [20].

The red/pink-colored component depicted in Fig. 1.5 belongs to the untrustworthy sector of the hardware design supply chain process from an SoC integrators perspective, and orange colored component belongs to untrustworthy sector from an IP vendor's perspective. Moreover, the green-colored components signify the trusted sector. The input consists of system specifications, which are the behavioral descriptions of the intended hardware design, as illustrated in Fig. 1.5 These specifications are then progressed

through the hardware design process. This process entails acquiring various intellectual property cores or designs from multiple third-party IP vendors, followed by the integration of these imported cores into a single chip carried out by System-on-Chip (SoC) integrator. The necessity for involving multiple third-party entities was discussed earlier. After integration, a corresponding register transfer level (RTL) file is generated, which subsequently undergoes synthesis to transform it into a gate-level design file, also known as a netlist file. This netlist file is subsequently transmitted to fabrication and manufacturing facilities. As depicted in Fig 1.5, hardware attacks are categorized into three main types: (a) IP piracy and false claim of IP ownership can be potentially performed illegally by an adversary in the SoC integrator house, (b) netlist level attacks that can be potentially performed by an adversary in the foundry (fabrication house), and (c) backdoor hardware Trojan insertion through 3rd party IP (3PIP) cores, (d) Integrated Circuit (IC) level attacks that can be potentially performed by an adversary in the foundry or open market. If an adversary gains access to the design netlist file, they could potentially execute attacks such as reverse engineering, and the insertion of hardware Trojan into the design file. Additionally, throughout the fabrication process and post-fabrication stage, an adversary situated within the fabrication facility might initiate attacks such as overproduction (exceeding the licensed IP limit) and false IC ownership claim. The dashed lines within Fig 5 delineates the various types associated with potential locations of attacks within the design supply chain process. Conversely, the solid lines, depicted in black color, represents the comprehensive hardware design flow from specification to IC manufacturing.

The different types of possible attacks are as follows: (a) IP piracy (counterfeiting and cloning), (b) reverse engineering (RE), (c) hardware Trojan insertion (insertion of malicious logic), (d) fraudulent claim of IP ownership, and (e) overproduction (producing more than the licensing limit). Counterfeited and cloned IPs may contain malicious logic, which can cause severe problems to both end consumers and IP vendors. Moreover, these counterfeited IPs may not be rigorously tested as the genuine ones and may cause various erroneous behavior such as (i) leakage of sensitive information,

(ii) improper functional output, (c) excessive heat dissipation, and (d) loss of esteem for IP vendor and SoC integrator. Further, an adversary can covertly insert malicious logic (hardware Trojan) into the design by carefully inspecting the design through reverse engineering the netlist file. These Trojan remain dormant until triggered, thus escaping the standard detection process. Additionally, IPs/integrated circuits (ICs) can be fraudulently claimed or overproduced over the original licensing limit. Therefore, securing these IP cores against such hardware threats is essential to protect IP vendors. Further, the details on IP piracy, false claim of IP ownership and hardware Trojan attack is discussed below.

1.4.1. IP piracy: counterfeiting and tampering

The elaborate process of design and distribution of hardware IPs is highly vulnerable to a multitude of security risks, largely due to the intricate network of multiple third-party entities and units participating in the global design supply chain process. One significant area of concern revolves around the piracy of hardware IP design once they are transferred from an IP seller to a buyer (typically a SoC integrator). The risk arises when a malicious actor within the SoC integrator's organization endeavors to illicitly/unlawfully replicate/pirate the original design, subsequently marketing it either under the same or a different brand name. This not only complicates the authentication of genuine products but also leads to financial losses for the original IP seller. Furthermore, within a rogue foundry, an attacker might unlawfully pirate the IP without the designer's knowledge or consent. From the opposite perspective, it is also crucial for the SoC integrator to isolate pirated/counterfeited hardware IP designs before integration into the final product to uphold the safety and reliability standards for end-users. A SoC integrator can acquire IP cores either directly from an IP vendor or through a broker who acts as an intermediary between the IP designer and the SoC integrator. However, in some cases, rogue IP suppliers, motivated by national interests or the desire for illicit profit, may introduce counterfeit or pirated components into the design supply chain. These fake components, masquerading as genuine, can negatively impact both the consumer electronics (CE) system integrators and the end users. Ensuring security against IP piracy

is critical for several reasons: first, counterfeit designs are often not subjected to the rigorous testing necessary to guarantee reliability and safety. Second, these counterfeit IPs may contain hidden malicious logic, such as hardware Trojans, which compromise the security and functionality of the devices. When such infected IPs or ICs are integrated into CE systems, they become unreliable and pose significant safety risks to consumers.

Moreover, the issue of IP counterfeiting becomes evident when individuals within a rogue foundry, in collaboration with new or secondary IP sellers, engage in the unauthorized replication or imitation of the original IP design. This illicit activity not only undermines the IP rights of the original IP seller but also poses grave risks to consumers. The compromised quality and performance of counterfeit products, often stemming from the use of inferior materials or outdated technology, have a detrimental impact on the overall functionality and reliability of the systems in which they are incorporated. Moreover, the proliferation of counterfeit components tarnishes the reputation of authentic IP sellers, casting doubt on the integrity of their products and services. This erosion of trust can have far-reaching consequences, particularly in critical sectors such as military systems, aviation, automotive industries, and beyond. These vital applications rely heavily on the authenticity and quality of the components they integrate, making them especially vulnerable to the repercussions of IP counterfeiting. Furthermore, the ease with which intentional hardware Trojans or malicious logic can be incorporated into counterfeit/pirated IPs exacerbates the security risks inherent in the integrated design supply chain system. This presents a significant challenge for ensuring the integrity and safety of the products and systems reliant on these components [20], [21], [31]-[41].

1.4.2. False claim of IP ownership

In the IC supply chain, an adversary/deceitful IP buyer, possibly within the SoC integrator and foundry houses, may fraudulently claim ownership of an IP, causing substantial financial loss to the original IP owner/seller. This false claim of ownership is a growing security concern. Traditional IP protection methods like trademarks, industrial design rights, patents, and copyright, are not effective for reusable IP designs. Therefore, safeguarding the ownership

rights of the actual IP owner is crucial. To address this, embedding a designer's signature (watermark) covertly within the IP core during its design process can be an effective strategy. This hidden signature serves as proof of ownership, enabling the original IP vendor to verify their rights and counteract fraudulent ownership claim by the adversary [20], [21], [40], [41].

Hardware watermarking serves as an essential tool to secure the hardware design from hardware security threats. The importance of hardware watermarking in the field of hardware IP core protection includes (a) protection from IP piracy: hardware watermarking serves as a detective countermeasure against IP piracy and false IP ownership assertion. By embedding a unique watermark, the original seller/vendor/designer can assert their ownership and identify any instances of piracy, (b) enhancing design integrity: embedding a watermark within a hardware design also ensures the integrity of the design by making it difficult for malicious actors to alter the hardware without affecting the watermark. Any tampering with the watermark would indicate a potential breach or unauthorized modification, (c) enabling traceability and accountability: watermarked hardware can be traced back to the original designer or manufacturer, which is essential for accountability in the global supply chain. This traceability helps in maintaining a transparent and secure supply chain, reducing the risk of counterfeit components being introduced, and (d) fostering trust in the market: the use of hardware watermarking enhances trust among stakeholders, including manufacturers, designers, and end-users. When the authenticity and ownership of hardware can be reliably verified, it fosters a trustworthy market environment where high-quality and original designs are valued.

1.4.3. Hardware trojan attack

Hardware Trojans can be embedded by malicious actors at any stage of the chip design process, posing significant threats to the functionality and reliability of electronic systems. Research has demonstrated that functional hardware Trojans can lead to incorrect outputs, compromising the safety and dependability of the end product. When these Trojans are covertly inserted into real-time hardware systems of custom computing devices, they can cause unpredictable and unreliable behavior. From an attacker's perspective, the

motivations behind Trojan attacks are varied and may include a) damaging a company's reputation and market standing to gain a competitive advantage, and b) causing the malfunction of electronics used in critical infrastructure, which could disrupt safety and mission-critical applications. Therefore, it is the responsibility of system integrator to develop a Trojan free or resistant hardware design, as the third-party IP vendors supplying IPs for integration may contain functional hardware trojans in them. These Trojans can remain dormant and activate only under specific conditions, detecting and isolating them during the testing phase of the DSP hardware IP core is extremely challenging, complicating the defense against such attacks [25], [53], [57].

1.5. Structure of the Thesis

The chapters of this thesis are structured as follows: Chapter 2 reviews the state-of-the-art techniques relevant to the proposed research. Chapter 3 presents the proposed exploration of low-cost hardware IPs during HLS using multiphase encryption and crypto-chain signature framework. Chapter 4 introduces an enhanced security framework for hardware IPs using IP seller's protein molecular biometrics and facial biometric-based encryption key. Chapter 5 describes the proposed method for securing hardware IPs by exploiting statistical watermarking using encrypted dispersion matrix and eigen decomposition framework. Chapter 6 discusses the proposed security framework for securing GLRT cascade hardware IP using IP seller's fingerprint and CIG framework for ECG detector. Chapter 7 outlines a novel security methodology by exploiting voice biometric-based watermarking framework for securing hardware IP cores. Chapter 8 proposes an HLS-based exploration of low-cost (optimal) functional trojan-resistant hardware IP designs. Chapter 9 presents the experimental results of the proposed techniques and compares them with existing state-of-the-art methods. Finally, Chapter 10 concludes the thesis and outlines potential directions for future research.

Chapter 2

Literature Survey: State-of-the-Art

Over the past few years, various hardware security techniques have been developed to address threats to IP cores during the IC design process. This chapter reviews these state-of-the-art techniques and identifies their limitations, forming the foundation for the proposed hardware security methods tailored for hardware IPs in this thesis. The first section covers the state-of-the-arts in countering IP piracy and false IP core ownership claim, as well as hardware Trojan detection methods. The second section outlines the objectives of the thesis, followed by a discussion of the key contributions made in this thesis in section three.

2.1. State-of-the-Art on Hardware IP Attacks

As discussed in the previous chapter, the incorporation of pirated IP versions into system in the hardware SoC designs can result in several significant consequences: (i) it can pose safety risks to end consumers, (ii) the system may malfunction (unreliable functioning) due to hidden malicious logic, such as hardware Trojans, within the IP. These compromised IPs or ICs are unreliable and unsafe when used in consumer electronics, (iii) they may create security vulnerabilities, particularly in critical applications like military systems , medical diagnostics, aerospace, etc., (iv) they can lead to financial losses for the original IP vendors/designers and tarnishes the reputation of the original IP vendor. Thus, detecting and isolating pirated/counterfeited IP versions is essential to maintaining system integrity and safety. Section 1.4.1 of Chapter 1 discusses the threat of IP piracy/counterfeiting in detail. Additionally, protecting the rights of original IP vendors against false ownership claims is equally important. Section 1.4.2 of Chapter 1 discusses the threat of false IP ownership claim in detail. Moreover, the presence of hardware Trojans in IP designs is regarded as a significant threat, as they can lead to various security issues. Among these, the most alarming is the risk of incorrect functional computation. Section 1.4.3 of Chapter 1 discusses the threat of hardware Trojan in detail. In the literature, various detective control

mechanisms are explored to combat the challenges of IP piracy and resolve disputes over false IP ownership claims, emphasizing the need for robust security measures to safeguard intellectual property rights. Additionally, various hardware Trojan detective mechanism have been explored in the literature.

Detective control mechanism against IP piracy and false IP ownership claim: To prevent the integration of pirated IP versions into multimedia, electronics and computing systems, various security techniques have been developed in the past. The techniques for hardware security against IP piracy and false IP ownership claim include hardware watermarking [31] - [35], [36], [42], [93] steganography [37], encryption based security approaches [38], [39], [43], and biometrics-based security techniques [40], [41], [44]. Apart from security, there also exists some works on generating low-cost hardware designs, such as [32], [45], [46] - [49].

As discussed in the introduction section, one common approach at this level is hardware watermarking, which embeds a unique identifier into the design to secure IP designs. Koushanfar *et al.* [31] discusses a dynamic watermarking methodology for DSP IP cores using a binary variable (0/1) encoding process. This is achieved by adding watermarking constraints (additional edges) into a color interval graph (CIG) of the hardware design, which represent the IP vendor's watermark. Initially, the author's signature data is processed through the MD5 cryptographic hash function. This hash is then encrypted using the designer's RSA public key. The resulting cipher is inputted into the RC4 stream cipher, which creates a pseudorandom keystream. This keystream is combined with the original signature data using a bitwise ex-or operation to generate the ciphertext signature data. Lastly, this ciphertext is embedded as additional watermark constraints in the design. Post-generation of watermarking constraints, they are embedded during the register allocation phase of HLS process using CIG framework. The embedded watermarking/security constraints provides detective countermeasure against IP piracy and false IP ownership claim.

Next, Sengupta and Bhadauria [32] proposed hardware watermarking approach with a quadruple variable encoding mechanism by exploiting the

register allocation phase of HLS process. This approach uses multi-variable (quadruple variable) signature encoding, which enhances robustness by employing a complex encoding process with four watermarking variables, resulting in multiple constraints for embedding into the design. To optimize the process, Particle Swarm Optimization (PSO) is used to balance latency and area overhead, achieving a cost-effective solution. The embedded security constraints enable the detection of piracy and resolution of false IOP ownership claim in case of an ownership conflict, providing a reliable means of safeguarding IP. Another notable approach by Hong and Potkonjak [33] involves using a watermarking technique where the vendor's signature or covert mark is encoded as a set of design and timing constraints and embedded into the IP core during behavioral synthesis. Detection of pirated IPs is facilitated by identifying the presence of the vendor's watermark, thereby ensuring that only authorized versions of the IP are used.

Next, Gal and Bossuet [34] developed a watermarking technique based on mathematical relations between input/output data and initial internal values at certain timing values. This watermark not only protects the IP owner's rights but also ensures that the design meets user constraints related to latency and area. To minimize overhead in terms of area, delay, power consumption, and design time, the watermark is integrated automatically during the behavioral synthesis phase using the HLS tool. This method leverages "temporally free" output slots to embed watermarking constraints. The watermark in [34] comprises of mathematical relationships among the IP's input data, initial internal values, and output data, referred to as sub-marks. These sub-marks are indistinguishable from normal output data, making the watermark invisible to IP buyers, integrators, and users, and undetectable during static analysis. Two watermarking algorithms are proposed in [34]: (i) a low-cost watermark and (ii) a costless watermark. The low-cost version randomly selects internal computation values and transfers them to free output slots, while the costless version further reduces the set of internal values. The technique is effective for applications like digital signal, image, and video processing but is unsuitable for data security applications due to potential security breaches from exposed internal data. This method allows the integration of security features without

significantly impacting the performance of the design. Subsequently, Karmakar and Chattopadhyay [35] explored IP protection through a combination of watermarking and logic encryption. They examined vulnerabilities in existing logic encryption techniques and proposed the use of cellular automata to watermark finite state machine designs. This approach not only enhances the security of the IP but also adds a layer of protection against unauthorized modifications and use.

A three-phase watermarking-based security technique is proposed by Sengupta *et al.* [36] for securing IP designs. [36] involves multi-variable signature encoding, using seven variables to generate and embed the watermark into three different phases of HLS process, to secure the design against IP piracy and illegal IP ownership claim. The vendor's signature is embedded during three separate phases of the HLS process: the *scheduling phase*, the *hardware allocation phase*, and the *register allocation phase*. Initially, operations are sorted in ascending order within each control step (CS). During the first phase, non-critical operations (starting from CS-1) are shifted to the next CS for each occurrence of signature bit γ, ensuring data dependency and hardware constraints are maintained. This generates a modified timing table for non-critical operations. In the second phase, functional units (FUs) are reallocated based on the encoding rules α and β, creating an updated hardware allocation table. Storage variables in the SDFG are then allocated. A register allocation table (RAT) is then generated from the SDFG. Watermarking constraints, determined by the IP seller's selected encoding digits *i*, *I*, *T*, and *!*, are embedded into the RAT/CIG. Finally, the RAT of the triple-phase watermarked hardware IP core is generated using HLS. This phased approach ensures that the watermark is deeply integrated and uniformly distributed into the design, making it difficult to remove or alter.

Next, Roy and Sengupta [93] developed a multi-level watermarking technique, specifically designed to secure DSP IP cores against piracy. This approach involves embedding hardware security constraints that correspond to the vendor's signature at multiple design abstraction levels, including high-level and register transfer level. The process begins by accepting the CDFG of the DSP application and performing key tasks such as scheduling based on

resource constraints, allocation, and resource binding. Afterward, the RTL design is obtained using an HLS framework, which includes components like multiplexers, demultiplexers, and registers. The vendor's signature is then decoded to generate watermarking constraints, which are embedded by adjusting the hierarchy of multiplexers and demultiplexers and encoding register sharing. The resulting multilevel watermarking-based RTL design provides a comprehensive and robust solution for securing IP cores against piracy and false IP ownership claim.

Next, Chen and Schafer in [42] have discussed a practical watermarking method for commercial HLS tools. The approach exploits pragma directives for embedding watermark signature in the functional unit allocation phase of HLS process. The process begins by taking the initial behavioral description to be watermarked and transforming it to highlight all operations requiring a functional unit (FU). Next, the expanded behavioral description is synthesized to determine the scheduling of operations across clock cycles. This yields a scheduling report and an FU constraint file from the HLS process. The core watermarking step involves creating a distinct FU binding solution, ensuring the resulting RTL code is unique. Since the watermark relies on FU binding, it remains undetectable. Inputs for this step include the expanded C code, HLS scheduling report, FU constraint file, and the watermark key. The output is a modified C code with pragmas linking operations to specific FUs. The objective is to follow the sequence defined by the watermark key and balance FU usage to minimize multiplexer area. Finally, the uniquely watermarked C code is synthesized to produce the watermarked RTL code with a distinct FU binding pattern. The primary weakness of this approach is its limited security strength due to lesser watermark strength.

Further, Sengupta and Rathor [37] introduced a hardware steganography-based technique aimed at detecting pirated DSP IP versions before their integration into electronics and computing systems. [37] generates stego-constraints based on design data, secret stego-keys, thresholding parameter, and mapping rules. Further, it embeds these generated stego-constraints in the form of secret information into target hardware. The complex process of stego-constraints generation using secret stego key that renders the

steganography approach stronger than watermarking. This method embeds covert stego-marks directly into the hardware IP design without relying on any external signature. The amount of covert digital evidence (stego-constraints) embedded is controlled by a thresholding parameter set by the designer. The process begins with accepting the CDFG of the hardware application, which is then transformed into a scheduled data flow graph (SDFG). Next, the corresponding CIG is constructed, and stego-constraints (artificial edges) are determined for insertion into the CIG. Swapping pairs are identified for each stego-constraint, and the maximum entropy is calculated. A subset of stego-constraints are selected based on a threshold value chosen by the designer, and added to the CIG, resulting in a secured design with embedded stego-constraints. Next, Yu and Zhu [38] presented a hardware description language (HDL) design-level IP watermarking approach using SHA1 and RSA. In [38], a specialized watermark module is introduced into the original HDL code, replacing a specific set of stable register data. This watermark remains intact through the synthesis, placement, and routing stages, providing resistance against forgery and removal attack also.

Moreover, Sengupta *et al.* [39] proposed a digital signature-based approach for providing detective countermeasure against piracy and false IP ownership claim, utilizing encrypted-hash techniques to secure reusable IP cores. Authors in [39] employed the RSA cryptosystem and SHA-512 hash computations to generate security constraints. The process begins by taking the CDFG of the hardware application and IP vendor-specified resource constraints as inputs. Based on this, the input CDFG undergoes scheduling, and the resulting SDFG is input into a phase-1 encoding process, which generates a bitstream using specific encoding rules. This bitstream is then processed through the SHA-512 hashing algorithm, resulting in a bitstream digest of the DSP application. The generation of this digest involves various computations, including circular right shifts, left shifts, and modulo additions on 64-bit arguments. In the subsequent post-processing phase, the generated bitstream is divided into equal-sized blocks and converted into their equivalent decimal values. These decimal values are then encrypted using the IP owner's private key through RSA encryption, enhancing the security of the embedded digital signature.

The encrypted data is converted back into a binary bitstream during further post-processing steps. This encrypted bitstream is then used as input for the phase-2 encoding process, which generates covert security constraints that correspond to the digital signature's strength—a parameter chosen by the IP designer, balancing security needs with design cost considerations. Finally, these covert security constraints are embedded during the register allocation phase of the HLS process, resulting in a secured reusable hardware IP core with an embedded digital signature. Next, Castillo *et al.* [43] presented an encryption-based hardware watermarking approach using MD5 and SHA-crypto algorithm. This method involves embedding the bits of a digital signature directly at the HDL design level, utilizing resources already present within the original system. It also incorporates a secure signature extraction process that necessitates only minor adjustments to the existing system.

Further, the biometric-based hardware watermarking techniques are the most recent one, which includes the use of IP vendor's/seller's biometric traits, to generate and embed a robust watermark. Hardware watermarking [31], [32], [33], [34], [35], [36], [38], [39], [42], [43], [93] methodologies involve embedding a seller's signature into hardware IP design. Adversaries might be able to forge and replicate the watermark to evade detection or make false ownership claim. Further, regular watermarking and steganography techniques are not capable of producing large-size signature strength, which in turn leads to a lack of sufficient uniform distribution of the watermark constraints during embedding. Weak distribution is prone to removal by an attacker, thereby compromising security. Therefore, using the IP seller's biometric traits for sophisticated watermarking ensures a unique and tamper-resistant watermark signature of large strength, facilitating seamless detection of piracy and verification of genuine ownership. Biometric-based watermarking offers several advantages over traditional methods, including uniqueness, robustness, and stronger security. Authors in [41], [44], and [41] have exploited IP seller's facial, palmprint, and fingerprint biometric information, respectively to generate a robust watermark. Initially, a high-resolution image of the IP seller's facial and palmprint biometric is captured using digital cameras. Similarly, a fingerprint biometric scanner is used to capture the image of an IP

seller's fingerprint. Subsequently, the captured biometric image is subjected to the IP seller's specified grid size and spacing to extract the precise feature nodal and minutiae points. Additionally, the fingerprint image is subjected to various preprocessing steps such as FFT enhancement, binarization, and thinning to improve the quality of the image, which facilitates a smooth extraction of minutiae feature points. Post-generation of nodal points, the corresponding features are generated on the facial and palmprint biometric image. Similarly, the minutiae points (comprising of bifurcation and ridge ending points) are generated on the fingerprint image. The coordinates corresponding to each nodal and minutiae point are extracted for feature dimension computation Similarly, the parameters corresponding to fingerprint minutiae feature points, such as *x-coordinate*, *y-coordinate*, *ridge angle*, and *minutes type* (*i.e.,* ridge ending and bifurcation), are extracted. Post feature dimension computation, all decimal values are converted into their binary equivalents, which are further concatenated as per IP seller's concatenation fashion to generate the final corresponding facial, palmprint, and fingerprint biometric watermark signatures. The generated individual biometric watermark signature is converted into watermarking constraints using mapping/embedding rules. Finally, the determined watermarking constraints are implanted into the hardware design as the IP seller's digital evidence during the register allocation phase of the HLS process. The implanted digital evidence provides a detective countermeasure against IP piracy and an instance of false assertion of IP ownership. Overall, by embedding security measures directly into the design process, these aforementioned techniques provide security/detective countermeasure against IP piracy and false IP ownership claim. Each approach offers a unique way to integrate security into different phases of the IC design process.

In addition to security-focused designs, several studies [32], [45], [46], [47], [48], [49] have explored methods for generating low-cost hardware designs through design space exploration (DSE). These works often emphasize optimizing factors such as power performance, power-delay tradeoffs, and multi-objective optimization to develop cost-effective solutions. For instance, the study in [45] used HLS methodologies aimed at creating low-power

designs on FPGAs, concentrating on throughput constraints without addressing area-delay tradeoffs for secure designs. Other research, such as [46], applied machine learning techniques for DSE but did not focus on generating secure, low-cost architectures for image processing cores. Tools like Autopilot for HLS [47] and approaches such as simulated annealing [48] have also been utilized to explore power-performance tradeoffs, though these methods often overlook tradeoffs involving design area, latency, and security. Additionally, the bacterial foraging-driven DSE approach discussed in [49] targets the creation of low-cost designs for fault-tolerant hardware systems, further highlighting the diverse approaches to achieving efficient hardware designs.

Detective control mechanism against hardware Trojans: Further, over the years, various methods have been developed to detect hardware Trojans in ICs, with some approaches focusing on Trojan detection without necessarily making the designs resistant to such threats. For example, research by Sengupta and Mohanty [45] explored low-cost scheduling strategies during HLS to develop DSP IP cores that can detect Trojans but cannot prevent them. Another approach by Sengupta *et al.,* [46] used property checking techniques, where specific properties of the IP are verified to ensure they meet expected behaviors. If discrepancies are found, they may indicate the presence of a Trojan.

Next, code-coverage analysis, as proposed by Hu [47], is another technique used for hardware Trojan detection. This method identifies suspicious signals in the RTL design by analyzing which signals remain stable during coverage testing. Since Trojans typically activate only under specific conditions, these stable signals are flagged as potential indicators of malicious hardware. Additional test vectors can then be applied to further investigate uncovered parts of the design. However, this approach can be time-consuming and may not definitively distinguish all Trojans from other anomalies. To refine this method, Bushnell and Agrawal [48] employed equivalence analysis, which helps reduce the number of flagged suspicious signals, although it introduces runtime overhead and may still misclassify some non-Trojan signals (as not all suspicious signals are Trojans).

Some researchers have developed system-level Trojan detection techniques, such as those presented by Tehranipoor and Koushanfar [49], while others have explored lower-level detection methods [50], [51]. A notable technique proposed by Wang *et al*., [52] involves using multiple supply transient current integration to detect Trojans by monitoring abnormal current variations in the IC. Once detected, Trojans are isolated from the system through a defined isolation process. Another strategy, concurrent error detection (CED), was utilized by Rajendran *et al*., [53], although it relied on multiple sets of third-party IP (3PIP) vendors for identifying Trojans, which complicates implementation and still does not render the system Trojan-resistant.

Despite the advances in Trojan detection, most of these methods only focus on identifying the presence of Trojans without enhancing the resistance of the hardware against such attacks. To further improve hardware security, researchers have explored various other mechanisms, including the use of approximate circuits to reduce the risk of Trojan insertion [54], obfuscating triple modular redundancy (TMR) techniques [55], and neutralizing Trojans in Supervisory Control and Data Acquisition (SCADA) systems [56]. In [54], a gate-level approximation circuit scheme was proposed, demonstrating its effectiveness on ISCAS C-series benchmark circuits by reducing the likelihood of Trojan insertion. The obfuscated TMR approach discussed in [55] was also applied to C-series benchmark circuits, enhancing Trojan detection coverage by camouflaging low-observable signals. The work in [56] focused on securing SCADA systems commonly used in industrial control applications, employing TMR on select pathways to neutralize potential hardware Trojans. Additionally, [57] explored functional camouflage to design adversarial hardware that covertly inserts Trojans in low-centrality locations within the circuit, making them harder to detect. Moreover, [58] utilized equivalence checking based on finite state machines with datapath (FSMD) to identify Trojans that cause functional changes in the hardware.

Machine learning (ML) has also been leveraged for real-time hardware Trojan detection, as highlighted by Kulkarni *et al*., [59], which presents an ML-based methodology that improves the accuracy of Trojan detection in many-core designs. Further, Kulkarni *et al*., [60] introduced a discrepancy analysis (DA)

based security approach that automatically detects hardware bugs, while Abderehman *et al*., [61] proposed a C/C++ to RTL equivalence checking framework specifically for HLS verification. Moreover, Fern and Cheng [62] utilized a simulation-based assertion set completeness analysis to uncover hardware Trojans and address verification blind spots. A reconfigurable assertion checker-based security framework was also suggested by Alsaiari and Gebali [63] for detecting hardware Trojans within SoC designs. Overall, while numerous Trojan detection techniques have been developed, many still focus primarily on detection rather than prevention or resistance. There remains a need for more robust, integrated solutions that can secure hardware IP cores against both the insertion and activation of Trojans, ensuring comprehensive protection throughout the design and manufacturing processes.

Limitations: In hardware watermarking techniques [31], [32], [33], [34], [35], [36], [42], [93], the generated signatures rely on factors like the number of variables, their combinations, watermark signature length, and encoding rules. This dependency on intermediate factors makes watermarking susceptible to attacks, as these elements can be easily compromised. On the other hand, hardware steganography [37] offers a signature-free method to protect hardware IP cores, providing stronger security with lower design overhead compared to watermarking. However, steganography also has its vulnerabilities; an adversary could potentially exploit stego-keys, encoding methods, and threshold entropy value, undermining the security of the system. The primary weakness of the above methods [31]-[38] lies in their limited security variables, such as private keys, encoding algorithms, and signature combinations. These factors can be targeted by adversaries to replicate or regenerate signatures, compromising the security of hardware IP cores against piracy. Additionally, these methods do not focus on creating cost-effective (low-cost), secure RTL IP datapath architecture. Apart from all of the above limitations, these watermarking approaches leads to generation of limited watermarking (security) constraints, which in turn decreases the robustness of the security methodology.

Next, the digital signature based approaches [38], [39], [43] involves generating digital signatures (watermark) through encryption algorithms, such

as MD5, SHA, SHA-512, and RSA cryptosystem, and encodings. While these approaches use intricate calculations/computations to prevent watermark/signature replication, their reliance on standard encryption algorithms still make them vulnerable to key-based attacks. In case of a leaked/forged encryption key, the overall security can be compromised. The attacker can easily regenerate the exact watermarking constraints with the help of compromised encryption key and encoding rules. These approaches have also a primary limitation of limited watermarking constraints generation, apart from generation of a low-cost secure RTL design. Additionally, the hardware watermarking and digital signature/encryption based security approaches do not integrate/embed any unique natural identity of original IP seller with the hardware design.

The biometric-based approach [40] for generating accurate fingerprint signatures involves an image enhancement phase using FFT, which adds complexity and also requires an optical scanner to capture the IP vendor's fingerprint. This method is also susceptible to inaccuracies due to injuries or external factors that can affect the accurate fingerprint generation. Alternatively, facial [41] and palmprint [44] biometric approaches use naturally unique IP vendor's facial and palmprint features to generate signatures. While these methods embed the IP vendor's natural identity into the design, they still fall short in providing robust security because they generate fewer watermarking constraints. Consequently, despite leveraging unique biometric features, these approaches do not fully ensure the protection of hardware IP cores, leaving room for potential security breaches. A more resilient strategy that combines enhanced security measures with biometric uniqueness is needed to overcome these limitations and offer comprehensive protection against piracy and unauthorized use.

In summary, more comprehensive and resilient approaches are needed to safeguard hardware IPs effectively from unauthorized replication and exploitation. Security approaches that have capability to generate massive watermark strength along with IP vendor's natural uniqueness are the need of the hour, as they are able to provides robust digital evidence (author credibility proof) against IP piracy and false claim IP ownership. Further, they must also

depict higher withstand ability against standard threats of ghost insertion search attack (watermark collision), tampering attack (brute force), forgery attack and watermark removal attack.

Further, existing research on Trojan detection, such as in [45] and [46], addressed detection but lacks comprehensive resistance strategies against hardware Trojans. Further, [47]-[53] also discussed Trojan detection; however, fails to provide complete Trojan resistance. Next, [54] and [55] do not focus on functional hardware Trojan isolation within DSP hardware IP cores, particularly in the context of third-party IP (3PIP) cores. Moreover, these works do not explore the development of optimized (low-cost), secure architectures resistant to Trojans. [55] falls short in handling DSP hardware IPs, while [56] struggles with managing the design overhead caused by triplication logic. Although the approaches in [57]-[63] present various Trojan detection techniques, they do not extend to providing Trojan resistance, particularly in DSP hardware circuits. This highlights a significant gap in the field: the need for more robust and integrated solutions that not only detect but also resist functional Trojan insertions and activations, ensuring complete protection of hardware IP cores throughout the design and manufacturing lifecycle.

2.2. Objective of the Thesis

The objective of the thesis is to develop novel alternative paradigms of hardware security for addressing threats of IP piracy and Trojan during HLS. This is achieved by setting out the following goals and objectives:

1. To explore low-cost secure hardware IP design during HLS using multiphase encryption and crypto-chain signature.
2. To develop enhanced security framework for hardware IPs using IP seller's protein molecular biometrics and facial biometric-based encryption key.
3. To develop a statistical watermarking framework using encrypted dispersion matrix and eigen decomposition framework for securing hardware IPs.

4. To develop a secure GLRT cascade hardware IP using IP seller's fingerprint and CIG framework for ECG detector.
5. To develop a voice biometric-based watermarking framework for securing hardware IP cores.
6. To develop an HLS-based, low-cost (optimal) functional trojan-resistant hardware IP designs security framework.

2.3. Overview of Key Contributions

- A novel low-cost security framework for hardware IP design during HLS using multiphase encryption and crypto-chain signature. (Publications: #10, #11, #15, #28, #30)
  - Proposes a novel low-cost exploration framework of secured image processing filter IP core datapath architecture for detective control against IP piracy during HLS.
  - Exploits PSO based design space exploration process for performing design area-delay tradeoff of secured image processing filter IP core datapath.
  - The proposed approach explores low-cost optimized design architecture of filter IP core datapath that embeds robust security constraints based on the proposed multi-phase encryption algorithm at zero design cost overhead. The proposed approach demonstrates the exploration and embedding of low-cost resource configuration and watermarking constraints on sharpening filter.
  - It also proposes a firefly based design space exploration to determine an optimal JPEG-CODEC IP core datapath after performing the design area-delay tradeoff.
  - Presents low-cost hardware security approach to explore optimal architecture (design) for JPEG-CODEC IP core datapath that contains secret watermarking/security constraints.
  - The secret security constraints are generated using the proposed key-driven crypto-chain based security methodology/algorithm. It explores a

secured JPEG-CODEC IP core datapath capable of providing detective control against IP piracy and fraudulent claim of IP ownership using the HLS framework.

- Presents the optimality analysis of the proposed low-cost multi-phase encryption algorithm and proposed low-cost key-driven crypto-chain based security methodology.

- A novel enhanced security framework for hardware IPs using IP seller's protein molecular biometrics and facial biometric-based encryption key. (Publications: #1, #18, #24)
  - Proposes a novel molecular biometric-based hardware security approach based on protein molecule sequence to secure hardware IP cores.
  - In the proposed approach, an IP vendor selected protein sequence comprising of 20 unique amino acid combinations, is used for molecular signature generation.
  - The generated signature (watermark) is then encrypted through AES using an encryption key generated with the facial biometric of authentic IP vendor. Thus, the proposed approach incorporates two classes of biometrics of IP vendor to ensure highly robust and unique authentication.
- A novel statistical watermarking framework using encrypted dispersion matrix and eigen decomposition framework for securing hardware IPs. (Publications: #6, #17)
  - Proposes an HLS based watermarking methodology using design parameter driven encrypted dispersion matrix with eigen decomposition based security framework for protecting hardware IP cores.
  - Presents a security framework that extracts the characteristics of the IP vendor selected design space parameters and the design space's characteristics in terms of IP vendor chosen resource configuration values and exploits them as unique features to act as digital evidence for securing hardware IP cores.

- Exploits HLS design methodology for embedding the mathematical watermark signature, generated using dispersion matrix, Eigen decomposition, and AES encryption block, during the register allocation phase. It also demonstrates the complete end-to-end watermarking algorithm and its embedding on an 8-point DCT hardware IP core.
- Depicts stronger security with a lower probability and higher tamper tolerance value at minimal design cost overhead.
- Presents security analysis of the generated template against forgery, ghost signature search and brute-force/tampering attack.

▪ A novel secure GLRT cascade hardware IP design using IP seller's fingerprint and CIG framework for ECG detector. (Publications: #4)

- Presents the design methodology of GLRT hardware IP core for ECG detector for the first time in the literature.
- Presents secure GLRT hardware IP core for ECG detector using fingerprint biometric-based security methodology during HLS.
- Presents CIG framework and RTL datapath of a secure GLRT hardware micro IP core and secure GLRT hardware cascade IP core.
- Discusses the security of life-critical critical medical hardware systems for first time in literature.

▪ A novel voice biometric-based watermarking framework for securing hardware IP cores. (publications: #2)

- Proposes a novel contactless voice biometrics-based hardware watermarking technique for robust IP core authentication and verification. This is the first voice biometric-based hardware IP protection technique.
- Presents a security framework for generating a unique voice signature digital template using distinct voice features such as jitter and shimmer along with pitch and intensity values.

- Present a feature extraction scheme for extracting different pitch and intensity values at different timestamps from the spectrograph of the voice sample.
- Present a scheme of encoding voice signature template into covert hardware security/watermarking constraints based on four-fold mapping.
- Presents the HLS design methodology of embedding a voice signature during the register allocation phase to generate secured IP cores. It demonstrates the embedding of voice signature on IIR filter.
- Depicts stronger security with a lower probability and higher tamper tolerance value at minimal design cost overhead.
- Presents security analysis of the generated template against forgery, side channel attack (SCA), ML-attacks, ghost signature search and brute-force/tampering attack.

▪ A novel HLS-based, low-cost (optimal) functional Trojan-resistant hardware IP designs security framework. (Publications: #12)

- Presents a novel exploration framework of optimized Trojan resistant (capable of detection and isolation both) hardware design architecture during HLS process.
- Exploits particle swarm optimization-driven design space exploration (PSO-DSE) to determine an optimal hardware IP core datapath after performing the design area-delay tradeoff.
- Proposes a Trojan-resistant design flow for the reusable hardware IP core using TMR-based distinct multivendor allocation policy.

Chapter 3

Exploration of Low-Cost Hardware IPs during HLS using Multiphase Encryption and Crypto-Chain Signature

The widespread use of electronics and multimedia devices, such as smartphones, digital cameras, and IoT devices, underscores the importance of efficient data processing and management technologies. Key among these are JPEG-CODEC and image processing filters, which play pivotal roles in enhancing the functionality and performance of these devices. JPEG-CODEC, a popular compression-decompression standard, enables efficient storage and transmission of multimedia content by significantly reducing the file sizes of images and videos without compromising essential quality. This is particularly crucial in scenarios where storage space and bandwidth are limited, such as in digital cameras and medical imaging equipment like MRI and CT scanners. In medical applications, the JPEG-CODEC facilitates the handling of high-resolution images by compressing them, thus optimizing storage and streamlining the transfer of critical data for remote diagnosis and treatment. Further, image processing filters are equally important, as they perform a range of functions that are essential for extracting meaningful information from images. These filters are used for tasks such as noise reduction, edge detection, and image enhancement, which are critical in applications spanning from military and robotics to advanced medical imaging and biometric systems. For instance, they help in analyzing medical images for disease diagnosis or in identifying objects in automated systems [72] - [76]. The development of these technologies as dedicated, low-cost reusable hardware IP cores using HLS and DSE enhances their performance and cost-efficiency, making them integral components in the modern digital ecosystem. By optimizing (using DSE) these hardware designs, it is possible to meet the stringent requirements of speed, power efficiency, and accuracy that are demanded in real-world applications.

Further, with the globalization of the digital design process, these hardware IP cores face significant security challenges, including IP piracy, counterfeiting,

and false IP ownership claim. These threats can lead to unpredictable device behaviors, such as incorrect pixel computation in medical imaging or data leaks, posing risks to users and manufacturers. Therefore, it is essential to prioritize security alongside performance and cost optimization during the design of these IP cores using high-level synthesis frameworks to ensure reliable and secure hardware solution. The details on the involved threat model is discussed in the first section of this chapter.

Previous research has explored the design of hardware accelerators for image processing, including FPGA-based solutions [64] - [67] and those utilizing convolutional neural networks (CNNs) [68] - [71] for convolutional tasks. However, these studies have not presented a dedicated low-cost design framework for JPEG-CODEC and image processing filters, beside addressing the security challenges, such as IP piracy, associated with dedicated hardware accelerators for image processing filters. Further, several state-of -the-art watermarking techniques and their limitation has already been discussed in the previous chapter. The proposed low-cost security (watermarking) approaches involves multi-layer security through usages of several IP vendor selected key values, apart from the generation of greater watermarking constraints. Moreover, the proposed approaches incorporates design space exploration block along with security block to generate a low-cost optimized hardware architecture.

This chapter presents the proposed two low-cost security approaches for generating low-cost secure image filters and JPEG-CODEC RTL datapath. The first section of the chapter outlines the problem formulation, threat model and undelaying motivation. The second section discusses the details of proposed low-cost multiphase encryption and crypto-chain signature based security methodologies. Following this, the third section illustrates the embedding of the proposed watermarking constraints with relevant examples. The fourth section then covers the process of watermark detection. Lastly, the fifth section provides the chapter's conclusion.

3.1. Problem Formulation

3.1.1 Threat model and underlying motivation

Both JPEG-CODEC and image processing filter IP cores face significant security threats, including piracy, counterfeiting, cloning, and fraudulent IP ownership claim. These vulnerabilities arise particularly when multiple third-party entities are involved in the design process, making it easier for adversary in the SoC integrator and fabrication houses, to pirate and resell the original IPs or fraudulently claim ownership. Such actions not only lead to revenue loss for the original IP vendors but also pose risks to end consumers, as pirated IP cores may not undergo rigorous testing and could contain malicious logic. The presence of malicious logic inside a pirated/counterfeited IP version may lead to incorrect pixel computation value (causing severe consequences for end patient in case of medical imagining system), unpredictable device behavior (such as excessive heat dissipation, etc.), and leakage of sensitive information. The designed dedicated reusable IP core is susceptible to piracy when an IP vendor sells the IP core/design to the customer (SoC integrator). Here, a potential adversary or threat actor may be an SoC integrator who may pirate the original design and resell it under the same brand name, making it challenging to make a clear distinction between the authentic and the pirated one. Further, adversary can also claim the ownership of the IP design. Thus, ensuring robust security measures (detective countermeasure) to protect these IP cores is essential. Further, a low-cost design is also crucial to generate an optimized design within given design (area and latency/delay) constraints.

3.1.2. Input and Outputs

The primary inputs are (a) input image pixel matrix, (b) transfer and computation function obtained through image filter kernel coefficients, (c) particle swarm optimization (PSO) initialization parameters (such as swarm size, random number, social and cognitive factors, acceleration coefficients, termination criterion, and inertia weight), (d) module library, (e) LIST scheduling algorithm, (f) different IP vendor selected key values for multi-phase encryption, (g) truncation length, (h) keys for TRIFID cipher computation, (g) encoding rules, (i) transfer function of JPEG-CODEC, (j) firefly algorithm (FFA) initialization parameters (such as attractiveness parameter, step size control parameter, design constraints, absorption coefficients, and population size), (k) IP vendor selected keys for crypto-chain

algorithm, (l) bit-padding and embedding rules, and (m) mapping/embedding rule. And the final outputs consists of low-cost secure RTL datapath corresponding to image processing filter and JPEG-CODEC application.

3.1.3 Target platform

The proposed security methodologies can be seamlessly integrated with any electronic design automation (EDA) tools. The techniques can easily be combined with HDL, or any high-level language used for IP generation within design tools.

3.2. Low-Cost Multiphase Encryption and Crypto-Chain Signature based Security Methodologies

Fig. 3.1 depicts the overview of the low-cost multi-phase encryption based security methodology and Fig. 3.2 depicts the overview of low-cost crypto-chain signature based security methodology. As shown in Fig. 3.1, the proposed multi-phase encryption-based hardware security methodology for protecting image processing filter IP cores involves several key steps. First, the input image and target filter kernel are provided to the approach to generate a secure hardware accelerator design. The image is then converted into its pixel values, which are used along with filter kernel coefficients to formulate the mathematical function of the target image filter IP core. This function is converted into a data flow graph (DFG), which undergoes structural transformations like loop unrolling and tree height transformation to enable parallel pixel computation and improve performance by reducing latency. Next, a heuristic-based architectural exploration, using PSO, is employed to identify a low-cost resource configuration from various potential designs. The inputs for this exploration include the transformed DFG, PSO parameters, and a module library, producing an optimal low-cost configuration. This configuration, along with the transformed DFG, is input into the scheduling, allocation, and binding block of the HLS process, resulting in a SDFG. An initial RAT is generated using the SDFG, and multi-phase encryption-based security constraints are embedded into the RAT, producing a secure RAT. The design cost is then calculated, and the global best solution is identified using PSO. Finally, a low-cost secured hardware accelerator datapath is generated through HLS, embedding security constraints

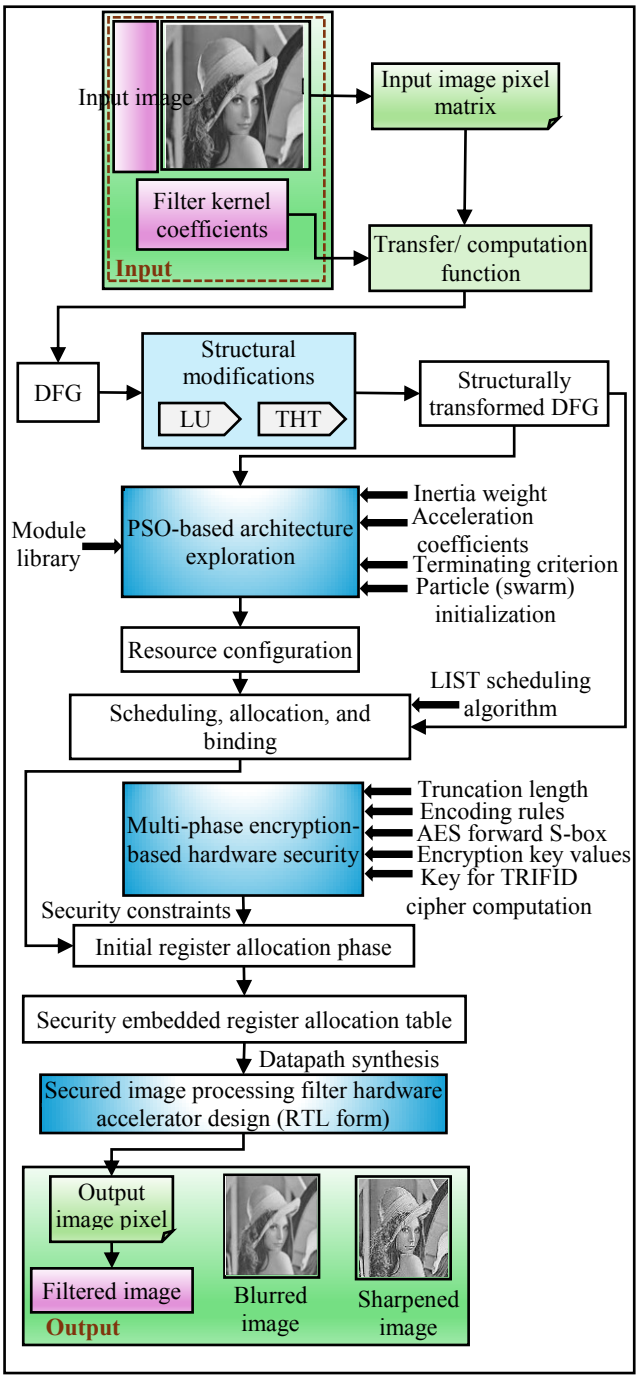


Fig. 3.1 Overview of low-cost multiphase encryption based security methodology

to provide detective countermeasure against piracy while effectively performing various image processing functions.

Next, as shown in Fig. 3.2, the proposed approach for generating a low-cost secure JPEG-CODEC IP design employs the firefly algorithm-based DSE and proposed key-driven crypto-chain hardware security methodology. The methodology comprises two main components: (a) a firefly-based architecture exploration block, which identifies an optimal secured architecture for the

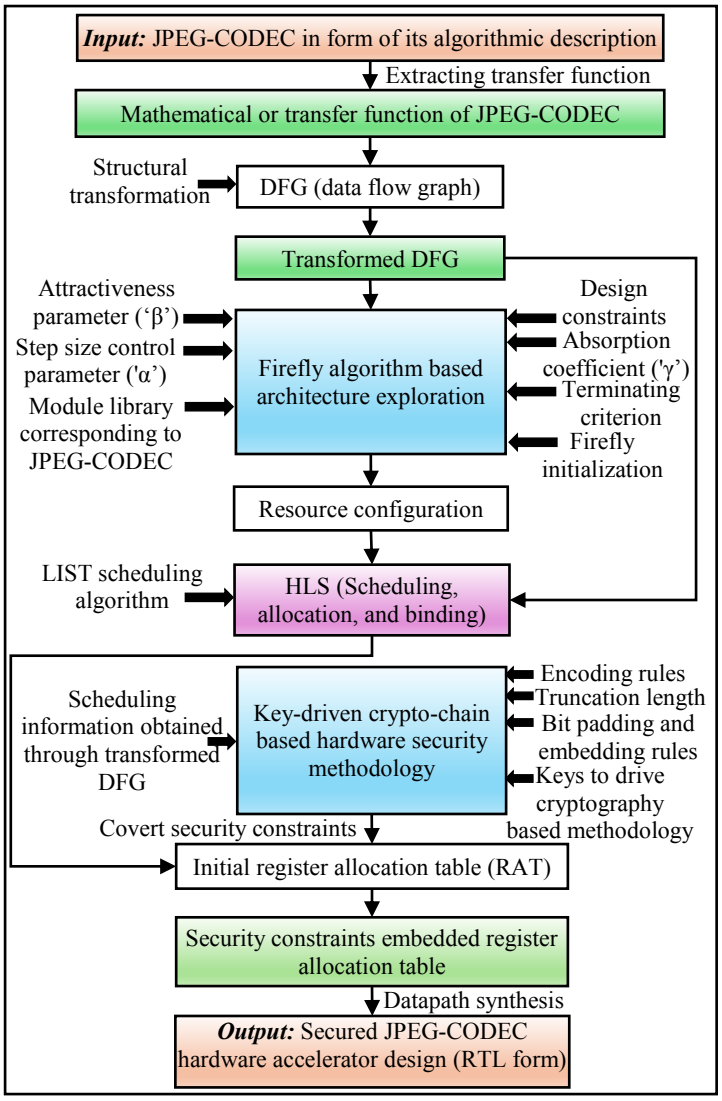


Fig. 3.2 Overview of low-cost crypto-chain signature based security methodology

JPEG-CODEC IP core, and (b) a key-driven crypto-chain hardware security module. The process begins by feeding the JPEG-CODEC algorithmic description into the system, from which a corresponding mathematical function is derived. This function is then converted into a DFG or CDFG. The CDFG undergoes structural transformation using the tree height transformation (THT) technique, which reduces latency by allowing parallel evaluation of sub-computations, thereby enhancing performance. The transformed DFG is fed into the firefly-based architecture exploration block to find an optimal low-cost resource configuration. This configuration, along with the transformed DFG, is then input to the scheduling, allocation, and binding unit of the HLS framework, using the LIST scheduling algorithm to manage control steps and allocate functional units (FUs) and registers. Subsequently, a RAT is generated based on the SDFG. The key-driven crypto-

chain module then produces secret security/watermarking constraints, which are embedded during the register allocation phase (in the RAT) to secure the JPEG-CODEC IP core against hardware threats. The firefly algorithm continues to explore configurations until the optimal, secure architecture is found, resulting in the generation of a secure and optimized JPEG-CODEC IP core RTL datapath design. The details are discussed in the following subsections.

3.2.1. Overview of image processing applications/ filters and jpeg-codec

The filter kernel coefficients of the target image processing filter IP core and input image pixels are used to derive the mathematical function, which is used to generate the DFG of the respective filter IP core. The mathematical kernel function with filter kernel coefficients corresponding to some important image processing filter IP cores such as blur filter (BF), sharpening filter (SF), laplace edge detection filter (LED), vertical embossment filter (VE), and horizontal embossment filter (HE) are given as:

$$Kernel_{BF} = \left(\frac{1}{9}\right) * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} Kernel_{LED} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} Kernel_{SF} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$Kernel_{VE} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} Kernel_{VE} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

For the sake of demonstration, we have considered SF here. The derived function corresponding to SF using sharpening filter kernel coefficients ($Kernel_{SF}$) and input image pixels for performing two parallel pixel computations (by exploiting loop unrolling (LU) transformation) is mentioned in equations (1) and (2), respectively.

$$O_0 = [(I_{00}*(-1)) + (I_{01}*(-1)) + (I_{02}*(-1))] + [(I_{10}*(-1)) + (I_{11}*(9)) + (I_{12}*(-1))] + [(I_{20}*(-1)) + (I_{21}*(-1)) + (I_{22}*(-1))] \quad (3.1)$$

$$O_1 = [(I_{01}*(-1)) + (I_{02}*(-1)) + (I_{03}*(-1))] + [(I_{11}*(-1)) + (I_{12}*(9)) + (I_{13}*(-1))] + [(I_{21}*(-1)) + (I_{22}*(-1)) + (I_{23}*(-1))] \quad (3.2)$$

Here, I_{00}-I_{23} are input image pixel values. The final DFG is generated using equations (3.1) and (3.2), which further undergo another structural transformation, *viz.* tree height transformation (THT), to optimize schedule

latency. The obtained structurally modified SDFG of SF is scheduled based on the output of heuristic-based architecture exploration, *viz.,* four adders and two multipliers using LIST scheduling. *Note: heuristic-based architecture exploration is explained in the next subsection*. An initial RAT is designed using obtained SDFG, which is further used to extract the designer's secret information required to generate secret hardware security constraints based on the proposed multi-phase encryption algorithm.

The JPEG-CODEC application is responsible for performing JPEG compression on the images and is commonly used in several multimedia and consumer electronic devices. The input of the JPEG-CODEC application is a pre-processed image. Mathematically a grayscale image is represented using a pixel intensity matrix ranging from 0 to 255, where 0 denotes pure black, and 255 denotes pure white (this scale is for 8-bit depth grayscale images). Further, *'F'* denotes a generic 2D- discrete cosine transform (DCT) matrix used to process input grayscale images. As *'F'* is an 8×8 matrix and can process a maximum of 8×8 pixel values at one time. Further, f_n indicates the elements of the *F* matrix. Therefore, the input image data is divided and grouped into 8×8 matrix blocks. *'Z'* represents an 8×8 matrix block of the input image in a generic form. The relationship between W_{ij} and Z_{st} variables is defined in equations (3.4), (3.5), and (3.6). The standard quantization matrix *(Q)* is also an important input component. Next, each pixel intensity value from the input 8×8 matrix block is subtracted with 128 as discrete cosine transform coefficient matrix can only handle pixel values from range -128 to 127. Moreover, the JPEG algorithm comprises of steps such as zigzag scanning and run-length encoding to generate a compressed image from an

$$F=\begin{bmatrix} f_4 & f_4 & f_4 & f_4 & f_4 & f_4 & f_4 & f_4 \\ f_1 & f_3 & f_5 & f_7 & -f_7 & -f_5 & -f_3 & -f_1 \\ f_2 & f_6 & -f_6 & -f_2 & -f_2 & -f_6 & -f_6 & f_2 \\ f_3 & -f_7 & -f_1 & -f_5 & f_5 & f_1 & f_7 & -f_3 \\ f_4 & -f_4 & -f_4 & f_4 & f_4 & -f_4 & -f_4 & f_4 \\ f_5 & -f_1 & f_7 & f_3 & -f_3 & -f_7 & f_1 & -f_5 \\ f_6 & -f_2 & f_2 & -f_6 & -f_6 & f_2 & -f_2 & f_6 \\ f_7 & -f_5 & f_3 & -f_1 & f_1 & -f_3 & f_5 & -f_7 \end{bmatrix} \quad Z=\begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} & z_{16} & z_{17} & z_{18} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} & z_{26} & z_{27} & z_{28} \\ z_{31} & z_{32} & z_{33} & z_{34} & z_{35} & z_{36} & z_{37} & z_{38} \\ z_{41} & z_{42} & z_{43} & z_{44} & z_{45} & z_{46} & z_{47} & z_{48} \\ z_{51} & z_{52} & z_{53} & z_{54} & z_{55} & z_{56} & z_{57} & z_{58} \\ z_{61} & z_{62} & z_{63} & z_{64} & z_{65} & z_{66} & z_{67} & z_{68} \\ z_{71} & z_{72} & z_{73} & z_{74} & z_{75} & z_{76} & z_{77} & z_{78} \\ z_{81} & z_{82} & z_{83} & z_{84} & z_{85} & z_{86} & z_{87} & z_{88} \end{bmatrix}$$

input image. Similarly, the process for decompression is the reverse of the compression process algorithm. The first compressed image pixel data is represented using equation (3.3) [77].

$$I_{11} = (f_4 * W_{11} + f_4 * W_{12} + f_4 * * W_{13} + f_4 * W_{14} + f_4 * W_{15} + f_4 * W_{16} + f_4 * W_{17} + f_4 * W_{18}) \quad (3.3)$$

where, $W_{11}, W_{12}, W_{13}, \ldots\ldots W_{18}$ is evaluated as follows:

$$W_{11} = (f_4 * Z_{11} + f_4 * Z_{21} + f_4 * * Z_{31} + f_4 * Z_{41} + f_4 * Z_{51} + f_4 * Z_{61} + f_4 * Z_{71} + f_4 * Z_{81}) \quad (3.4)$$

$$W_{12} = (f_4 * Z_{12} + f_4 * Z_{22} + f_4 * * Z_{32} + f_4 * Z_{42} + f_4 * Z_{52} + f_4 * Z_{62} + f_4 * Z_{72} + f_4 * Z_{82}) \quad (3.5)$$

Similarly,

$$W_{18} = (f_4 * Z_{18} + f_4 * Z_{28} + f_4 * * Z_{38} + f_4 * Z_{48} + f_4 * Z_{58} + f_4 * Z_{68} + f_4 * Z_{78} +$$

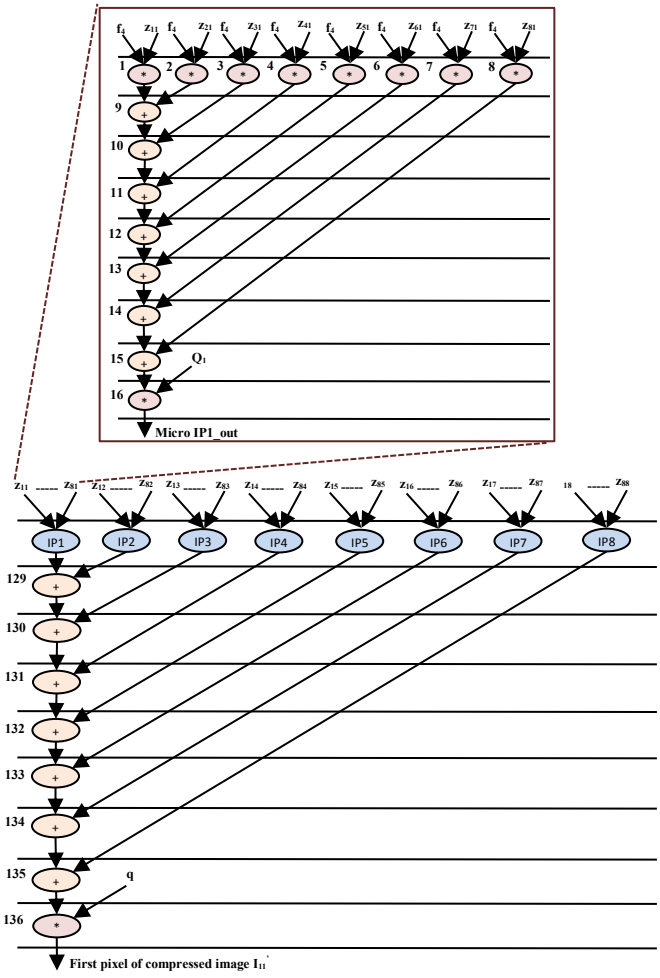


Fig.3.3. DFG of JPEG-CODEC for determining first pixel of the compressed image I_{11}'

$f_4 * Z_{88})$ (3.6)

Next, the remaining pixel value output is computed in a similar fashion. A control data flow graph corresponding to the untransformed JPEG image compression IP core is shown in Fig. 3.3. There are eight micro IP components (*namely from IP1 to IP8*) in Fig. 3. Moreover, Fig. 3.3 contains a zoomed image of one of the micro IP cores. Fig. 3.3 also highlights the quantization operation as each final generated output is multiplied with *'Q'*. JPEG algorithm uses different quantization matrixes to generate better results. Now, tree height transformation is applied to the initially generated DFG of the JPEG-CODEC IP core to obtain a structurally transformed DFG corresponding to the original one. This structural transformation (*i.e.,* THT) induces several interconnection-level changes while preserving the original functionality. As explained in the overview section, this produces a structurally different but functionally equivalent design. The final structurally transformed DFG of JPEG-CODEC IP core is shown in Fig. 3.4. Further, the obtained structurally transformed DFG is scheduled using the resource

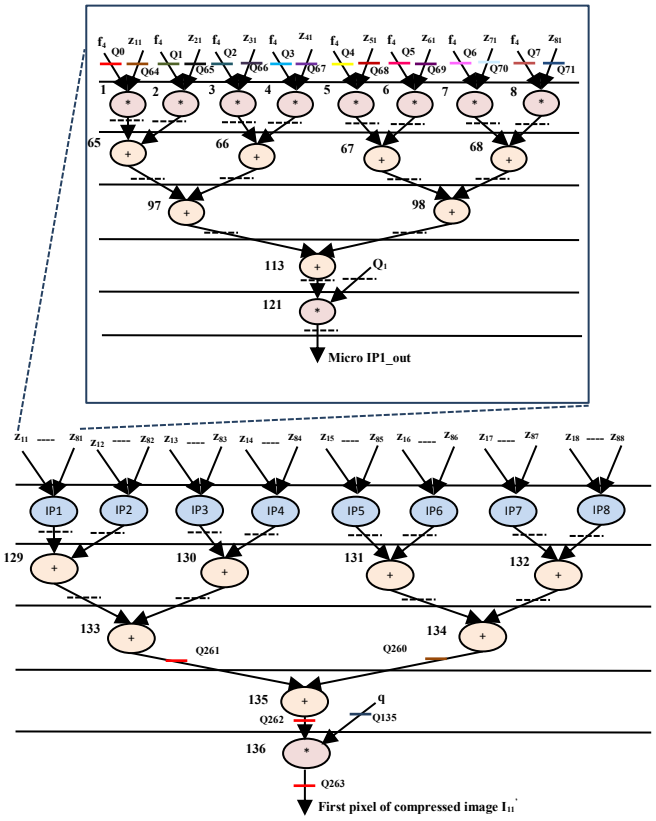


Fig. 3.4. DFG of structurally transformed JPEG-CODEC for determining first pixel of the compressed image I_{11}' with registers at input, output and intermediate storage points

configuration generated as output of firefly-based resource exploration (*for example,* three adders and three multipliers) and LIST scheduling algorithm. Post generation of scheduled DFG (SDFG), bitstreams corresponding to each IP vendor/designer specified encoding rules are generated using scheduling information (*i.e.,* control step and operation numbers in SDFG). These bitstreams are fed as input to key-driven crypto-chain-based security block to produce the final signature that is subsequently converted to hardware security constraints using IP vendor chosen embedding/mapping rules.

3.2.2. Low-cost secure architecture exploration using PSO-DSE

Fig. 3.5 highlights the details of PSO-DSE. The process of PSO-based architecture exploration is as follows: At first, the swarm population (*N*) and its dimensions (*d*), corresponding to resource types, are initialized (assuming *N=3* and *d=2* for adder and multiplier resource types). The first particle position *(P_1)* of the swarm is initialized with the maximum number of resource configurations (functional resources) possible corresponding to the image processing filter (taken from the module library). Similarly, the second particle position is initialized with the minimum number of functional resources possible. The third position is initialized with the average of the first and second particle position values. The initial velocity (V_i) corresponding to all particle positions is assumed to be zero. Next, the initial design cost (in terms of area and latency) is computed for all particles, and respective local best *(P_{lbi})* and global best *(P_{gb})* are evaluated. The particle having minimum design cost is termed as P_{gb}. The functions used for the calculation of design area *(A_{IP}),* design execution latency *(L),* and design cost (quality of results – QoR) are shown in equation numbers (3.7), (3.8), and (3.9), respectively.

$$Area\ (A_{IP}) = \sum_{i=1}^{2}\big(A(X_i) * (X_i)\big) \tag{3.7}$$

Where $A(X_i)$ indicates the area of a resource type (X_i) and (X_i) shows the number of instances utilized for a particular resource type.

$$\text{Latency (L)} = (C_M * L_M) + (C_A * L_A) \tag{3.8}$$

$$Design\ cost = w1 * \left(\frac{(A_{IP} - A_C)}{Amax}\right) + w2 * \left(\frac{(L - L_C)}{Lmax}\right) \tag{3.9}$$

Where *w1*=0.5 and *w2*=0.5 are designer-defined weighing factors that provide

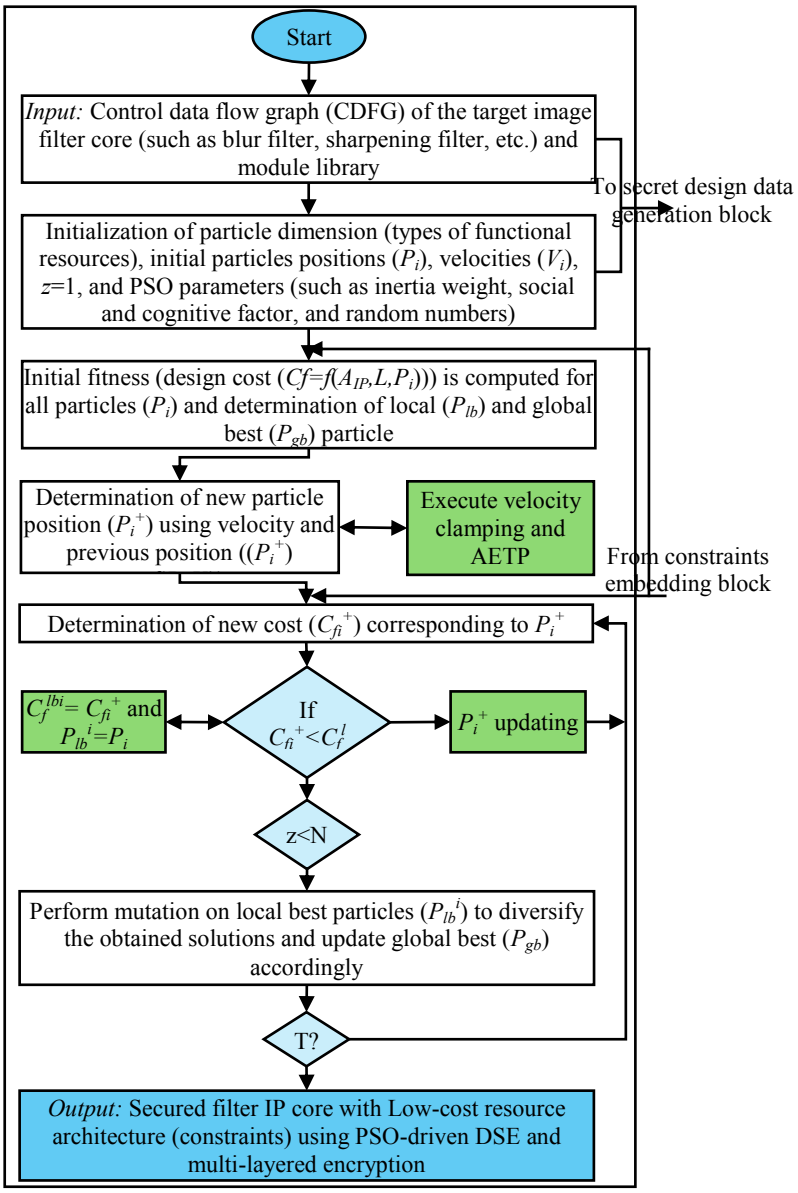


Fig. 3.5. Details of proposed PSO based DSE

equal weightage to design area (A_{IP}) and execution latency (L) during design cost function evaluation. *'C_M', 'C_A', 'L_M', and 'L_A'* are control steps required using multiplier, control step required using adder, latency of a multiplier, and latency of an adder, respectively. Further, A_{max} and L_{max} represents maximum design area and maximum latency. And A_C and L_C are IP vendor defined area and latency constraints.

Further, new particle positions (P_i^+) are determined based on computed new velocity (V_i^+ using inertia weight). The function used for computing new particle position and new velocity is shown in equations (3.10) and (3.11).

$$P_i^+ = P_i + V_i^+ \tag{3.10}$$

$$V_i^{+} = \omega V_i + f_1 k_1 [P_{lbi} - P_i] + f_2 k_2 [P_{gb} - P_i] \quad (3.11)$$

Where 'ω' is inertia weight, 'f_1 *and* f_2 ' are social and cognitive factors and 'k_1 *and* k_2 ' are random numbers.

Again, the design cost corresponding to all new particle positions is calculated. If the newly computed design cost is less than previously computed, then the respective local and global bests are updated. Next, velocity clamping and adaptive end terminal perturbation (AETP) are performed to keep computed velocity and particle positions in the desirable range. At last, the mutation is performed amongst particle positions to diversify during design space searching. After mutation, the design cost is computed again, and respective local and global bests are updated in case of lower design cost. The process is repeated until the terminating criterion (*T*) is not achieved. The algorithm gets terminated if either the algorithm does not show any further improvement till ten consecutive runs, or the max run limit (assuming *I*=50) is exhausted [78]. Fig. 3.2 illustrates the integration of PSO-based architecture exploration with multi-phase encryption to yield low-cost secured image processing filter IP core datapath. The pseudo-code of PSO based architecture exploration is as follows:

Input: N = Swarm size, *Max_ITR* = the maximum number of iterations, dim (d) = the number of dimensions, P_i = i^{th} particle in swarm, V_i = velocity of i^{th} particle, P_{gb} = global best particle, P_{lbi} = local best i^{th} particle, ω = inertia weight, f_1 and f_2 = social and cognitive factors, k_1 and k_2 = random numbers, P_i^{+} = new particle position, V_i^{+} = new velocity, AETP = adaptive end terminal perturbation, *minadder and minmulti* = the minimum number of adder and multiplier available in respective libraries of IP cores, *maxadder and maxmulti* = the maximum number of adder and multiplier available in respective libraries of IP cores, P_{Max} = particle position with the maximum number of adder and multipliers, P_{Min} = particle position with the minimum number of adder and multipliers.

Output: P_{gb} (global best particle).

FOR each particle P_i in N

FOR each dimension d in dim

$P_0 = P_{Max}$

$P_1 = P_{Min}$

P_2 = Average (P_{Max}, P_{Min})

FOR i in range (3, N)

$\dot{\alpha}_x = Rand$ (*minadder, maxadder*)

$\dot{\alpha}_y = Rand$ (*minmulti, maxmulti*)

IF (i%2 ==0):

$P_i = \ulcorner((minadder,\ maxadder)/2)\urcorner + \dot{\alpha}_x$, $\ulcorner((minmulti,\ maxmulti)/2)\urcorner - \dot{\alpha}_y$

ELSE:

$P_i = \ulcorner((minadder,\ maxadder)/2)\urcorner - \dot{\alpha}_x$, $\ulcorner((minmulti,\ maxmulti)/2)\urcorner + \dot{\alpha}_y$

END IF

END FOR

END FOR

$V_i = 0$ // initially

END FOR

Iteration $I = 1$

DO

FOR each particle P_i

Calculate design cost value according to equation

$$Design\ cost = t1 * \left(\frac{(A_{IP} - A_C)}{Amax}\right) + t2 * (\frac{(L - L_C)}{Lmax})$$

IF the current design cost value of P_i is lesser than P_{gb}

Set P_{gb} ← current P_i

END IF

END FOR

FOR each particle P_i

FOR each dimension *d* in dim

Calculate velocity (V_i) according to the equation

$$V_i^+ = \omega V_i + f_1 k_1 [P_{lbi} - P_i] + f_2 k_2 [P_{gb} - P_i]$$

Perform velocity clamping if required

Update P_i according to the equation below

$P_i^+ = P_i + V_i^+$ // Particle position updating

Perform AETP if required

END FOR

END FOR

Calculate design cost value for updated particles

FOR each particle P_i

IF the current design cost value of P_i is lesser than P_{gb}

Set P_{gb} ← current P_i

END IF

END FOR

FOR each particle P_i

Perform mutation on P_i

Perform AETP if required

END FOR

Calculate design cost value for updated mutated particles

FOR each particle P_i

IF the current design cost value of P_i is lesser than P_{gb}

Set P_{gb} ← current P_i

END IF

END FOR

$r = r+1$

WHILE Terminating condition is not achieved

Next, the pseudo-code of mutation used in PSO based architecture exploration is as follows:

Input: P_{lb} = local best architecture (resource)configuration

Output: New local best configuration after mutation, P_{lbi}

FOR each particle P_i in N

IF (i%2 ==0): //Left rotation

FOR t=1 to d

$temp = P_t$

$P_t = P_t + 1$

$P_t + 1 = temp$

t++

END FOR

END IF

IF (i%2 ==1):

FOR t=1 to d

$temp = P_t$

$P_t = P_t \pm R$ // R is a random number between [1,3]

t++

END FOR

END IF

i++

END FOR

Further, the pseudo-code of AETP operation used in PSO based architecture exploration is as follows:

Input: Architectural configuration violating design space boundary given in respective library files

Output: New architectural configuration within the design space boundary

// When P_i violates the design space boundary

WHILE (P_i<LW)

$P_i = P_i + J$

WHILE (P_i<UP)

$P_i = P_i - J$

/* J is a value between the minimum and maximum architectural configuration given in respective library files. "LW" and "UP" is lower and upper boundary limit (architectural configuration value) given in the respective library file of the image processing filter application*/.

The necessary and sufficient condition for the particle to eventually settle at the equilibrium, subsequently enabling the PSO algorithm to converge when the value of f_1 *and* f_2 (given in *eq.* 3.11)lies between the range of [1,4] and the value of inertia weight (ω) lies between [0.9,0.1]. This has been mathematically established by [78]. Further, the value of inertia weight must not be a constant value throughout the exploration process and must be linearly decreasing in every iteration throughout the exploration process. This has been empirically established in [78], [79]. In the context of the current problem, the population size of PSO for $p=3,5,7$ and terminating criterion (T = the algorithm runs for ten iterations if there is no improvement in the results or it will run for fifty iterations before termination) are sufficient for achieving optimal solution in an acceptable convergence and exploration time. This has been established in the literature [78], [79].

Advantage of population-based algorithm for solving multi-objective problem over single solution-based or hybrid algorithm: Since, the target problem in this chapter deals with generation of optimal secured image processing filter IP core datapath used in modern embedded systems, therefore it results in multitude of application mapping possibilities that exhibit high variance in performance metrics such as security, design area, and latency.

The architecture exploration process in the context of the problem requires optimized solutions for conflicting design objectives such as area vs latency vs security; where the optimized solution is in terms of resources needed for allocation, binding each task of the application to resources, and determining a feasible schedule for execution of tasks. This results into an intractable NP complete problem which is potentially suitable for resolution using population based algorithm like PSO. Population based heuristics like PSO are exploration oriented as they allow better diversification in whole search space compared to single solution based meta heuristics that only have the power to intensify the search in local regions. Single solution based heuristics such as simulated annealing algorithm requires extensive exploration time due to heavy number of iterations. Further, hybrid algorithms are also very computationally heavy as they often need to switch from one algorithm to another during the iterative process of architecture exploration.

3.2.3. Low-cost secure architecture exploration using FFA-DSE

The Firefly-driven design search exploration (FFA-DSE) equips the proposed methodology with the ability to prune undesirable designs (higher cost or lower fit) based on IP vendor-specified design objectives such as latency (delay) and area to explore the optimal secured design architecture of JPEG-

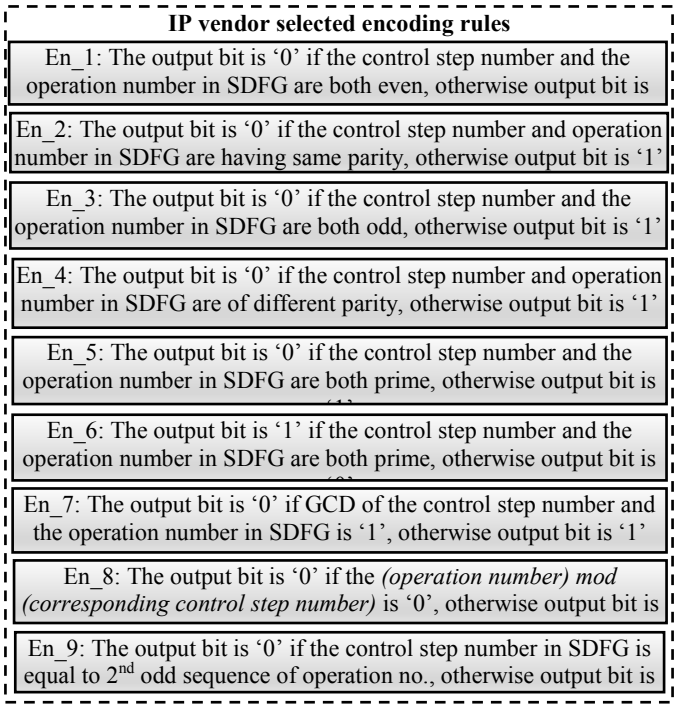


Fig. 3.6. Proposed IP vendor selected encoding rules

CODEC IP core. The IP vendor chosen encoding rules and the detailed flow of the proposed FFA-based architectural exploration is highlighted in Fig. 3.6 and 3.7. The main inputs to the proposed methodology are as follows: (a) absorption coefficient ('γ'), (b) terminating criterion, (c) firefly initialization parameter, (d) design constraints, (e) module library corresponding to JPEG-CODEC (containing necessary information such as DFG, area, and delay values), (f) step size control parameter ('α'), (g) design cost parameters, and (h) attractiveness parameter ('β').

Assuming that initially, the firefly population (Y) and its corresponding dimensions (d) are set at Y=3. A higher magnitude population size may also be assumed. The value of 'd' indicates the number of hardware resource types. For example, d = 2 when the number of hardware resource types comprises of adder and multiplier. The value of 'd' will change depending on the number of hardware resource types used for a given application. In the proposed approach, firstly, the design constraints for the area (A_{cons}) and latency (L_{cons}) are validated to be within the range of minimum and maximum value of area and time (latency) corresponding to JPEG-CODEC IP core (*i.e.,* $A_{min} < A_{cons} < A_{max}$ *and* $L_{min} < L_{cons} < L_{max}$). The initial firefly potions are initialized on meeting the valid design constraints requirement, as depicted in Fig. 2. The first firefly position (Y_1) is set with the maximum quantity of functional units. Likewise, the 2nd firefly position is set with minimum functional units. The 3rd firefly position is indicated with the average of the 1st and 2nd firefly positions. Next, positions for the rest of the fireflies are initialized based on the formula discussed in the FFA-based architecture exploration pseudo code (discussed below in this sub-section). After the initialization of the firefly positions, an initial design cost (fitness value) is evaluated corresponding to each firefly position using the design cost function (with respect to area and latency). *Note:* We have considered area and latency specifications parameters for the evaluation of design cost in our proposed approach. After determining the initial design cost, the respective local and global best positions are updated. The local best (Y_{lbi}) positions are the initial positions of fireflies, and the initial global best (Y_{gb}) position is the firefly position with the minimum design cost value (fittest solution) among all. The equations for the determination of area,

latency, and design cost are same as that of PSO-DSE (described in (3.7),

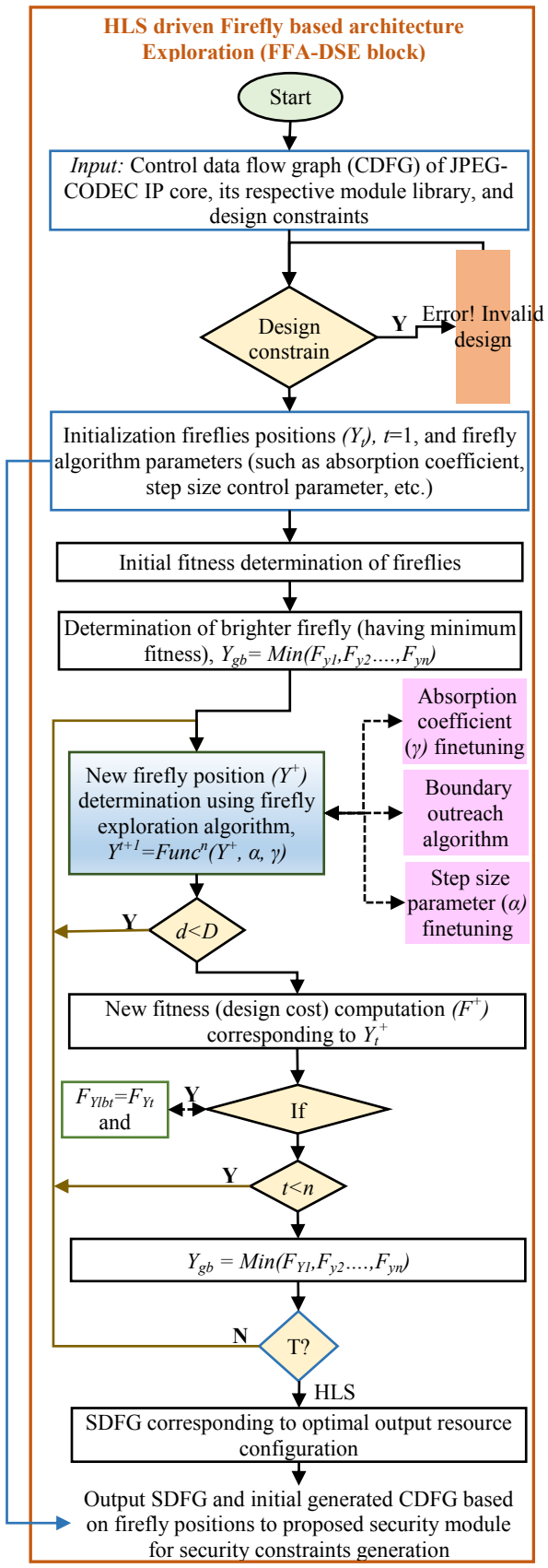


Fig. 3.7. Detailed flow diagram of the FFA-DSE algorithm

(3.8), and (3.9)).

Further, new firefly positions (Y^{t+1}) are computed using the absorption coefficient (γ), control step size parameter (α), and attractiveness parameter (β). If the newly generated firefly position exceeds the boundary limit (i.e., minimum and maximum resource value corresponding to each resource type). In that case, the boundary outreach algorithm is executed to bring that particular firefly position within acceptable limits. The new firefly positions are evaluated using equations (3.12), (3.13), and (3.14), respectively [80], [82], [94].

$$Y_i^{t+1} = Y_i^t + (\beta(Y_j - Y_i) + \alpha(rand - 1/2) \tag{3.12}$$

Here, new firefly positions are determined by adding a drift factor to the original firefly position.

$$\beta = \beta_0 e^{-\gamma Y^2_{ij}} \tag{3.13}$$

$$Y_{ij} = |Y_i - Y_j| = \sqrt{\sum_{r=1}^{d} (Y_{i,r} - Y_{j,r})^2} \tag{3.14}$$

where $'Y_i^{t+1}$ is the new firefly position, $'Y_i^t'$ is the previous firefly position, $'Y_i'$ and $'Y_j'$ are positions of $'i^{th}'$ and $'j^{th}'$ firefly, respectively. $'\gamma'$, $'\alpha'$ and $'\beta'$ are above-defined hyperparameters (tuning parameters). $'\beta_0'$ is attractiveness at zero distance and $'Y_{ij}'$ is the cartesian distance between $'Y_i'$ and $'Y_j'$ fireflies.

After determining new firefly positions, the new design cost corresponding to each firefly position is computed. Here, the boundary outreach algorithm (BOA) is executed if the generated new firefly positions violate boundary limits. The Pseudo code of BOA is explained below. Post-design cost computation, if the new design cost is lesser than the previously computed design cost for any firefly position, the local best corresponding to all such fireflies is updated. And, again, the firefly with minimum design cost is declared the global best firefly. The complete process is executed till the terminating criterion is not satisfied. The terminating criterion (T) for our proposed work is that the algorithm will either run until there is no improvement in design cost till fifteen iterations or run for a maximum of fifty iterations [80]. Finally, an optimal architecture configuration corresponding to

JPEG-CODEC IP core is obtained as the output. The obtained optimal resource configuration is forwarded to security constraints embedding block for further embedding of generated security constraints to generate a secure JPEG-CODEC hardware IP design.

Parameter Tuning for customizing FFA to solve DSE in HLS: The necessary and sufficient conditions (or values of different hyperparameters) for the fireflies to ultimately settle at the equipoise, consequently empowering the firefly algorithm to converge, are adopted from [80], [82], [94]. The value of *'β_0', 'γ', 'α_x', 'α_y', and* rand are 1, 0.5, maximum value of the first dimension, maximum value of the second dimension, and 1.5, respectively [80], [82], [94]. Moreover, the value of the absorption coefficient ('γ') and control step size parameter ('α') must not be kept as a constant value. They should be linearly decreasing in nature, as discussed above in this sub-section and established in [80]. The pseudo-code of FFA based architecture exploration is as follows:

Input: Y = firefly population size, dim (d) = total dimensions (*i.e.,* #resources), $Y_k = k^{th}$ particle in firefly population, Y_{gb} = global best firefly position, Y_{lbk} = local best k^{th} firefly, Y_k^{t+1} = new firefly position, BOA = boundary outreach algorithm, T = terminating criterion, *least_add and least_mult* = the minimum (least) quantity of resources (*i.e.,* adder and multiplier) present in the library of JPEG-CODEC application, *highest_add and highest_multi* = the maximum (highest) quantity of resources present in the library of JPEG-CODEC application, Y_{Max} = firefly position with the highest quantity of resources, Y_{Min} = firefly position with the least number of resources, β = attractiveness parameter, γ = absorption coefficient, and α = control step size parameter.

Output: Y_{gb} (global best firefly position).

FOR each firefly position Y_k in Y

FOR every dimension d in dim

$Y_0 = Y_{Max}$

$Y_1 = Y_{Min}$

Y_2 = Average (Y_{Max}, Y_{Min})

FOR k in range (3, N)

u = *Rand* (*least_add, highest_add*)

v = *Rand* (*least_multi, highest_multi*)

IF (k % 2 ==0):

Y_k = ⌈((*least_add, highest_add*)/2)⌉ + u , ⌈((*least_multi, highest_multi*)/2)⌉ - v

ELSE:

Y_k = ⌈((*least_add, highest_add*)/2)⌉ - u , ⌈((*least_multi, highest_multi*)/2)⌉ + v

END

END

END

END

Iteration p = *1*

DO

FOR each firefly Y_k

Compute the design cost value as per the equation below:

$$Design\ cost\ (F_{Yk}) = t1 * \left(\frac{(A_{IP} - A_C)}{Amax}\right) + t2 * (\frac{(L - L_C)}{Lmax})$$

IF the present design cost ($Y_k < Y_{gb}$)

Set Y_{gb} ← current firefly Y_k

END

END

FOR every firefly Y_i

FOR every dimension d in dim

Compute new firefly positions as per the equation below:

$$Y_i^{t+1} = Y_i^t + (\beta(Y_j - Y_i) + \alpha(rand - 1/2)$$

Update firefly positions accordingly

Execute BOA if essential

END

END

Design cost computation corresponding to newly obtained fireflies

FOR each firefly Y_k

IF the present design cost $(Y_k < Y_{gb})$

Set $Y_{gb} \leftarrow$ present Y_k

END

END

$p = p+1$

WHILE stopping criterion is not met

Further, the boundary outreach algorithm (BOA)'s pseudo-code is as follows:

Input: Resource configuration exceeding design space periphery as specified in the library file

Output: Modified architecture (resource) configuration within the periphery of the design space

// When Y_i exceeds the design space periphery

WHILE (Y_i<*Low_Lim*)

$Y_i = Y_i + H$

WHILE (Y_i<*Up_Lim*)

$Y_i = P_i - H$

/* H specifies a magnitude between the least and highest resource configuration in the library file. "*Low_Lim*" and "*Up_Lim*" is lower and upper boundary limit (resource configuration) specified in JPEG-CODEC's library file */.

3.2.4. Advantages of employing PSO-DSE and FFA-DSE for low-cost secure architecture exploration

Heuristic based architecture exploration has been employed to perform design space pruning based on area-latency tradeoff. The heuristic employed in our work is particle swarm optimization (PSO) and firefly algorithm (FFA) for architecture exploration. It is performed to determine a low-cost resource architecture among numerous potential competitive designs.

Particle swarm optimization shows more benefits as compared to other metaheuristics algorithms such as genetic algorithm (GA) [95], [96], bacterial foraging algorithm (BFOA) [97], ant colony optimization algorithm (ACO) [98], etc. PSO depicts the ability to achieve a global optimal solution in an acceptable amount of time (fewer iterations) and provides a clinical balance between exploration and exploitation time compared to GA, BFOA, ACO, etc. [78]. The PSO algorithm's implementational complexity is lesser than GA, BFOA, ACO, etc. [78]. It is tough to achieve the best solution using a genetic algorithm because of its premature convergence problem [99]. GA takes a higher number of iterations in case of higher variables and constraints [100]. The time complexity of GA is higher as compared to the PSO algorithm [100]. Further, the bacterial foraging optimization algorithm also has the drawback of getting stuck into the local optimum because of weak connections among bacteria. BFO provides a poor balance between exploration and exploitation time because of its fixed step size [97]. Next, the ant colony optimization algorithm has the limitation of falling into the local optimum trap [101]. Additionally, the involvement of pheromone laying activity (which is further used by ants as a communication medium) increases the implementation complexity of the ACO algorithm [98]. The PSO algorithm comprises various hyperparameters (also known as tuning parameters), which provide a clinical balance between exploration and exploitation tradeoff, which is missing in the case of GA, BFOA, ACO, etc.[79]. Further, the reasons for employing PSO

over other heuristics in the context of the design space exploration problem addressed in the chpater are the following: (a) it incorporates a parameter named inertia weight, which considers the magnitude of the previously computed velocity and supports in escaping local minima, (b) it provides a clinical balance between exploration and exploitation process with the help of linearly decreasing value of inertia weight from 0.9 to 0.1 (bigger steps at the beginning and smaller later) [78], and (c) it includes various other hyperparameters such as social and cognitive factor, which reduces the convergence time of reaching the optimal/ near-optimal solutions. All of these features are not present in [96], [97], and [101]. Therefore, due to its inherent capability of escaping local minima and achieving a fitter solution (design cost in terms of area and latency) at lower convergence time, PSO-based architecture exploration is used for design space searching of secured low-cost image processing filter IP core.

Advantages of integrating PSO-driven DSE:

a. The advantages of PSO-DSE over different other design space exploration algorithms (such as genetic, bacterial foraging algorithm, etc.) are as follows: PSO-DSE considers the magnitude of the previously computed velocity with the help of a parameter called inertia weight, while genetic algorithm-driven DSE (GA-DSE) [95], [96] and bacterial foraging-driven DSE (BFO-DSE) [97] do not consider the momentum of prior iterations, which increases the probability of getting stuck in the local minima during architecture exploration.

b. PSO-DSE creates a balance between exploitation and exploration time with the help of linearly decreasing the value of inertia from 0.9 to 0.1. The algorithm takes more significant steps at the beginning and smaller steps on reaching higher fitness solutions, which is missing in GA-driven DSE and BFO-driven DSE. This also enhances the chance of reaching global optimal solution.

c. The inclusion of various other factors (hyperparameters), such as social and cognitive factors in PSO-DSE, helps achieve higher fitness

solution within a very low exploration time. On the contrary, there is no such provision in GS-DSE and BFO-DSE.

Key benefits of employing FFA for performing DSE in HLS:

a. FFA comprises of various hyperparameters such as step-size control parameter and absorption coefficient. These parameters control randomness throughout design search exploration, ultimately resulting in better convergence time to the optimal solution. The value for both parameters linearly decreases from larger to smaller as the DSE algorithm proceeds. This is because the distance between the current and global best firefly position is large in the initial exploration phase. Therefore, larger steps are required in the initial phases, and step size decreases upon nearing the global optimal solution.

b. FFA works similar to the divide and conquers approach. FFA-DSE is based on the attraction parameter, where attractiveness is inversely proportional to the distance between fireflies. This enables FFA to divide its population into subgroups, where different subgroup swarms around different local optimums, and a final optimal solution are obtained among them.

c. The linearly decreasing value of the step size control and absorption coefficient parameters enables FFA to maintain steady stability between diversification (exploration) and intensification (exploitation).

Therefore, because of the ability to escape local optimum to attain a global optimal solution in lesser iterations (or at lower convergence time), FFA is employed for DSE of key-driven crypto-chain based secured JPEG-CODEC IP core. The methodology of FFA-based hardware resource exploration is explained subsequently.

3.2.5. Multi-phase encryption-based security for image processing filter IP cores

Goal: The multi-phase encryption algorithm is applied to the extracted secret information from the initial RAT to generate the final encrypted signature, which is further used to generate hardware security constraints. The steps

involved in the multi-phase encryption-based hardware security algorithm are described below.

Details of the algorithm: First, all storage variable pairs (R_x, R_y) allocated on the same-colored register are listed using Table 3.1 (with black and indigo-colored storage variables). Next, all the indices values (x, y) of the storage variable pairs (of the initial RAT) are converted into their hexadecimal equivalents. The final generated indices value in hexadecimal equivalents is the secret information used for encryption. Fig. 3.8 depicts the complete multi-phase encryption algorithm in detail. The first step of multi-level encryption is to generate an initial state matrix with the help of secret information based on the IP vendor selected encryption key (K_1). A particular row of the state matrix can contain a maximum of four elements. The initial state matrix formation depends on a two-bit key value, as highlighted in Fig. 3.8. In the next phase, the bit manipulated state matrix is generated using AES forward S-box. All elements of the initial state matrix are substituted with their AES forward S-box equivalents. Next, row-diffusion is performed based on the IP vendor selected encryption key (K_2) in the subsequent phase. Each row has a different key for row diffusion in the bit-manipulated state matrix. Therefore,

Table 3.1: Register allocation table before and after embedding hardware security constraints corresponding to sharpening filter

	C0	C1	C2	C3	C4	C5	C6
Red(R)	R0	R22	R22	R32	R34	R35	R36
Green (G)	R1	R23	R23	R33	R39	R40	R41
Indigo (I)	R2	R24	R24	R37	-	-	-
Blue (BL)	R3	R25	R25/ R28	R38	R34	-	R36
Yellow (Y)	R4	R26	R26/ R27	R26	R26	R26	-
Black (B)	R5	R26	R27/ R26	R26	R26	R26	-
Violet (V)	R6	R25	R28/ R25	-	-	-	-
Pink (P)	R7	-	R29	-	-	-	-
Lime (LI)	R13	R13	R13	R13	R13	-	-
Olive (O)	R8	R8	R30	-	-	-	-
Aqua (A)	R9	R31	R31	R31	R31	R31	-
Teal (T)	R10	R10	-	-	-	-	-
Gray (G)	R11	R11	-	-	-	-	-
Maroon (M)	R12	R12	-	-	-	-	-
Silver (S)	R14	R14	-	-	-	-	-
Khaki (K)	R15	R15	-	-	-	-	-
Lavender (L)	R16	R16	-	-	-	-	-
Crimson (C)	R17	R17	-	-	-	-	-
Wheat (W)	R19	R19	-	-	-	-	-
Beige (B)	R18	R18	R18	R18	R18	-	-
Magenta (M)	R20	-	-	-	-	-	-
Orange (O)	R21	R24	R24	-	-	-	-

row-diffusion depends on the 2*N bits encryption key (K_2) (where N is the # rows in the state matrix). Subsequently, in the next phase, digit equivalents corresponding to unique alphabets (A-F) are computed using TRIFID cipher (using IP vendor selected encryption keys) and alphabetic substitution (using IP vendor selected encryption key (K_3)). The three properties of the TRIFID cipher are fractionation, substitution, and transposition, which help to impart confusion and diffusion in the multi-phase encryption algorithm. For computing the TRIFID cipher, first IP vendor selects a 27-bit unique key value for all unique alphabets in the state matrix. Then, the key value is arranged in a three-square matrix of size 3*3 each. The output of the TRIFID cipher is a state "*tuv*" (where *'t'* is row number, *'u'* is column number, and *'v'* is the matrix number corresponding to the respective alphabet). The determined state value ("*tuv*") is fed as input to the alphabetic substitution phase to finally generate digit equivalence based on the IP vendor selected encryption key (K_3). E*.g.,* TRIFID cipher computation and its digit equivalent corresponding to the alphabet 'A' is explained below.

Let the IP vendor selected key for the alphabet ‘A’ = EDRFTV$QA WSZMXNCBGYHUJIKOLP.

Table 3.2 depicts a square matrix representation of the IP vendor selected key for TRIFID cipher computation corresponding to alphabet A. Here, row number (t) is 3, column number (u) is 3, and square matrix (v) number is 1. So, the state corresponding to "A" is **331.** Now, let the assumed IP vendor selected key for alphabetic substitution corresponding to the alphabet 'A' is "100". From the alphabetic substitution rules in Fig. 3.8, the calculated digit equivalent corresponding to the alphabet 'A' is '6'. Similarly, alphabetic substitution is performed for all remaining alphabets using TRIFID cipher computation and IP vendor selected encryption key (K_3). After determining digit equivalence, all alphabet values in the state matrix are substituted with their digit equivalents. Further, the obtained state matrix is transposed. Then,

Table 3.2: Square matrix representation of the key for TRIFID cipher computation corresponding to alphabet A

Square matrix 1			Square matrix 2			Square matrix 3		
E	D	R	W	S	Z	Y	H	U
F	T	V	M	X	N	J	I	K
$	Q	**A**	C	B	G	O	L	B

all elements are concatenated to generate an encrypted byte sequence using the byte concatenation rule based on the IP vendor selected encryption key (K_4). Finally, all elements are converted into binary equivalents to generate the final encrypted signature and truncated based on the IP vendor selected truncation length. The generated encrypted signature is further used to generate hardware security constraints based on the IP vendor selected encoding rule (if the

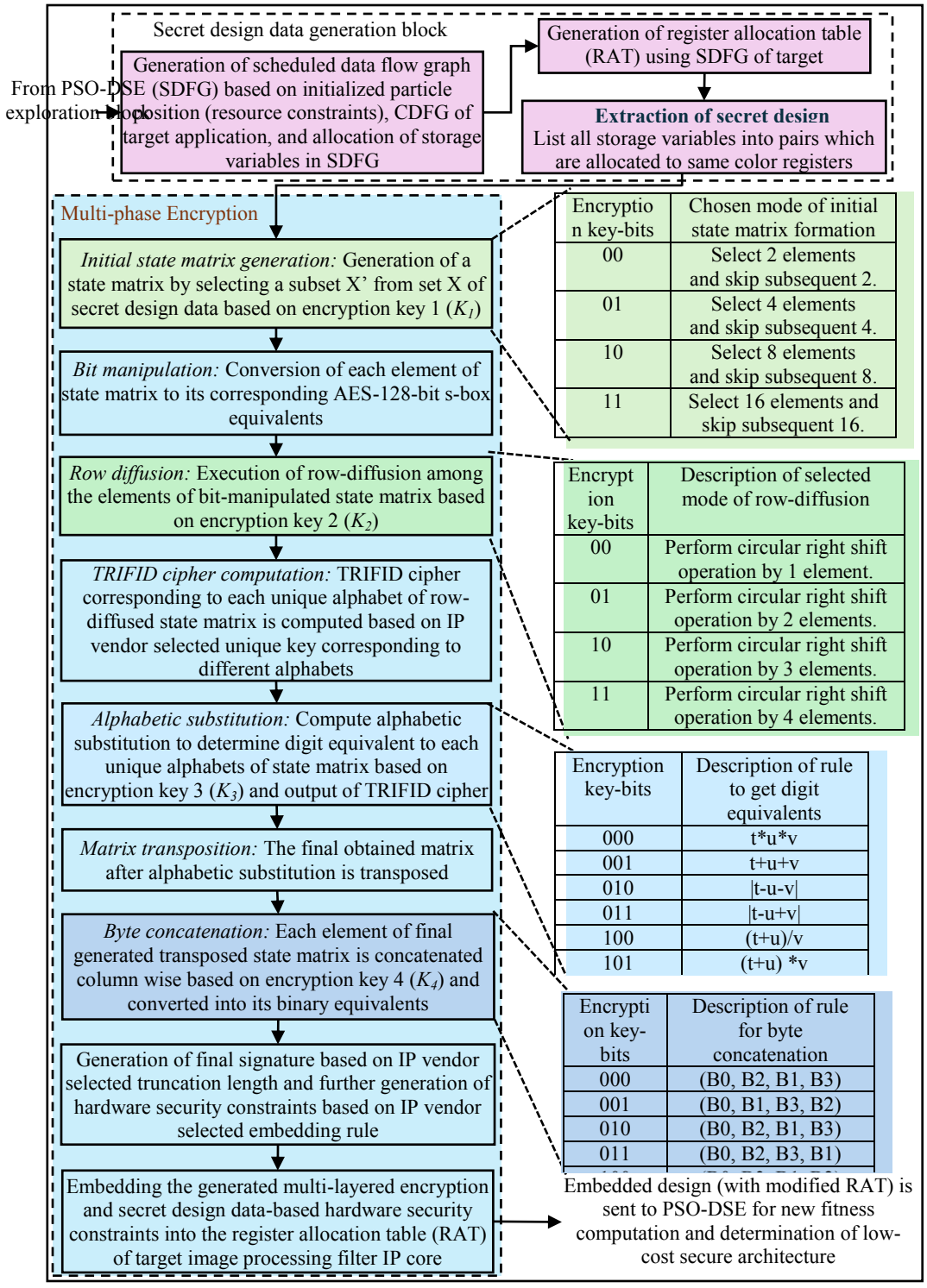


Fig. 3.8. Details of multi-phase encryption-based hardware security methodology

signature bit is '0', then implant an artificial edge between (even, even) storage variable pair, otherwise embed an artificial edge between (odd, odd) storage variable pair). The artificial edges embedded between the storage variable pairs of the RAT (corresponding to the image processing filter application) indicate the covert hardware security constraints implanted into the design. An artificial edge implanted between the storage variables of the design signifies that the corresponding storage variables cannot be assigned to the same register (*i.e.,* forced distinct register assignment is made).

3.2.6. Key-driven crypto-chain-based hardware security methodology

The primary goal of the proposed key-driven crypto-chain-based hardware security methodology is to produce secret hardware security constraints using the scheduling information of the JPEG-CODEC, IP vendor specified encoding rules, and IP vendor chosen crypto-keys. The primary inputs to this security block are (a) IP vendor specified encoding rules, (b) IP vendor specified keys to drive crypto-chain based security methodology, (c) IP vendor specified bit padding and embedding rules, (d) IP vendor specified truncation length, and (e) scheduling information obtained through transformed JPEG-CODEC SDFG. Next, the obtained hardware security constraints are covertly inserted into the design of the JPEG-CODEC using the HLS framework. The presence of embedded security constraints in the design provides immunity against IP piracy and fraudulent claim of IP ownership problems. Fig. 3.9 illustrates the proposed key-driven crypto-chain based security methodology and its integration with FFA. The various steps involved in the generation of secret security constraints are as follows:

Generation of initial bitstreams based on IP vendor specific encoding mechanism: Initially, a bitstream is generated using scheduling information of the JPEG-CODEC (SDFG), and IP vendor specified encoding mechanisms. The scheduling information of JPEG-CODEC is highlighted in Table 3.3. Further, the IP vendor specified encoding rules are shown in Fig. 3.6. The encoding rules used to generate the initial bitstreams are decided by the authentic IP vendor/designer, thus remaining completely unknown to an attacker. There are 136 *(l)* operations in the SDFG of the JPEG-CODEC, as depicted in Table 3.3. These are scheduled among different control steps using

resource constraints information (*for example*, scheduled using three adders and three multipliers). For the sake of demonstration, the scheduling information is taken from the SDFG of JPEG-CODEC that is scheduled using three adders and three multipliers. However, in each iteration of the proposed FFA-based security approach, the scheduling is performed using obtained

Control step	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Operation allocated to Mul_1	1	4	7	18	21	24	35	38	49	52	55	66	69	72	83	86	97	100	103	114	117	120	48	96	-	-	-	-
Operation allocated to Mul_2	2	5	8	19	22	33	36	39	50	53	56	67	70	81	84	87	98	101	104	115	118	16	64	112	-	-	-	-
Operation allocated to Mul_3	3	6	17	20	23	34	37	40	51	54	65	68	71	82	85	88	99	102	113	116	119	32	80	-	-	128	-	-
Operation allocated to Add_1	-	9	10	12	25	27	28	42	43	57	58	60	73	75	76	90	91	105	106	108	121	123	124	126	127	-	132	-
Operation allocated to Add_2	-	-	11	13	26	29	41	30	44	46	59	61	74	77	89	78	92	94	107	109	122	125	129	130	131	-	-	134
Operation allocated to Add_3	-	-	-	-	14	15	-	-	45	31	47	-	62	63	-	-	93	79	95	-	110	111	-	-	133	-	-	-

: Scheduling information of JPEG-CODEC based on its SDFG scheduled using three adders and three multipliers

architectural configuration through FFA-driven DSE. That particular scheduling information is used to generate the initial bitstream. Therefore, these initial bitstream changes in each iteration for each architectural configuration. Moreover, for the sake of brevity and simplicity during the explanation of the proposed approach, we have assumed nine IP vendor specified rules *(e),* as shown in Fig. 3.6. (*Note:* the value of *'e=9'* is generally very large as the IP vendor can devise potentially innumerable encoding algorithms). Therefore, an initial *l-bit* bitstream is generated corresponding to the nine different encoding rules (blocks). For example, as shown in Fig. 3.6, encoding 1 (En_1): the output bit is 0 if the control step number and the operation number in SDFG are both even; otherwise, output bit is '1'. Therefore, if we compare the first operation number with its control step number (*i.e.,* also one as shown in Table 3.3), the output bit is *'1'.* Similarly, the output for all remaining operations of the JPEG-CODEC application is computed.

Conversion of the initially generated bitstream into 1024 bits and details of the proposed crypto-chain algorithm*:* The proposed key-driven crypto-chain based security methodology uses *'2k'* hash slices (each hash slice comprises SHA-512 based crypto-chain module for the generation of the encrypted bitstream). The SHA-512 accepts input in the form of 1024 bits. Therefore, the initial *l-bit* bitstream is converted into a 1024-bit bitstream using an IP vendor specified initial pre-processing technique. The initial pre-processing technique used in the proposed security methodology is as follows: first, *l-bits* are taken as input and converted to 896 bits by performing bit stuffing after *l-bits.* The 896 bits are generated post appending the initial *l-bits* with '1', followed by the continuation of '0' till 896 bits. Further, the length of the initial bitstream (*i.e., l*) is converted into its 128-bit representation (for this, first convert '*l'* into binary and append continuation of '0' till 128-bit before binary bitstream). Finally, these 128 bits are appended to the 896 bits to generate 1024 bits, which is fed as input bitstream to the first hash slice of crypto-chain based security methodology. The inclusion of nine different IP vendor-specific encoding rules in the proposed hardware security methodology increases the robustness of the proposed security methodology. Each encoding mechanism

generates a unique *l-bit* initial bitstream data, which is fed as input to the different hash slices of the proposed security methodology. This uniqueness in the generation of different initial bitstreams increases the robustness of the proposed security methodology.

***Crypto-chain**:* The proposed security methodology comprises several hash slices connected as a chain in a cascaded way. The output of one hash slice becomes the input of its subsequent hash slice. The cascading arrangement of hash slices is illustrated in Fig. 3.8. The first *'k'* (in this chapter, *k* is nine) hash slices are the primary crypto components that accept the encoding blocks generated bitstreams as input after performing IP vendor-specific pre-

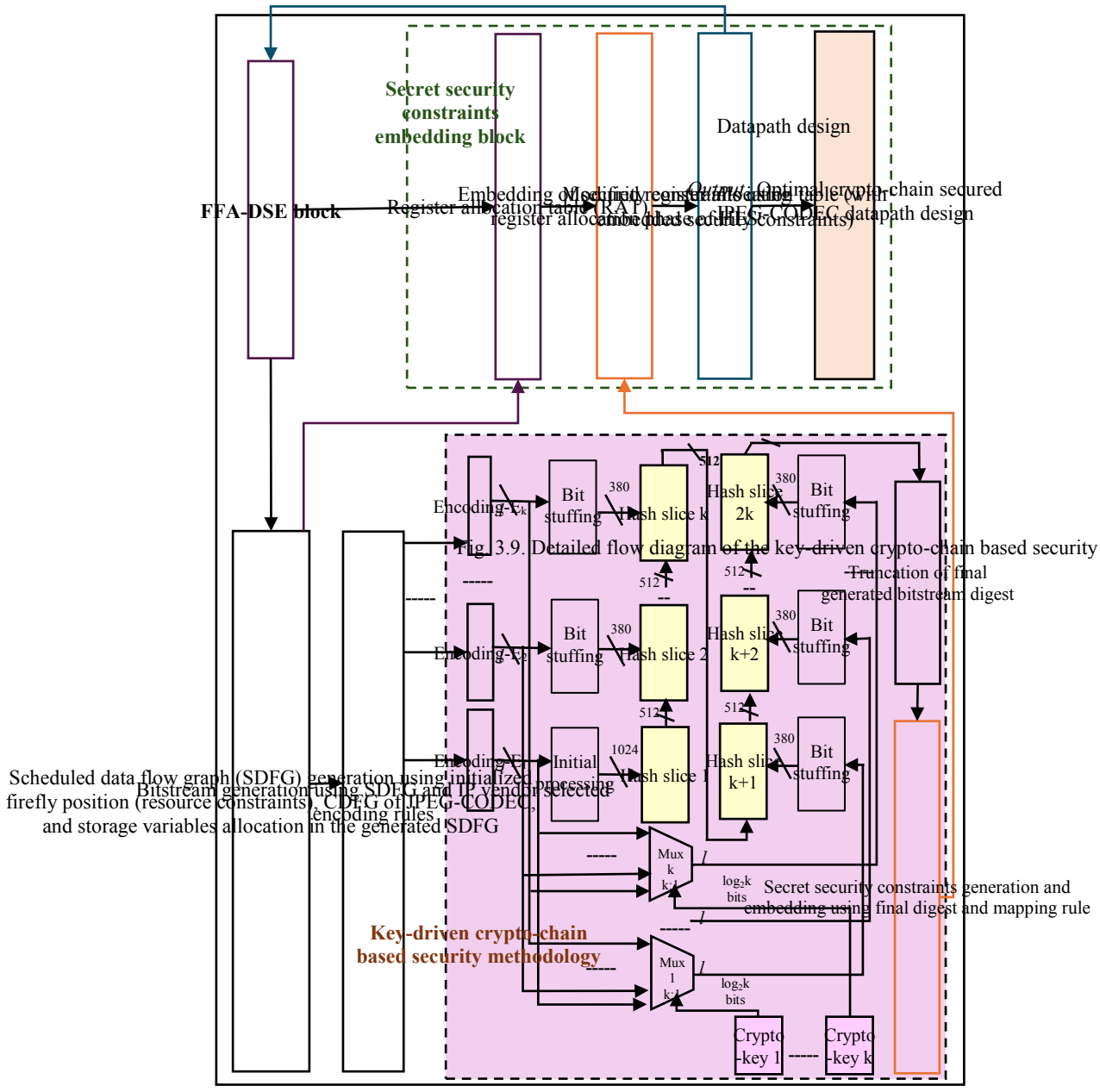


Fig. 3.9. Detailed flow diagram of the key-driven crypto-chain based security Methodology

processing. However, the remaining *'k'* hash slices are controlled using multiplexers (IP vendor specified crypto keys are used to control the output of multiplexers).

***Bit stuffing**:* The method of generating 1024 bits from *l-bits (i.e.,* 136 for JPEG-CODEC*)* for the first hash slice is already discussed above in this sub-section. Next, the 136-bit output of the remaining encoding blocks (*i.e.,* except the first encoding block) is converted into 380 bits. For this, (380-*l*) bits of continuous '0' is stuffed as a suffix of each *l-bit* output of the remaining output block. All these bit-appending and stuffing algorithms are only known to the IP vendor/designer, thus making it challenging for an adversary to regenerate the exact output as the proposed security methodology.

Further, the method of generating 1024 bits (which acts as input to the second hash slice) from 512 bits of the previous hash slice and 380 bits of primary bitstream input (after pre-processing) is as follows: 512-bit output of the previous hash slice is appended with "1000" followed by 380 bits (generated using bit stuffing) to generate 896-bits. Finally, the length of the output of the previous hash slice, *i.e.,* 512, is converted into its 128-bit representation. This conversion is done in a similar way as explained earlier in this sub-section, and the obtained 128 bits are appended after 896 bits to generate 1024 bits. Similarly, the output of each of the hash slices (512 bits) is converted into 1024 bits using 380 bits (obtained after pre-processing of input *l-bit* bitstream) and 128 bits (generated using the length of the previous hash slice output, *i.e.,* 512). The output of $(k\text{-}1)^{th}$ hash slice becomes the input of the k^{th} hash slice.

***Hash slice**:* There are total *'2k'* hash slices in the proposed hardware security methodology. Each hash slice is executed only once to generate the encrypted 512 bits as output which becomes the input to its subsequent hash slice. Further, the round function within each hash slice (SHA-512) is executed as per IP vendor specified iterations. The input to the first *k* hash slices is the pre-processed output of encoding blocks. However, the input for the remaining *'k'* hash slices is controlled using multiplexers, as shown in Fig. 3.8. For *'k'* encoding blocks, *'k'* multiplexers are required. The input of additional *'k'* hash slices is decided using IP vendor specific crypto keys. The maximum possible

hash slices for the *k* number of encoding blocks are *2k* in our proposed approach.

Further, the final generated 512-bit output is converted into its equivalent hardware security constraints using IP vendor specified mapping/embedding rules (discussed in the following sub-section). Finally, the generated hardware security constraints are embedded into the design of the JPEG-CODEC IP core during the register allocation phase of the HLS framework. Subsequently, FFA-DSE explores the optimal resource configuration against secured hardware IP design with the help of design cost function.

3.3. Illustrative Example: Watermark (Signature) Embedding Process

The proposed multi-phase encryption based methodology is demonstrated on the sharpening filter (SF) application. The SDFG of the SF application scheduled with four adders and two multipliers (obtained through heuristic-based architecture exploration) is illustrated in Fig. 3.10. Further, Table 3.1 depicts the initial RAT (with black and indigo colored storage variables) corresponding to the scheduled SF. The different steps involved in the demonstration with their corresponding outputs are shown in Fig. 3.11. The secret information extracted from Table 3.1 and its hexadecimal equivalents are shown in Figures 3.11. (a) and (b), respectively. An initial state matrix is generated using IP vendor selected key value 1 (K_1) (assumed "K_1=01" for demonstration). The generated initial state matrix is shown in Fig. 3.11. (c). Next, the bit manipulated state matrix generated using AES forward substitution box (S-box) is depicted in Fig. 3.11. (d). Then, the bit-manipulated state matrix is subjected to row-diffusion based on IP vendor selected key 2 (K_2) (assumed "K_2=10 00 11 01 11"). Fig. 3.11. (e) depicts a row-diffused state matrix. Next, the alphabetic substitution of the state matrix is shown in Fig. 3.11. (f). Further, the transposed state matrix is shown in Fig. 3.11. (g). Finally, all generated transposed state matrix elements are concatenated based on IP vendor byte concatenation key 4 (K_4) (assumed "K_4=010 101 000 100 011"). The obtained byte concatenated string is shown in Fig. 3.11. (h). Subsequently, the final generated signature and its respective hardware security constraints

are depicted in Figures 3.11. (i) and (j). Further, the generated hardware security constraints are embedded into the design. Two storage variable pairs cannot be allocated to the same register while embedding the security constraints in the RAT. Therefore, either the registers (colors) are swapped, or a new register is added to accommodate the storage variable artificial edges. Table 3.1 also depicts the final RAT (modified locations with red colored storage variables) with embedded security constraints (as obtained in Fig. 3.11. (j)) generated using low-cost multi-phase encryption.

Next, Fig. 3.12 demonstrates the security constraints generation and embedding flow of the proposed key-driven crypto-chain based security methodology. As

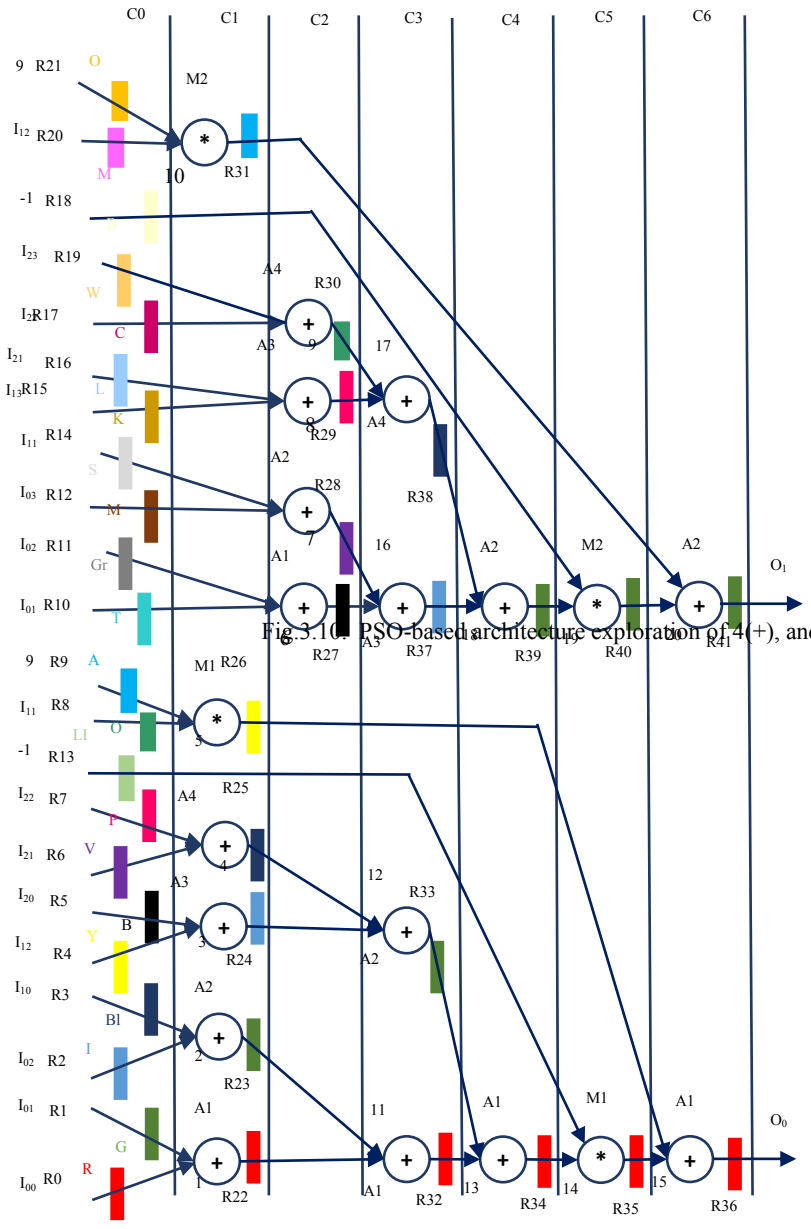


Fig.3.10. PSO-based architecture exploration of 4(+), and 2(*) resources used for schedul

shown in Figures 3.12. (a) and (b), the scheduled DFG of JPEG-CODEC is initially taken as input, and bitstreams are generated corresponding to different IP vendor selected encoding rules using scheduling information. Further, Fig. 3.12. (c) highlights the generated bitstreams corresponding to different IP vendor selected encoding rules (or encoding blocks such as En_1 to En_9). Each encoding block generates 136 bits value as JPEG-CODEC comprises 136 operations (shown in Table 3.3). Next, the generated 136 bits corresponding to the first encoding block (*i.e.,* En_1) are passed through the pre-processing step to generate 1024 bits. The steps to generate 1024 bits from 136 bits are discussed in the previous subsection. The final generated 1024 bits are fed as input to the first hash slice to generate an encrypted output (hash digest) of 512 bits. Figures 3.12. (d), (e), and (f) represent the initial 136 bits corresponding to En_1, generated pre-processed 1024 bits, and 512 bits output of the first hash slice, respectively. Further, as shown in Figures 3.12. (g) and (h), the 512-bit output of the previous hash slice (*i.e.,* first hash slice) and 136 bits from the second encoding block (*i.e.,* En_2) are taken as input. Again, processed 1024 bits are generated and fed as input to the second hash slice. The complete steps regarding the generation of 1024 bits from 512 bits of the previous hash slice and 136 bits of the current encoding block for the remaining hash slices (*i.e.,* from the second to ninth hash slices) are discussed in the previous subsection. The primary input to hash slices #2-#9 is the output bitstreams generated from encoding blocks #1-#8, respectively. Figures 3.12 (i) and (j) represent the processed 1024 bits and the output of the second hash slice. Similarly, 512-bit hash digest as output is generated corresponding to the remaining hash slices till the ninth hash slice.

Further, the primary inputs of the remaining hash slice (*i.e.,* from #10-#18 hash slices) are controlled using multiplexers and IP vendor selected crypto keys. Let IP vendor selected crypto keys for our proposed approach are as follows; "1001 0010 0100 0001 0111 1000 0011 0110 0101". Therefore, as shown in Figures 3.12. (k) and (l), the tenth hash slice accepts 512 bits output of the ninth hash slice and 136 bits output of encoding block nine (*i.e.,* En_9 based on IP vendor selected crypto key "1001"), respectively. Figures 3.12. (m) and (n) denotes generated processed 1024 bits, fed as input to the tenth hash slice, and

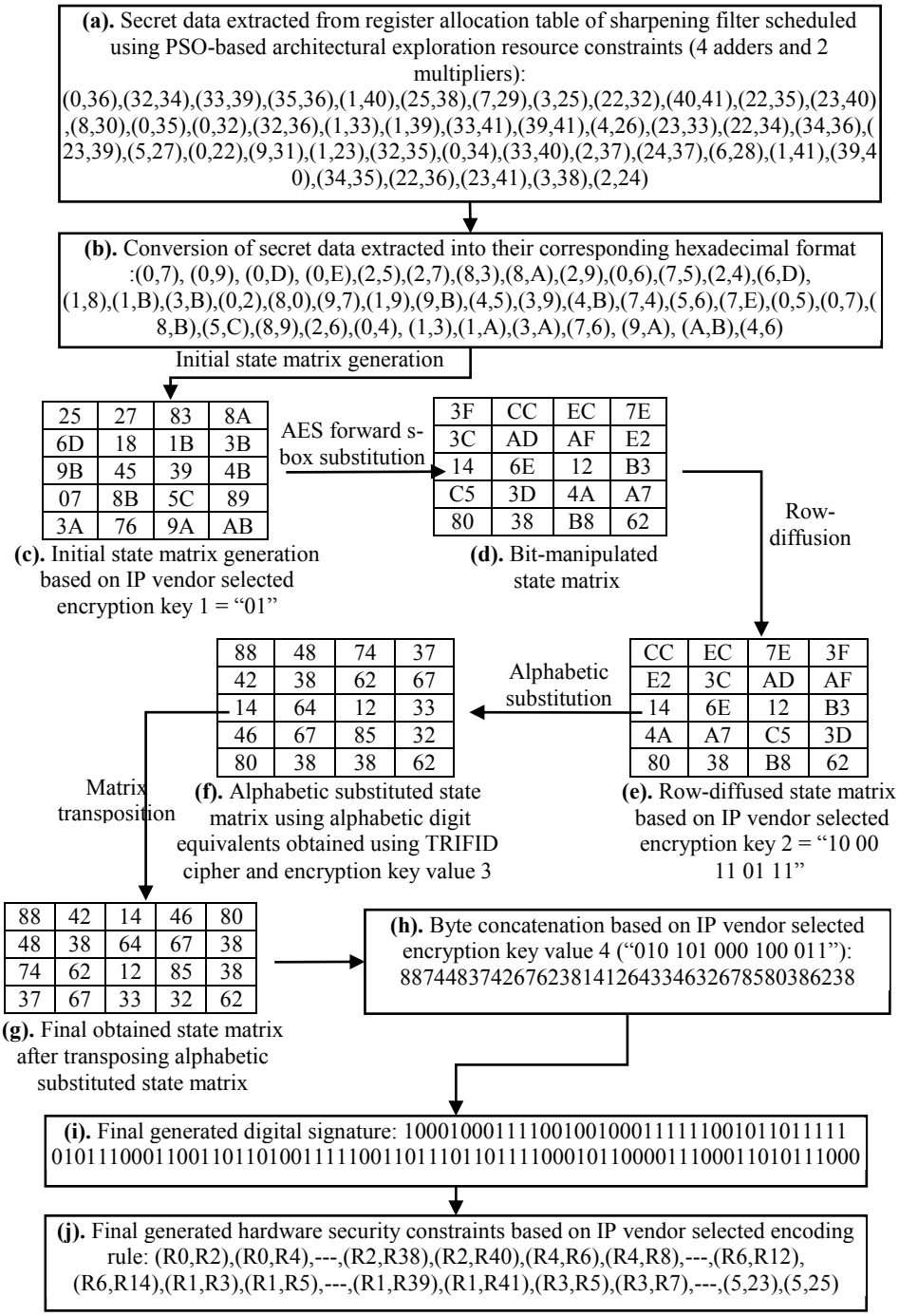


Fig. 3.11. Demonstration of the proposed multi-phase encryption methodology on sharpening filter IP core with its corresponding outputs

obtained 512 bits of tenth hash slice output. Next, the hash digest corresponding to the remaining hash slices is computed in a similar fashion. Figures 3.12. (o), (p), (q), and (r) represent the 512-bit hash digest from the seventeenth hash slice, input bitstream based on IP vendor selected crypto key "0101" (*i.e.,* En_5), processed 1024 bits, and finally generated 512 bits hash digest (signature), respectively. Figures 3.12. (s) and (t) depict the final 512-bit signature (digital template) and secret security constraints generation process.

The generated signature is converted into its corresponding hardware security constraints using IP vendor chosen truncation magnitude and embedding or mapping rules. The IP vendor chosen mapping rule for the proposed hardware security methodology is as follows:

- Embed an additional (i.e., artificial) edge between (even, even) storage

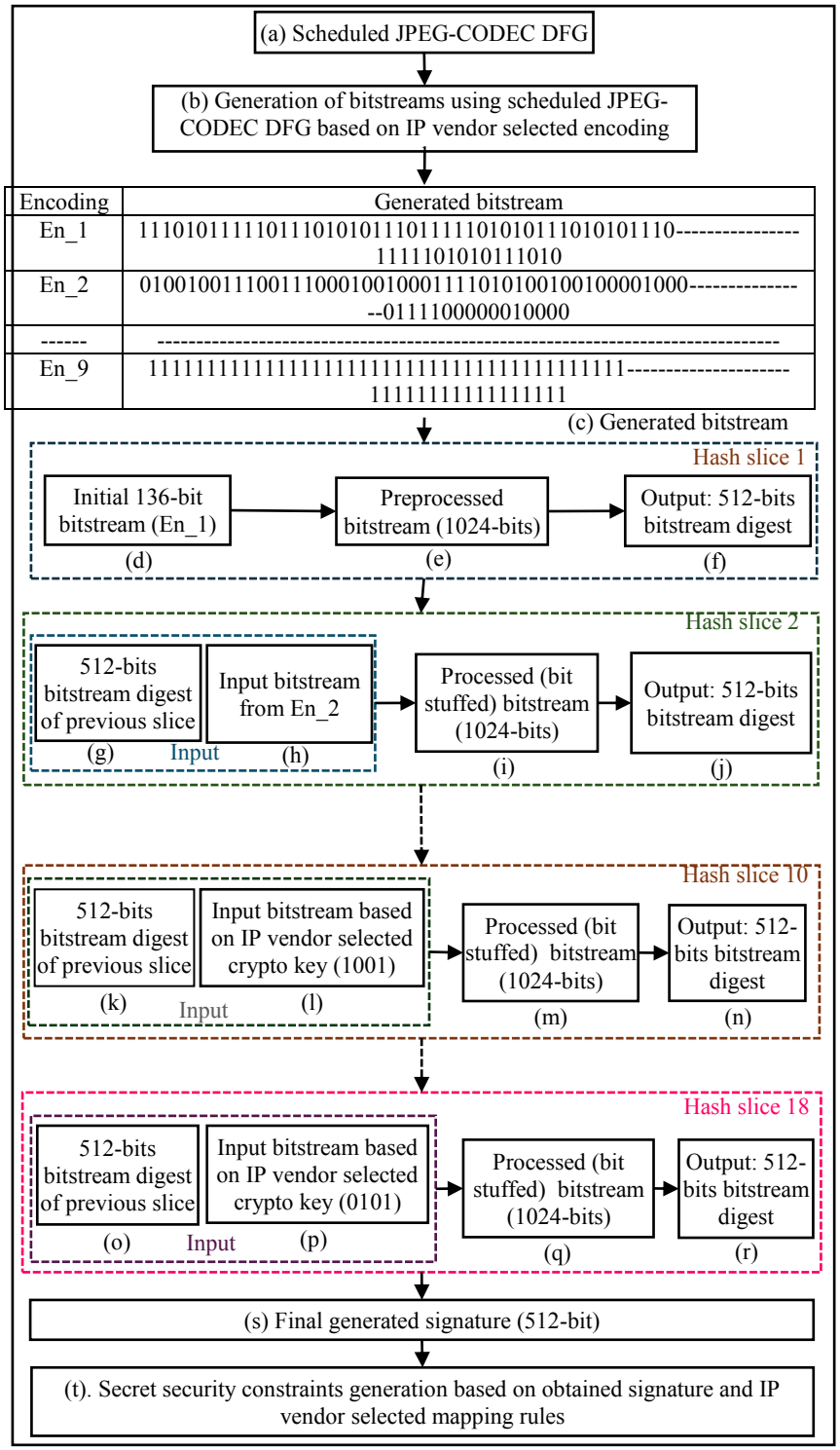


Fig. 3.12. Security constraints generation flow of the proposed low-cost key-based crypto-chain methodology

variables pair in the RAT framework in case of bit '0'; otherwise, embed an edge between (odd, odd) storage variables pair.

Further, the generated key-driven crypto-chain based secret security constraints are implanted into the design of the JPEG-CODEC IP core using register allocation information i.e., RAT framework of the HLS process. Some of the generated security constraints are as follows: *<Q0, Q2>, <Q0, Q4>---<Q0, Q262>---<Q244>, Q128>,<Q1, Q3>---<Q1, Q263>---<Q3, Q261>*. Fig. 3.12. also includes the security constraints embedding process flow. The key concept behind embedding secret security constraints (artificial edge) is that

Table 3.4: Register allocation table corresponding to JPEG-CODEC prior to and after implanting secret security constraints for proposed low-cost crypto-chain signature

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
R_1	Q0	Q128/**Q129**	Q136/**Q137**	Q144/**Q145**	Q152/**Q153**	Q160/**Q161**	Q168/**Q169**	Q176/**Q177**	Q184/**Q185**	Q220/**Q221**	Q238/**Q239**	Q244/**Q245**	Q252/**Q253**	Q258/**Q259**	Q261	-	Q263
R_2	Q1	Q129/**Q128**	Q137/**Q136**	Q145/**Q144**	Q153/**Q152**	Q161/**Q160**	Q169/**Q168**	Q177/**Q176**	Q185/**Q184**	Q221/**Q220**	Q239/**Q238**	Q245/**Q244**	Q253/**Q252**	Q259/**Q258**	-	**Q262**	-
R_3	Q2	Q130/**Q131**	Q138/**Q139**	Q146/**Q147**	Q154/**Q155**	Q162/**Q163**	Q170/**Q171**	Q178/**Q179**	Q186/**Q187**	Q222/**Q223**	Q240/**Q241**	Q246	Q254	Q260	Q260	-	-
R_4	Q3	Q131/**Q130**	Q139/**Q138**	Q147/**Q146**	Q155/**Q154**	Q163/**Q162**	Q171/**Q170**	Q179/**Q178**	Q187/**Q186**	Q223/**Q222**	Q241/**Q240**	Q247/**Q248**	Q255	-	-	-	-
R_5	Q4	Q132	Q140	Q148	Q156	Q164	Q172	Q180	Q188	Q236	Q236	Q248/**Q247**	Q256	-	-	-	-
R_6	Q5	Q133	Q141	Q149	Q157	Q165	Q173	Q181	Q189	Q237	Q237	Q249	Q257	-	-	-	-
R_7	Q6	Q134	Q142	Q150	Q158	Q166	Q174	Q182	Q190	-	Q242	Q250	-	-	-	-	-
R_8	Q7	-	Q143	Q151	Q159	Q167	Q175	Q183	Q191	-	Q243	Q251	-	-	-	-	-
R_9	Q8	Q8	Q192	Q196	Q200	Q204	Q208	Q212	Q216	-	-	-	-	-	-	-	-
R_10	Q9	Q9	Q193	Q197	Q201	Q205	Q209	Q213	Q217	-	-	-	-	-	-	-	-
R_11	Q10	Q10	Q194	Q198	Q202	Q206	Q210	Q214	Q218	-	-	-	-	-	-	-	-
R_12	Q11	Q11	Q195	Q199	Q203	Q207	Q211	Q215	Q219	-	-	-	-	-	-	-	-
R_13	Q12	Q12	-	Q224	Q224	Q224	Q224	Q224	Q224	Q224	-	-	-	-	-	-	-
R_14	Q13	Q13	-	Q225	Q225	Q225	Q225	Q225	Q225	Q225	-	-	-	-	-	-	-
R_15	Q14	Q14	-	-	Q226	Q226	Q226	Q226	Q226	Q226	-	-	-	-	-	-	-
R_16	Q15	Q15	-	-	Q227	Q227	Q227	Q227	Q227	Q227	-	-	-	-	-	-	-
R_17	Q16	Q16	Q16	-	-	Q228	Q228	Q228	Q228	Q228	-	-	-	-	-	-	-
R_18	Q17	Q17	Q17	-	-	Q229	Q229	Q229	Q229	Q229	-	-	-	-	-	-	-
R_19	Q18	Q18	Q18	-	-	-	Q230	Q230	Q230	Q230	-	-	-	-	-	-	-
R_20	Q19	Q19	Q19	-	-	-	Q231	Q231	Q231	Q231	-	-	-	-	-	-	-
R_21	Q20	Q20	Q20	-	-	-	-	Q232	Q232	Q232	Q232	-	-	-	-	-	-
R_22	Q21	Q21	Q21	-	-	-	-	Q233	Q233	Q233	Q233	-	-	-	-	-	-
R_23	Q22	Q22	Q22	-	-	-	-	-	Q234	Q234	Q234	-	-	-	-	-	-
R_24	Q23	Q23	Q23	-	-	-	-	-	Q235	Q235	Q235	-	-	-	-	-	-
R_25	Q24	Q24	Q24	Q24	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R_129	Q135	Q135	Q135	Q135	Q135	Q135	Q135	Q135	Q135	Q135	Q135	Q135	Q135	Q135	Q135	Q135	-

the storage variables corresponding to the additional edge cannot be allocated on the same register. If the storage variables of the incoming artificial edge are allocated on the same register. Then, either a color swapping between registers is performed, or a new register is allocated to meet the hardware security requirement. In other words, a forced distinct register allocation is performed while embedding security constraints. Further, Table 3.4 reports the register allocation table before and after embedding the crypto-chain signature into the JPEG-CODEC. (*Note:* the black color (normal) registers in the Table are the initial position of the registers, while the black colored (bold) registers are the final position of the register after embedding the secret security constraints). The embedded security constraints are represented in terms of change in the positions of registers in the SDFG of the JPEG-CODEC. The alteration in the final position/color of the register reports the embedding of the crypto-chain signature. The presented low-overhead security algorithm offers robust security with zero design cost overhead (*i.e.,* an additional register is not needed after embedding the secret security constraints).

3.4. Watermark Detection Process

3.4.1. Distinguishing between genuine and pirated/fake IP cores

The presence of the proposed hardware security/watermarking constraints (generated through the proposed security algorithms) clearly enables the detection of IP piracy. While performing piracy detection, the key-driven crypto-chain based signature is regenerated using the proposed algorithm and matched with the embedded signature (which is embedded as covert (security) constraints) of the IP core under test. At first, the total number of different bits (such as # *'0s'* and # *'1s'*) present in the signature is matched during litigation/conflict resolution. Next, if the count of the 0's and 1's matches, then, in that case, bit position matching is performed during litigation/conflict resolution. Only an authentic IP vendor would be able to successfully perform the above matching process to validate authenticity. On the contrary, an attacker would fail to regenerate the original signature and match it with the embedded one, thereby failing to validate authenticity. The dependence of the proposed hardware security methodologies on various security factors (explained in previous sub-sections) makes the proposed security methodology

highly robust. The presence of various security factors makes it challenging for an adversary to regenerate the exact signature. They require complete information corresponding to the proposed security methodology, which is only known to the IP vendor/designer.

3.4.2. Resolution of false IP ownership claim

Further, Further, the presence of the proposed signature in the IP core also protects from fraudulent claims of IP ownership. An IP can be fraudulently claimed by an attacker in the SoC house or foundry. For awarding correct ownership initially, the proposed signature/watermark is regenerated. Further, the signature is converted into hardware security constraints. Next, the regenerated security constraint information is matched with the extracted register allocation information of the IP core under test. The ownership is granted to the authentic IP vendor in case of a comprehensive information match. Therefore, the proposed security algorithm also nullifies false claims of IP ownership besides IP piracy.

3.5. Summary

Two novels security methodologies: (a) PSO driven multi-phase encryption and (b) firefly algorithm driven low-cost crypto chain based security methodologies for designing optimal secured image processing filter and JPEG-CODEC IP cores are presented in this chapter. The proposed PSO-driven multi-phase encryption mechanism employs strong security layers such as bit manipulation, row diffusion, TRIFID cipher computation, alphabetic substitution, and byte concatenation to generate a highly robust and tamper-tolerant signature. The main focus of the chapter is to offer a detective countermeasure against potential IP piracy and false claims of IP ownership by attackers in the SoC design house or foundry. The threat model considers the IP vendor as the defender and the SoC integrator/foundry as the attacker. Next, the proposed low-cost key-driven crypto-chain based security methodology incorporates an IP vendor specified encoding mechanism, crypto-keys, #SHA-512 hash slices, and mapping rules to generate secret hardware security constraints. These elements together create a tamper-tolerant signature, which is further converted into security constraints and

embedded into the optimal hardware IP core design obtained using the firefly algorithm based exploration technique. This embedding of security constraints (*i.e.,* digital evidence) safeguards the hardware IP cores from IP piracy and fraudulent claims of IP ownership. Both approaches involves multiple security layers to generate a tamper tolerant signature/watermark, which makes the regeneration harder for the adversary to evade the piracy detection process. The experimental results of the PSO driven multi-phase encryption and low-cost key-driven crypto-chain based security methodologies have been discussed and analyzed in the chapter 9 of this thesis.

Chapter 4

Enhanced Security for Hardware IPs Using IP Seller's Protein Molecular Biometrics and Facial Biometric-based Encryption Key

Hardware IP cores, embedded in a variety of electronic devices such as smartphones, cameras, and health bands, play a critical role in enhancing the performance and efficiency of these devices. In the digital ecosystem, DSP co-processors are essential for handling complex tasks like data compression and decompression, filtering, and image processing. These DSP algorithms are often implemented using dedicated IP cores, which can be designed at a higher level using HLS framework, later synthesized into a RTL data path. With the increasing globalization of the design supply chain, the development of electronic devices and multimedia systems often involves multiple offshore entities, including third-party IP (3PIP) vendors. This introduces the risk of hardware threats such as IP piracy and false claims of IP ownership. Pirated/Counterfeit IPs can compromise user safety and system reliability, potentially causing issues like sensitive data leaks, excessive heat generation, or malfunctioning of DSP hardware. Therefore, it is vital for SoC integrator to safeguard end user against these threats. Moreover, without adequate security mechanisms, adversaries may falsely claim IP ownership. To address such ownership and piracy threats, embedding a robust security marker within the IP can be instrumental in countering false ownership claim and identifying pirated IP cores. The SoC integrator can easily verify the embedded IP vendor's authentic watermark before integration into the final system.

Security against IP piracy is crucial for consumer electronics (CE) systems for two key reasons: (i) pirated designs often bypass thorough reliability and safety testing, and (ii) they are more likely to contain malicious hardware Trojans. A SoC integrator can source IP cores directly from vendors or through intermediaries (IP brokers). However, rogue suppliers may introduce counterfeit IPs into the supply chain, driven by illegal motives. This poses significant risks to both CE system integrator and end consumer, making it crucial to ensure only genuine IPs are used in CE systems. These compromised IP cores pose significant risks when integrated into CE systems.

By embedding a unique encrypted protein biometric signature/watermark of the IP vendor, the proposed approach allows authentic IP cores to be distinguished from pirated ones, preventing counterfeit designs from being incorporated into SoCs. This ensures the safety and reliability of CE devices for consumer. The method is particularly effective in identifying fake IPs, especially when rogue suppliers attempt to introduce pirated components into the supply chain. The proposed solution helps ensure that only legitimate IPs are used in the design and manufacturing process, safeguarding both the system integrator and the end consumer.

This chapter introduces a novel encrypted protein molecular biometric approach for securing hardware/DSP IP cores against piracy and fraudulent IP ownership claim. This method effectively counters false ownership claim and detects counterfeit IPs before they are integrated into electronic systems. The first section of the chapter outlines the problem formulation, threat model and overview. The second section discusses the details of encrypted protein molecular biometrics based security approach. Following this, the third section illustrates the embedding of the proposed watermarking constraints with relevant examples. The fourth section then covers the process of watermark detection. Lastly, the fifth section provides the chapter's conclusion.

4.1. Overview

4.1.1. Threat model and motivation

The threat model is clearly described in the introduction section of this chapter. The proposed approach effectively counters false claims of IP ownership by utilizing the inherently unique molecular and physical biometric properties of the legitimate IP vendor, such as protein samples for molecular signature generation and facial biometrics for encryption key creation. Unlike traditional watermarking and digital signature methods, this robust encoding and encryption process ensures that even if an adversary were to somehow replicate the signature (which is highly unlikely), they would be unable to claim the vendor's identity. The detailed process of signature generation remains unknown to potential attackers, further preventing fraudulent ownership claim. Additionally, the approach ensures strong detection and

isolation of counterfeit IP cores through the embedded, encrypted protein molecular signature, safeguarding the authenticity of the IP design.

4.1.2. Input and Outputs

The primary inputs are (a) transfer and computation function/CDFG of hardware (DSP) application, (b) IP vendor's amino acid sequence/protein biometric sample, (c) IP vendor's facial biometric sample, (d) AES encryption key, (e) IP vendor's concatenation and embedding rules, and (f) IP vendor's specified resource constraints. The final output is an encrypted protein biometric embedded secure RTL datapath of input hardware application.

4.1.3. Target Platform

The proposed security methodologies can be seamlessly integrated with any electronic design automation (EDA) tools. The techniques can easily be combined with HDL, or any high-level language used for IP generation within design tools.

4.1.4. Security framework using IP seller's protein molecular and facial biometrics

This chapter presents a novel approach for securing hardware IP cores by leveraging both "physical biometrics" (facial biometrics for encryption key) and "molecular biometrics" (protein molecular sequences as watermark signature) from the legitimate IP vendor. This method provides a unique cellular and molecular-level signature derived from the IP vendor's body sample, offering robust protection against piracy and false ownership claim. Fig. 4.1 depicts the overview of proposed encrypted protein molecular biometric based security methodology. The proposed system generates an encrypted protein molecular signature from the amino acid sequence, obtained through the protein sequenator process. Additionally, the encryption key for the signature is derived from the vendor's facial biometrics. One key advantage of this approach is that it does not require resequencing or recapturing the vendor's biometric data during the verification process. Instead, a pre-stored encrypted digital template (securely stored in a database), is used for authentication and counterfeit detection. This eliminates the need for continuous biometric input and ensures that the method is immune to

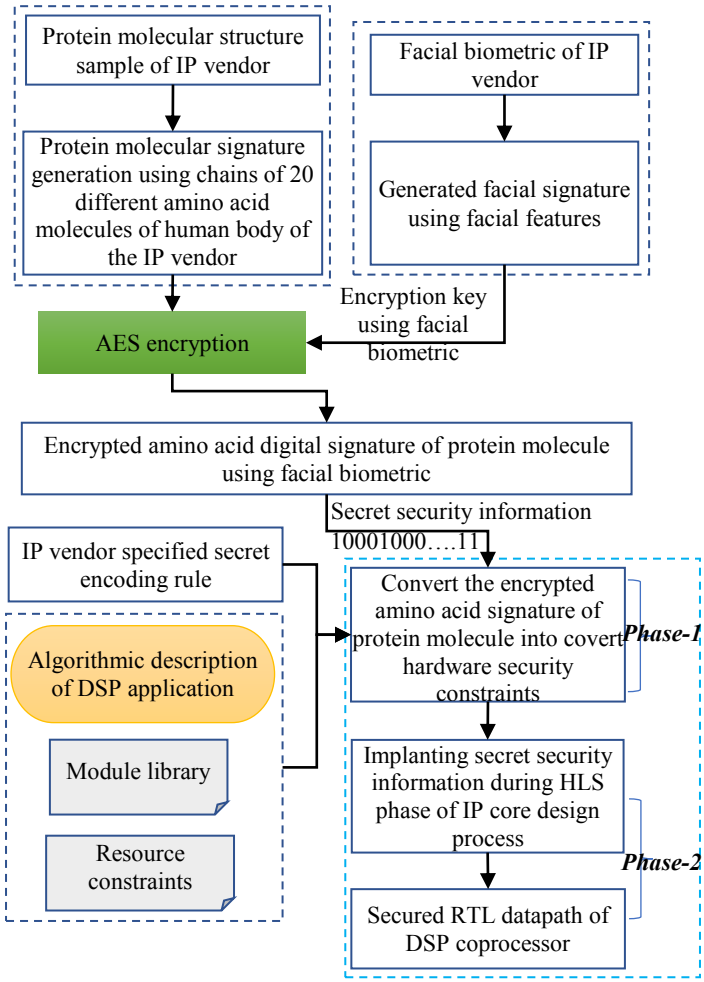


Fig. 4.1. Overview of proposed encrypted protein molecular biometric based security methodology

variations in facial recognition conditions, such as changes in camera angles, scales, or lighting. Thus, the facial biometrics and molecular signature remain consistent during both the embedding and detection phases, corresponding to the genuine IP vendor. This has already been established in the literature [20], [24], [32], [36], [37].

The encrypted protein molecular signature is generated using a sequence of 20 amino acid chemical components extracted from the IP vendor's body sample. This sequence is then fed as input into the AES encryption process, along with the encryption key derived from the vendor's facial biometric features. The result is a highly secure, robust encrypted molecular biometric signature/watermark. To further enhance security, covert hardware security/watermarking constraints are generated from the encrypted signature using an IP seller's embedding/encoding/mapping algorithm/rule (as shown in phase 1 in Fig. 4.1). These constraints are then embedded into the

DSP/hardware design during the register allocation phase of the HLS process (as shown in phase 2 in Fig. 4.1). Ultimately, the method produces a secure RTL design for the DSP coprocessor. The detailed process is elaborated below.

4.2. Encrypted Protein Molecular Biometrics based Security Approach

4.2.1. Introduction to protein molecular amino acid sequences

Human body protein is composed of linear sequence of amino acids. These amino acids are linked to one another through a bond by linking the carboxylic acid group of one amino acid with the amine group of another amino acid called peptide bond. The connected amino acids through peptide bonds in series, forms polypeptide chain or protein sequence. Each protein has a unique amino acid sequence. In a protein sequence chain, there are twenty different amino acids which by linking together forms the chain of amino acids. The largest polypeptide chain may contain 5000 amino acids [102]. A specific digit/letter is used to represent each of the twenty different amino acids present in the protein sequence chain. The process of determining the amino acid sequence from the collected body sample is termed as protein sequencing. It is the practical process for determining the sequence of amino acids in protein sample. However, partial sequencing is also capable to infer sufficient information for identification. The protein samples can be collected from human body samples such as hair, bone, fingernail, saliva, muscle and fingermark etc. Further, the human samples such as hair, bone and muscle samples are more effective for individual identification [103]-[105]. Upon collecting the sample(s), they are brought for protein sequencing, where the samples are applied on swabs and digested with trypsin in order to obtain peptides. Trypsin is an enzyme that starts the digestion of protein molecules by cutting long chains of amino acids into smaller pieces. In this process, the samples such as blood and saliva are characterized by the presence of hemoglobin and alpha-amylase1 biomarkers. Similarly, other samples are identified using corresponding biomarkers. There are two widely used methodologies for protein sequencing viz. mass spectrometry and Edman degradation [106]. Both these methodologies are used for performing protein

sequencing using sequenator (sequencer). The ongoing and recent developments in the field of proteomics, renders it impactful and important for forensics and unique identification. With the development in the field of proteomics in terms of increased availability of high-resolution mass spectrometers, maturity of sequencing and different methodology for protein sequencing have been effective and useful for enabling the usage of protein molecular for unique identification.

Typically, the sequencing process of protein comprises of the following steps:

a. Break the disulfide bridges in the protein using reducing agent.
b. Form the separate chain(s) of protein complex.
c. In each chain, determine the terminal and composition of amino acids.
d. Divide each chain into small size fragments (up to 50 amino acids in each).
e. Separate each of the resultant fragment and purify them.
f. Exploit the fragments to determine amino acid sequence.
g. Repeat steps 1-6, to construct overall protein sequence.

4.2.2. Advantages of protein molecular biometric signatures compared to DNA molecular biometrics

The protein sequencing is more advantageous than DNA sequencing from the perspective of molecular biometrics for generating unique and robust digital signature [102], [103]:

a. Compared to genomic analysis, proteomics (proteomics is large scale study of all proteins of an organism or system by using mass spectrometry) can provide an accurate distinctive detail of the human body sample. Protein analysis is a confirmatory and orthogonal technique that helps in forensic identification.
b. The proteins in sample are more persistent and chemically more robust than DNA and can persist for longer durations, whereas DNA can become degraded in the environment.
c. DNA sample contains the genome information whereas proteins are much of what determines a cell's characteristics and function.

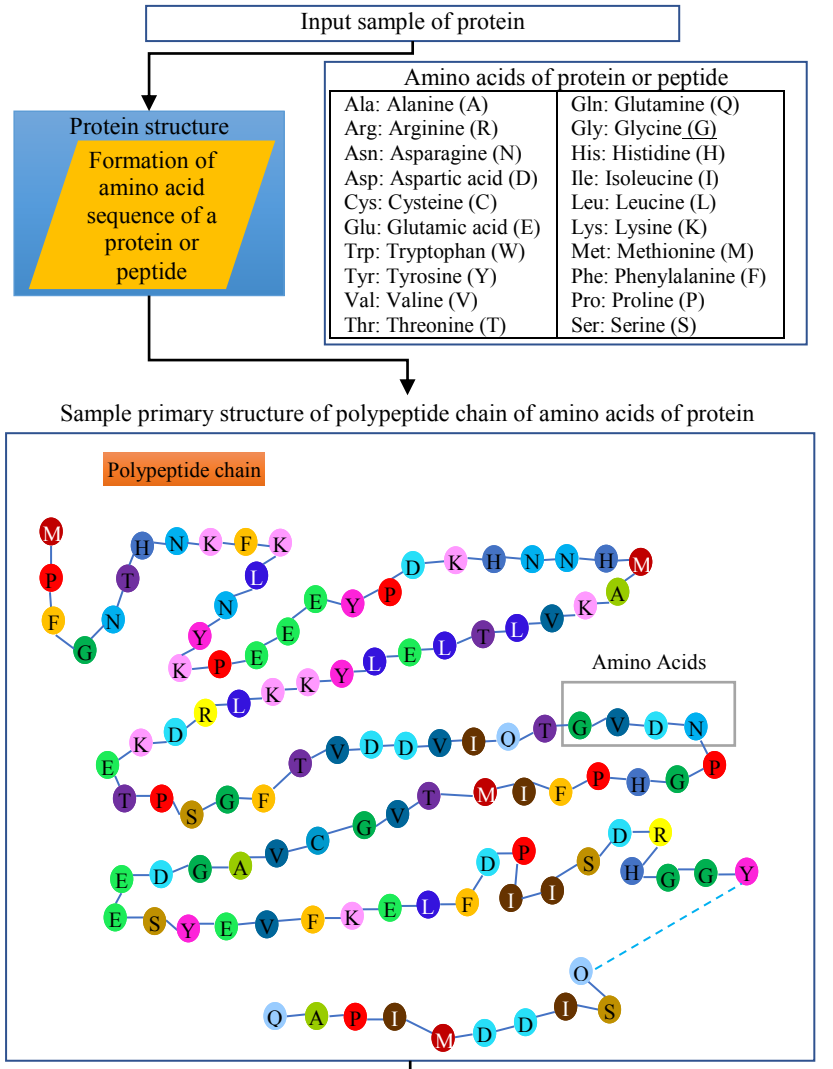
d. Protein sequencing is more accurate than DNA sequencing because the proteins are composed of twenty amino acids while the DNA contains only four chemical compositions. Therefore, the signal to noise ratio in protein sequence chain is much better as compared to DNA.
e. The generated protein molecular signature is highly distinctive than DNA signature which results in unique identification of sample.
f. The generated protein molecular signature results in higher tamper tolerance ability (robustness of digital evidence) and lower probability of coincidently detecting the authentic secret security constraints within an unsecured design (higher security strength).

Therefore, protein molecular signature, generated from the amino acid sequence contains more accurate distinction of human body sample at the cell level. Due to advancement in the field of biosciences, it is easy to analyze the protein sample and report the sequence of amino acids present in a human body sample. Moreover, the process of protein sequencing is cost effective which makes it feasible for unique sample identification using the concept of molecular biometrics.

4.2.3. Generation of the Proposed IP seller's protein molecular signature

As discussed in earlier section that protein sample can be analyzed using protein sequenator for generating the protein sequence. For explaining the process of proposed molecular signature generation, we have considered the protein sequence of the human body of the IP vendor as input, as shown in Fig. 4.2. In this protein molecular sequence of amino acids as shown in Fig. 4.2, the formation of amino acid chain comprises of 20 different amino acids, resulting into a long polypeptide chain. Therefore, in this protein sequence, the polypeptide chain consists of 380 amino acids. Each amino acid is represented by a unique alphabet. Further it is to be noted that the polypeptide chain length is scalable depending upon the smaller or larger amino acid sequence generated from the human body protein sample. The polypeptide chain length can be selected based on the IP vendor choice. Subsequently, for the selected amino acid sequence length of the polypeptide chain, encoding is performed. Each amino acid has a unique encoding based on their alphabetical positions. For example, for amino acid methionine (M), the alphabet position is 13 and

corresponding encoded binary bits are 1101, for proline (P) the alphabet position is 16 and corresponding encoded binary bits are 10000, while for tryptophan (W) it is 10111 etc. Subsequently, all amino acids are encoded with



Encoding amino acids of protein using encoding algorithm

Naming conventions of amino acids	Alphabet position & Binarize value	Naming conventions of amino acids	Alphabet position & Binarize value	Naming conventions of amino acids	Alphabet position & Binarize value
M	(13) -1101	K	(11)-1011	V	(22)-10110
P	(16) -10000	L	(12)-1100	I	(9)-1001
F	(6)-110	Y	(25)-11001	Q	(17)-10001
G	(7) - 111	E	(5)-101	C	(3)-11
N	(14) - 1110	D	(4)-100	R	(18)-10010
T	(20)- 10100	S	(19)-10011	W	(23)-10111
H	(8)-1000	A	(1)-1	-	-

Fig. 4.2. Generating the protein molecular signature corresponding to amino acid sequence of sample protein

a unique encoding bit. Finally, by concatenating the binarized bits corresponding to amino acids of polypeptide chain, protein molecular signature is obtained. The binarized protein molecular signature of 1500 bit from amino acid sequence chain corresponding to protein sample of IP vendor is generated.

4.2.4. Using IP seller's facial biometrics to generate encryption keys

In order to generate encrypted protein molecular signature for securing DSP hardware IP cores, facial biometric based encryption key has been used. The process of facial biometric key generation from the facial features of IP vendor has been discussed below:

a. Capture the facial image of IP vendor using high resolution camera.
b. This captured facial image is then subjected to a specific grid size and spacing (specified by the IP vendor for generating the nodal points on the facial image precisely). This also mitigates the impact of face movement, thereby resulting in accurate facial feature generation.
c. Based on the selected feature set (among eleven features as shown in Fig. 4.3), nodal points are generated on the facial image.
d. Assign the naming conventions on nodal points of facial image.
e. Generate the facial image with IP vendor chosen feature set.
f. Compute the feature dimensions between the nodal points corresponding to each facial feature chosen by IP vendor for encryption key generation. To do this, first the coordinates of each nodal points are computed and subsequently, feature dimensions are computed using Manhattan distance (|x2-x1|+|y2-y1|), where (x1, y1) and (x2, y2) are the coordinates of the facial feature nodal point.
g. Each feature dimension of selected facial features is transformed into their corresponding binarized value.

Finally, based on the concatenation order of facial features (decided by IP vendor), the facial biometric based encryption key is generated. For example: for the following concatenation order of facial features such as: (HFH) & (IPD) &(BOB) &(IOB) &(OB) &(WNR) &(WF) &(HF) &(WNB) &(NB)

&(OCW), the encryption key has been generated as shown in Fig.3. Further, the IP vendor generated facial key is to be used in AES encryption.

Note: Facial biometric renders robustness even if an adversary is look alike or twin to the true vendor. Furthermore, exact regeneration of encryption key is impossible for an adversary as it depends on several intricate parameters such as (a) grid size/spacing used in determining the precise coordinates of nodal points (b) type of chosen feature set by true IP vendor (among the exhaustive features for generating digital template) c) their concatenation order, all are only known to an authentic IP vendor.

4.2.5. Generation of encrypted protein molecular biometric signature using AES encryption

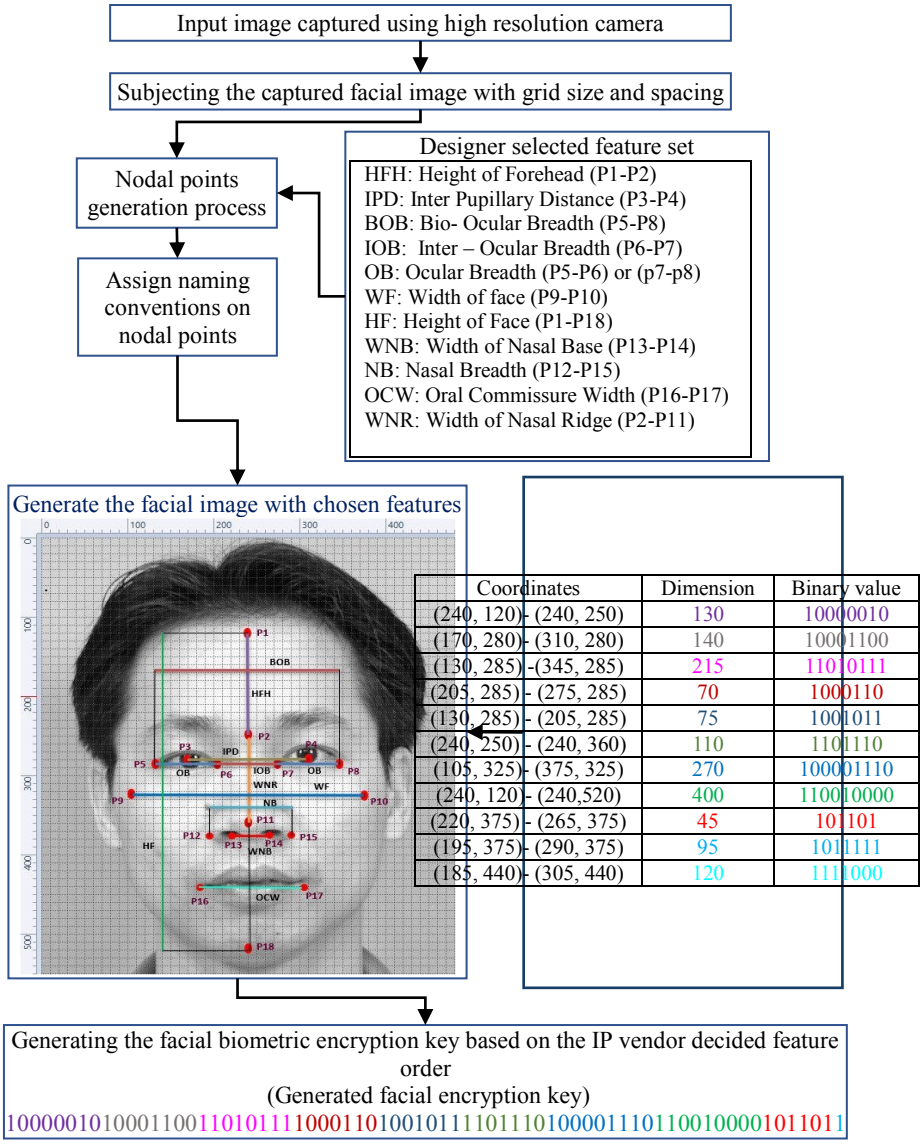


Fig. 4.3. Demonstration of facial biometric key generation used for encrypting the protein molecular signature

The inputs to the AES encryption phase of proposed approach are: 1) unique protein molecular biometric signature (generated using chain of twenty different amino acids) of body sample of IP vendor and 2) encryption key which is derived from the facial biometric features of IP vendor. The encrypted protein molecular digital signature of protein molecule using facial biometric is obtained as an output. After the completion of 10 iterations of AES encryption corresponding to initial protein molecular signature strength, the encrypted protein molecular digital signature is obtained (corresponding to protein molecular signature shown earlier in Fig. 4.2). Finally, the encrypted protein molecular signature is generated at the end of AES encryption process. For example, the encrypted protein molecular signature using facial encryption key corresponding to input protein molecular signature is as follows: "11101011111000101110111011010000010111110111011111010101100110 10111101011110……..11101000101011010100111101111011000100000011001000001001101110111010100011000100 1". This generated encrypted digital template is embedded into the design for securing the DSP hardware IP cores against piracy and false ownership claim.

4.2.6. Security properties of the proposed encrypted protein molecular biometric watermark signature

The proposed encrypted protein molecular biometrics-based hardware security approach renders several security properties as discussed below:

i. Exact regeneration of protein molecular signature is not possible for an adversary because of the following: a) length of the polypeptide chain in the protein molecule sequence for signature generation and b) encoding rule corresponding to 20 different amino acids c) secret facial encryption key, all are unknown to an adversary.

ii. The exact regeneration of facial key is not possible for an adversary as the following crucial details chosen by an IP vendor is not known to an adversary such as: a) specific grid size/ spacing, b) number of chosen facial features and c) concatenation order of facial features for final facial key generation.

iii. However, in the rare likelihood, even if an adversary manages to access the stored encrypted authentic digital template, the proposed

approach still provides robust security against fraudulent claim of IP ownership and pirated designs. This is because, it is not possible for an adversary to exactly regenerate the secret hardware security constraints corresponding to facial biometric encrypted protein molecular sequence of original IP vendor (used for embedding into the genuine IP core design for enabling robust security) due to obscured non-decodable crucial security parameters of the proposed approach. The details of obscured non-decodable crucial security parameters of the proposed approach are discussed earlier in this section, which demonstrates that replication or regeneration of secret hardware security constraints is not possible for an adversary.

iv. The AES encryption algorithm itself offers highly robust encryption.

v. The encoding rule used for generating the secret security constraints corresponding to encrypted protein molecular signature is also not known to an adversary.

vi. The proposed approach results into the generation of highly robust encrypted protein molecular signature which yields higher tamper tolerance ability of the secured design.

vii. It is not possible for an adversary to evade the counterfeit detection process as the complete matching of secret security constraints of regenerated signature is mandatory with the extracted register allocation information of the target design under test.

viii. Protein molecular signature is immune to aging effects, injury and other external environmental factors.

ix. It offers more distinctive and unique identification as compared to other non-biometrics and biometrics-based hardware security approaches.

4.3. Demonstration: Watermark Embedding and Secure RTL Design Generation Process

The process of obtaining the secured (embedded with encrypted protein molecular signature through facial biometric based AES encryption) RTL datapath of DSP co-processor using HLS, is discussed in two phases (as shown in Fig. 4.1).

Phase-1 is responsible for generating the covert hardware security constraints corresponding to encrypted protein molecular biometric signature generated using the proposed approach. The security constraints generation process accepts the following inputs: a) DSP application in the form of data flow graph (DFG) /transfer function b) vendor specified encoding rule (discussed subsequently). The DFG is the scheduled using 'LIST' scheduling algorithm based on designer specified resource constraints and the available dependency information of storage variables as highlighted in the scheduled data flow graph shown in Fig. 4.4. Where X0- X22 are the storage variables and required registers are designated using different colors corresponding to DCT-8 point IP cored design. As evident, nine control steps (C0 -C8) were required to schedule and obtain the final output value. Subsequently the register allocation table corresponding to scheduled data flow graph is constructed. Next, using the following encoding rule, the obtained encrypted protein molecular signature bits are converted into its respective hardware security constraints.

Encoding rule:

- Bit '1' signifies the embedding of security constraints between odd-odd storage variable pair X (i, j) of the scheduled DFG, where i and j represents the specific storage variable used for pairing.
- Bit '0' signifies the embedding of security constraints between even-even storage variable pair of the scheduled DFG.

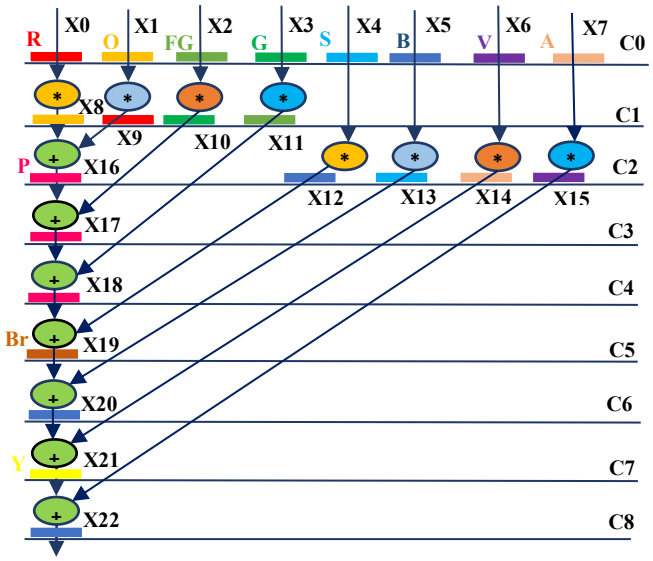


Fig. 4.4. Scheduled data flow graph of DCT-8 with 1(+) and 4(*) post embedding secret constraints.

For example, the secret hardware security constraints corresponding to encrypted protein molecular signature (108 bit) for 8-point DCT IP core are generated as follows: [X(0, 2), X(0, 4), X(0, 6), X(0, 8), X(0, 10), X(0, 12), X(0, 14), X(0, 16), X(0, 18), X(0, 20), X(0, 22), X(2, 4), X(2, 6), X(2, 8), X(2, 10), X(2, 12), X(2, 14), X(2, 16), X(2, 18), X(2, 20), X(2, 22),….., X(12, 16), X(1, 3), X(1, 5), X(1, 7), X(1, 9), X(1, 11), X(1, 13), X(1, 15), X(1, 17), X(1, 19), X(1, 21), X(3, 5), X(3, 7), X(3, 9), X(3, 11), X(3, 13), X(3, 15), X(3, 17), X(3, 19), X(3, 21),..……, X(19, 21)]. Finally, the security constraints are generated which is subsequently embedded into the design (in the register allocation of the scheduled data flow graph) as discussed in phase-2.

Phase-2 is responsible for implanting the generated secret security constraints corresponding to protein molecular signature into the design. The security constraints are embedded into the design during register allocation phase of HLS process. The register allocation table constructed earlier in phase-1, contains the details of required control steps for generating the output functionality, storage variables of the DSP coprocessor and details of registers required for accommodating the intermediate and final operational value of storage variables. The register allocation table is used for embedding the hardware security constraints by locally altering (modifying) the register assignments using the following rule such that two storage variables in a pair cannot be assigned to the same register. Finally, the encrypted molecular signature implanted modified register allocation table corresponding to 8-point DCT is obtained, as shown in Table 4.1. The storage variables marked in red color are indicating the local alterations, post embedding the secret security constraints into the design (covertly). *Note*: sometimes it may require allocation of new register(s) for accommodating the storage variable. Subsequently, the secured RTL datapath corresponding to 8-point DCT, embedded with encrypted protein molecular signature is obtained.

4.4. Detection of Fake/Pirated IP Versions and Resolution of False Claim of IP Ownership

Verification of false IP ownership claim: In case if an adversary (located at either at offshore third-party design houses or foundry) fraudulently claims IP ownership, then the pre-stored encrypted protein molecular sequence and

Table 4.1: Register allocation in 8-point DCT (after embedding encrypted protein molecular signature)

CS	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
CS0	X0	X1	X2	X3	X4	X5	X6	X7	--	--	--
CS1	X9	X8	X11	X10	X4	X5	X6	X7	--	--	--
CS2	--	--	X11	X10	X13	X12	X15	X14	X16	--	--
CS3	--	--	X11	--	X13	X12	X15	X14	X17	--	--
CS4	--	--	--	--	X13	X12	X15	X14	X18	--	--
CS5	--	--	--	--	X13	--	X15	X14	--	X19	--
CS6	--	--	--	--	--	X20	X15	X14	--	--	--
CS7	--	--	--	--	--	--	X15	--	--	--	X21
CS8	--	--	--	--	--	X22	--	--	--	--	--

facial biometric encryption key is used for robust and seamless verification of authentic IP ownership. Encrypted protein molecular sequence and encryption key using facial biometric traits are safely stored (in a safe database) with genuine IP vendor. This, therefore, makes the proposed approach independent of protein resequencing and recapturing face image. Thus, the proposed facial biometric is independent of variations caused by different angles, scales, or illuminations.

In the proposed approach seamless and robust verification of ownership can be performed without recapturing or reproducing biometric information of original IP vendor. This is because regeneration of exact secret hardware security constraints by decrypting the pre-stored digital template is possible by the authentic IP vendor only. Reproduction of the IP vendor biometric information during verification of IP ownership and detection of counterfeited IPs is not required, as its equivalent digital template is pre-stored in a safe database in encrypted form. Therefore, in order to nullify the false claim, the positions of authentic protein molecular signature bits are matched bit by bit with the embedded protein molecular digital template corresponding to DSP design under test. Consequently, based on the complete matching, ownership is awarded to the genuine IP vendor. As the protein molecular signature provides cellular/ molecular level distinction, therefore it is not possible for an adversary to possess similar molecular characteristics as that of genuine IP vendor to satisfy the claim of IP ownership.

Detection of counterfeited IP versions: The embedded protein molecular signature also enables the detection of genuine/authentic DSP IP cores by isolating them from counterfeited IP designs. During the counterfeit detection process, secret security constraints corresponding to protein molecular signature are regenerated and matched with the information of register

allocation extracted from RTL design under test. If the presence of protein molecular signature is not found in the design, then it is probably counterfeit. Moreover, the involvement of several complex information during encrypted protein molecular signature generation and implantation, makes it almost impossible for an adversary to evade the counterfeit detection process.

4.5. Summary

This chapter presented a novel hardware security method that combines a protein molecular biometric signature from a human body sample with a facial biometric-based encryption key specific to the IP vendor. To secure the IPs, the design embeds an encrypted version of the protein molecular signature—derived from a unique sequence of 20 amino acids—during the HLS process. This approach helps in identifying counterfeit IP designs and prevents false claims of IP ownership by integrating both the molecular signature and the vendor's facial biometric data. The method demonstrates superior security, offering a lower probability of coincidence and greater resistance to tampering compared to recent alternatives, discussed in result section in Chapter 9.

Chapter 5

Securing Hardware IPs by Exploiting Statistical Watermarking Using Encrypted Dispersion Matrix and Eigen Decomposition Framework

As described in the introduction chapter, the demand for application-specific computing has become increasingly vital due to the need for enhanced performance, power efficiency, real-time capabilities, scalability, and cost reduction. Application-specific hardware IP cores are designed (using HLS framework) to handle tasks that require significant computational power, such as data filtering, compression, and complex mathematical operations. HLS, a crucial technology in electronic design automation (EDA), simplifies the design of complex integrated circuits by bridging the gap between high-level descriptions, often written in programming languages like C/C++, and the lower-level hardware implementations. However, with the global design supply chain, security of hardware IP cores has emerged as a significant concern. Malicious actors within SoC integrator design houses may attempt to pirate or fraudulently claim ownership of imported hardware IPs. To combat this, this chapter introduces a security methodology that incorporates/embeds an encrypted dispersion matrix and eigen decomposition framework based watermarking constraints to safeguard against IP piracy. The proposed approach enables the generation of a unique secret mathematical (using statistical modelling) signature/watermark to secure the hardware IP against piracy and false IP ownership claim.

This chapter introduces a novel encrypted mathematical (statistical) watermarking approach for securing hardware/DSP IP cores against piracy and fraudulent IP ownership claim for the first time in literature. This method effectively counters false ownership claim and provide detective countermeasure against IP piracy. The first section of the chapter outlines the threat model and motivation of the proposed approach. The second section discusses the details of proposed statistical watermarking using encrypted dispersion matrix and eigen decomposition framework**.** Following this, the third section illustrates the embedding of the proposed watermarking constraints with relevant examples. The fourth section then covers the IP

piracy detection and resolution of false IP ownership claim process. Lastly, the fifth section provides the chapter's conclusion.

5.1. Threat Model, Motivation, and Target Platform

5.1.1. Threat model

A SoC integrator design house may be compromised by internal adversary who may pirate the original hardware IP design imported from authentic IP vendor. The motivations behind this IP piracy could include generating illegal profits, damaging the reputation of the legitimate vendor, or embedding malicious logic in the pirated design. Additionally, an adversary within the SoC integrator might attempt to fraudulently claim ownership of the IP. To counter these threats, the authentic IP vendor must secure the hardware IP before sharing it with the integrator. The proposed approach treats the IP vendor as the defender and the SoC integrator as the attacker, embedding secret digital evidence via an encrypted dispersion matrix and eigen decomposition watermarking framework, offering detective control against IP piracy and false ownership claim.

5.1.2. Motivation: using statistical watermarking framework for securing hardware IPs

The proposed watermarking methodology introduces a novel security mechanism for hardware IP cores by using a 2D design parameter-driven encrypted dispersion matrix combined with eigen decomposition framework to create a secure watermark. This approach leverages statistical techniques like variance and covariance of design space parameters to embed unique, irreproducible watermarks within the IP design. By using covariance, the method captures the relationship between key design metrics such as area and delay, while variance and eigenvalues help measure the spread and characteristics of the design parameters. The mathematical watermark is derived from the inherent properties of the IP design, such as resource configurations, without relying on external identifiers like signature or biometrics. This makes the watermark not only unique but highly resistant to tampering. Unlike other statistical models, this approach effectively captures the core characteristics of the IP design space, providing robust digital

evidence to secure hardware IP cores against piracy and unauthorized ownership claim.

5.1.3. Target platform, Inputs and output

The proposed security methodologies can be seamlessly integrated with any electronic design automation (EDA) tools. The techniques can easily be combined with HDL, or any high-level language used for IP generation within design tools. The primary inputs are (a) algorithmic description or control data flow graph (CDFG) of target hardware, (b) module libraries (containing necessary information such as area, delay, etc. corresponding to functional units used in hardware design), (c) IP vendor chosen *p-bit* key for initial resource configuration generation, (d) LIST scheduling algorithm, and (e) IP vendor selected key for AES encryption. The final output is a secure hardware IP core using proposed 2-D design parameter driven encrypted dispersion matrix and eigen decomposition based security framework.

5.2. Statistical Watermarking Using Encrypted Dispersion Matrix and Eigen Decomposition Framework

5.2.1. Overview

Fig. 5.1 highlights the overview of the proposed mathematical/statistical watermarking approach. The proposed security framework incorporates a multi-phase process to generate a tamper resistant mathematical watermark. In the first phase, the IP vendor's selected *p*-bit key is used to generate resource configurations for the specific hardware design. The IP design's characteristics, such as area (A_d) and latency (L_d), along with their variances and covariance, are extracted. These values form a dispersion matrix, which serves as the basis for the security constraints derived from the vendor's chosen design space parameters. The second phase focuses on further characterizing the hardware design by calculating eigenvalues (λ_n) from the selected resource configurations. These eigenvalues, or characteristic roots, leads to the generation of additional security constraints. In the third phase, both the elements of the dispersion matrix (variance and covariance of A_d and L_d, ($var\ (A_d)$, $var\ (L_d)$, $(cov\ (A_d, L_d))$) and the eigenvalues (corresponding to two 2*2 square matrices for minimum of four different resource

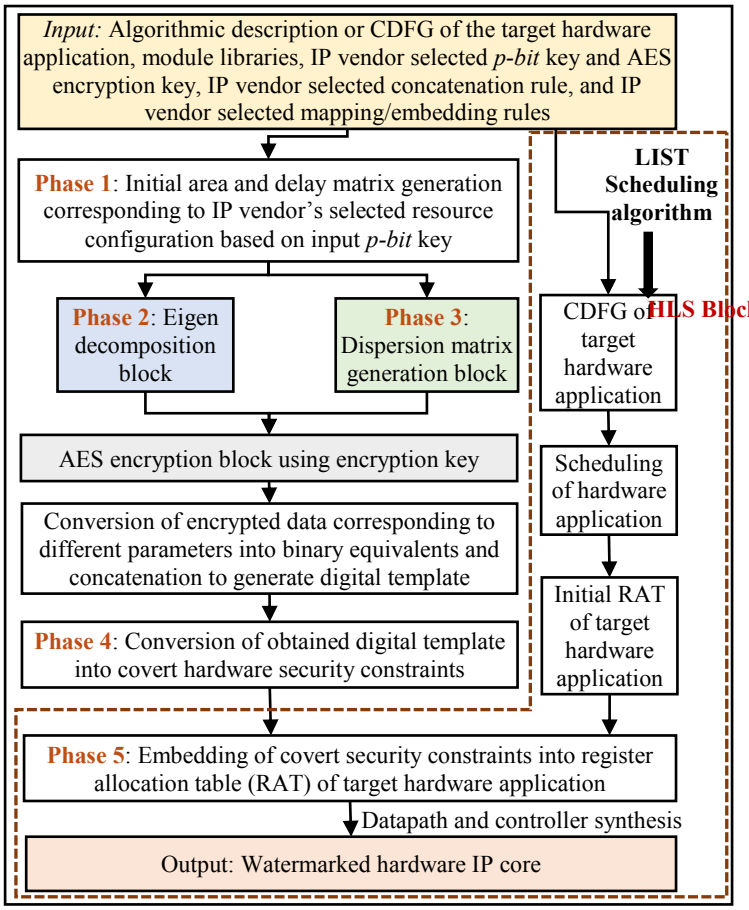


Fig. 5.1. Overview of the proposed mathematical watermarking methodology

configurations) are encrypted using the vendor's AES key. This results in seven unique variables that contribute to the generation of mathematical watermarking/security constraints. As more resource configurations are selected, more eigenvalues can be extracted, increasing the number of security variables and enhancing watermark strength. The encrypted data is then converted into binary form and concatenated according to the vendor's specific rule, resulting in a secure watermark signature. The fourth phase involves generating covert security constraints using the watermark signature, embedding them into the hardware design during the register allocation phase of HLS process. In the final phase, the covert security constraints are embedded into the RAT to generate the final, secure hardware IP core design. The inclusion of multiple convoluted security variables, such as dispersion matrix, eigen decomposition, encryption, etc. leads to the creation of a highly tamper-resistant watermark signature.

5.2.2. Extracting secret security data from encrypted dispersion matrix based on hardware design space parameters

The proposed security methodology accepts algorithmic description or CDFG of target hardware application (for example, 8-point DCT is used for demonstration) along with its module library and IP vendor selected *p-bit* key for generating resource configurations from the design space (as discussed in the overview section). Figure 5.2 illustrates the CDFG of an 8-point DCT application. The CDFG is used to generate the RAT, which is used for performing the embedding of security constraints, *i.e.,* digital evidence. In the first phase, initially, the IP vendor selected *p-bit* secret key is used to generate the resource configurations (*Note: The secret key size (p-bit) depends upon the design space size corresponding to the target application. For example, in the case of 8-point DCT, the maximum number of adders (A_M) and multipliers (M_M) required for parallel implementation is eight. The resources can be generalized to other design types depending on the application. Therefore, the exhaustive design space size is (1*8 = 8), i.e., 2^3. Hence, a 3-bit IP vendor selected key can represent 8 resource configurations (R_C) in the design space of an 8-point DCT. Here, for the sake of brevity, only four resource configurations, along with their area and latency, are shown for*

Table 5.1: Generation of resource configurations and its respective area (A_d) and latency (L_d) matrix corresponding to 8-point DCT based on IP vendor selected four different *p-bit* keys

p-bit key	R_C (IP vendor chosen - key controlled)	A_d (IP vendor computed)	L_d (IP vendor computed)
011	[1, 4]	327 um^2	927 ps
000	[1, 1]	101 um^2	2186 ps
111	[1, 8]	629 um^2	729 ps
100	[1, 5]	403 um^2	927 ps

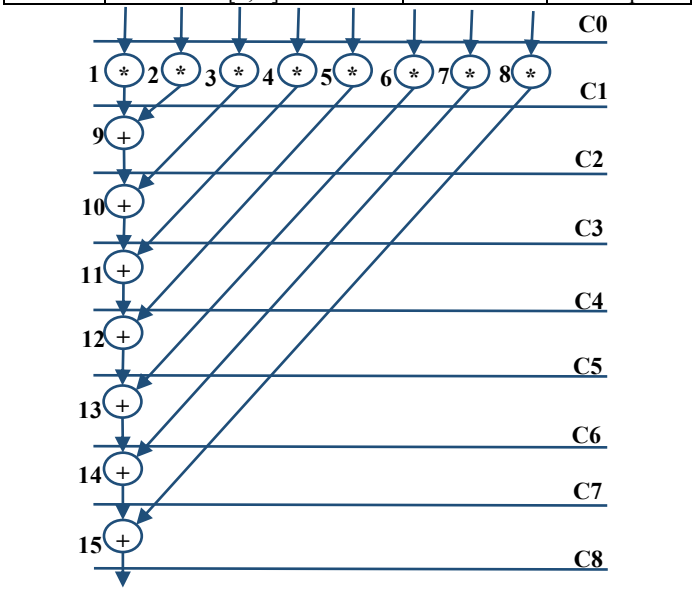


Fig. 5.2. Control Data flow graph of 8-point DCT

demonstration purposes in Table 5.1 based on the p-bit key selected by the IP vendor. However, it can be expanded as per the IP vendor's key selection). The area *(A_d)* and latency *(L_d)* corresponding to key-selected resource configurations are computed using the specified module library. Table 5.1 reports the area and latency corresponding to different key values.

Fig. 5.3. shows the details of the proposed dispersion matrix generation block. In the first step, the generated area *(A_d)* and latency *(L_d)* matrix (given in Table 5.1) is fed as input to the dispersion matrix generation block to extract the characteristics of the IP vendor selected design space parameters in terms of variance of A_d *(var (A_d))*, the variance of L_d *(var (L_d))*, and covariance *(cov (A_d, L_d))*. Next, the mean of the area ($\overline{A_d}$) and latency ($\overline{L_d}$) is computed in step 2. (a) and (b). Subsequently, in steps 3. (a) and (b), the mean value of the area ($\overline{A_d}$) and latency ($\overline{L_d}$) is subtracted from the area and latency parameter values of the resource configurations. Further, the sum of the square of the difference corresponding to the design area ($\sum_{i=1}^{n}(A_{di} - \overline{A_d})^2$) and latency ($\sum_{i=1}^{n}(L_{di} - \overline{L_d})^2$) are computed in step 4. (a) and (b). Post computing the sum of the square of the difference, the characteristics of the IP vendor selected design space parameters (area (A_d) and latency/delay (L_d)), *i.e., var (A_d), var (L_d),* and *cov (A_d, L_d)* are computed. Finally, a dispersion matrix is generated corresponding to the target application.

Further, the generated characteristic features (elements of dispersion matrix), *i.e., var (A_d), var (L_d),* and *cov (A_d, L_d),* are encrypted individually using the AES-128 encryption mechanism based on IP vendor chosen 128-bit private key for each (*128-bit*3 = 384 bits* in total). The corresponding encrypted values are generated as output. Subsequently, each encrypted output is converted into its binary equivalent to generate 128-bit encrypted data corresponding to each element of the dispersion matrix.

Demonstration of generating secret security data from the encrypted dispersion matrix corresponding to the 8-point DCT application*:*

Step 1. Computation of area *(A_d)* and latency *(L_d)* corresponding to IP vendor chosen resource configuration: The area and latency corresponding to selected resource configurations are shown in Table 5.1.

Step 2. (a). Mean computation of design parameter 'A_d':

$$\overline{A_d} = \sum_{i=1}^{n} A_{di} \tag{5.1}$$

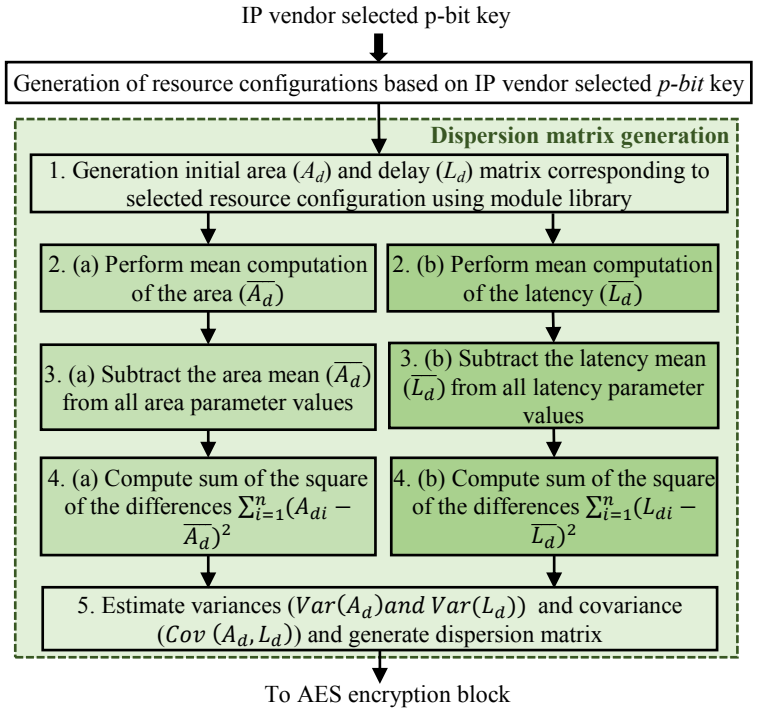


Fig. 5.3. Details of the proposed dispersion matrix generation block

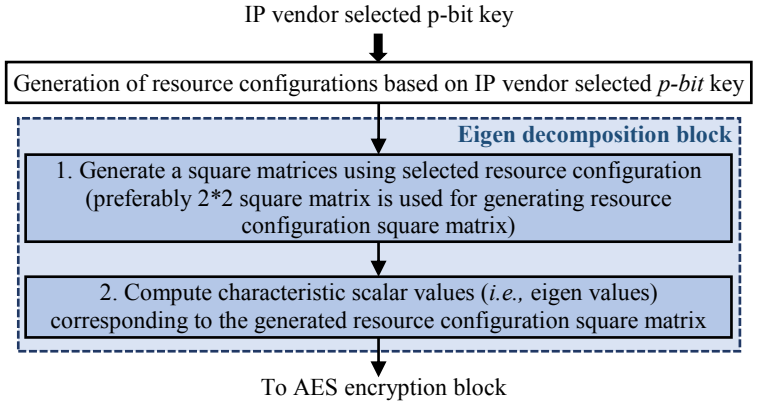


Fig. 5.4. Details of the proposed eigen decomposition block

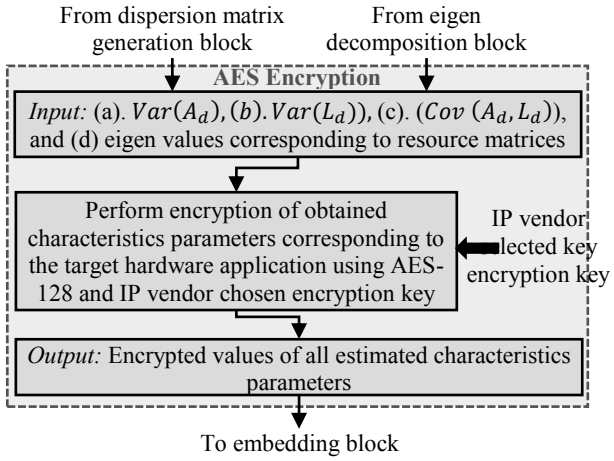


Fig. 5.5. Details of the AES encryption block

$$\overline{A_d} = \frac{(327 + 101 + 629 + 403)}{4} = 365$$

Step 2. (b). Mean computation of design parameter 'L_d':

$$\overline{L_d} = \sum_{i=1}^{n} L_{di} \tag{5.2}$$

$$\overline{L_d} = \frac{(927 + 2186 + 729 + 927)}{4} = 1192.25 = \sim 1192$$

Step 3. (a). Subtract the mean (A_d) from all area parameter values:

$$\Rightarrow (A_{d1} - \overline{A_d}), (A_{d2} - \overline{A_d}), (A_{d3} - \overline{A_d}), \ldots\ldots\ldots (A_{dn} - \overline{A_d}) \tag{5.3}$$

$\Rightarrow$ (327-365), (101-365), (629-365), (403-365)

$\Rightarrow$ (-38), (-264), (264), (38)

Step 3. (b). Subtract the mean (L_d) from all latency parameter values:

$$\Rightarrow (L_{d1} - \overline{L_d}), (L_{d2} - \overline{L_d}), (L_{d3} - \overline{L_d}), \ldots\ldots\ldots (L_{dn} - \overline{L_d}) \tag{5.4}$$

$\Rightarrow$ (927-1192), (2186-1192), (729-1192), (927-1192)

$\Rightarrow$ (-265), (994), (-463), (-265)

Step 4. (a). Compute the sum of the square of the differences corresponding to the design area:

$$S_A = \sum_{i=1}^{n} (A_{di} - \overline{A_d})^2 \tag{5.5}$$

$\Rightarrow$ $S_A = (-38)^2 + (-264)^2 + (264)^2 + (38)^2$

$\Rightarrow$ $(1444 + 69696 + 69696 + 1444) = 142280$

Step 4. (b). Compute the sum of the square of the differences corresponding to design latency:

$$S_L = \sum_{i=1}^{n} (L_{di} - \overline{L_d})^2 \tag{5.6}$$

$\Rightarrow$ $S_L = (-265)^2 + (994)^2 + (-463)^2 + (-265)^2$

$\Rightarrow$ $(70225 + 988036 + 214369 + 70225) = 1342855$

Step 5. Estimate *var* (A_d), *var* (A_d), and *cov* (A_d, L_d):

$$Var(A_d) = \frac{\sum_{i=1}^{n} (A_{di} - \overline{A_d})^2}{n-1} \tag{5.7}$$

$$Var(A_d) = \left(\frac{142280}{3}\right) = 47426.66 = \sim 48000$$

$$Var(L_d) = \frac{\sum_{i=1}^{n}(L_{di} - \overline{L_d})^2}{n-1} \quad (5.8)$$

$$Var(L_d) = \left(\frac{1342855}{3}\right) = 447618.33 = \sim 448000$$

$$Cov\ (A_d, L_d) = \sum_{i=1}^{n} \frac{(A_{di} - \overline{A_d}) \times (L_{di} - \overline{L_d})}{n-1} \quad (5.9)$$

Now, perform the multiplication of the corresponding pair's values obtained in steps 3. (a) and 3. (b).

- ⇨ {(-38)×(-265)}, {(-264)×(994)}, {(264)×(-463)}, {(38)×(-265)}
- ⇨ {10070}, {-262416}, {-122232}, {-10070}

Next, perform a summation of the above-obtained values to estimate $Cov\ (A_d, L_d)$.

- ⇨ (10070-262416-122232-10070)
- ⇨ (-384648)
- ⇨ $Cov\ (A_d, L_d) = (\frac{-384648}{4-1}) = -128216$

Finally, the generated dispersion matrix is:

$$Dispersion\ matrix\ (DM) = \begin{bmatrix} \frac{\sum_{i=1}^{n}(A_{di} - \overline{A_d})^2}{n-1} & \sum_{i=1}^{n} \frac{(A_{di} - \overline{A_d}) \times (L_{di} - \overline{L_d})}{n-1} \\ \sum_{i=1}^{n} \frac{(A_{di} - \overline{A_d}) \times (L_{di} - \overline{L_d})}{n-1} & \frac{\sum_{i=1}^{n}(L_{di} - \overline{L_d})^2}{n-1} \end{bmatrix}$$

$$DM = \begin{bmatrix} Var(A_d) & Cov\ (A_d, L_d) \\ Cov\ (A_d, L_d) & Var(L_d) \end{bmatrix}$$

$$DM = \begin{bmatrix} 48000 & -128216 \\ -128216 & 448000 \end{bmatrix}$$

Next, the elements of DM are encrypted using the IP vendor chosen encryption key. For example, the first 128-bit IP vendor chosen encryption key used to encrypt $Var(A_d)$ is "*aaaaabbbbbccccccd*". Similarly, the remaining 256 bits out of 384 bits are used to encrypt $Var(L_d)$ and $Cov\ (A_d, L_d)$.

$Var(A_d)^{NE}$ = 48000, (here *NE*: non-encrypted).

$Var(A_d)^{E}$ = 3b3cbe38153eaa9e73c4721249a570e1, (here *E*: encrypted).

$Var(L_d)^{NE}$ = 448000

$Var(L_d)^{E}$ = 42d86f986c536e33fe35f6963f5094d1.

$Cov(A_d, L_d)^{NE}$ = -128216

$Cov(A_d, L_d)^{E}$ = fff7b1da0a087f666f1dbcdb3f84be32.

5.2.3. Extracting secret security data from encrypted eigenvalues representing hardware design space characteristics

In the second phase, the IP vendor selected resource configurations are fed as input to the eigen decomposition block to extract the characteristics of the design space *w.r.t.* the target hardware application in terms of eigen values (λ_n) or characteristic roots. Fig. 5.4. shows the details of the proposed eigen decomposition block. Initially, square matrices are generated using the IP vendor selected resource configurations. The generated square matrices are used to compute the characteristic roots (*i.e.,* eigen roots) corresponding to selected resource configurations of the design space. The aim of this phase is to derive secret security constraints from the above-extracted characteristics eigen values. The generic representation of resource configuration square matrices for extracting eigen roots are as follows:

$$A = \begin{bmatrix} P_x^1 & P_y^1 \\ P_x^2 & P_y^2 \end{bmatrix}, B = \begin{bmatrix} P_x^3 & P_y^3 \\ P_x^4 & P_y^4 \end{bmatrix}$$

Further, eigen roots are computed as follows:

$$\Rightarrow det\left(\lambda\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} P_x^g & P_y^g \\ P_x^h & P_y^h \end{bmatrix}\right) = 0$$

$$\Rightarrow det\left(\begin{bmatrix} \lambda - P_x^g & -P_y^g \\ -P_x^h & \lambda - P_y^h \end{bmatrix}\right) = 0$$

$$\Rightarrow (((\lambda - P_x^g) * (\lambda - P_y^h)) - (P_y^g * P_x^h)) = 0 \quad (5.10)$$

For example, the generated square matrices corresponding to the 8-point DCT application are:

$$A = \begin{bmatrix} 1 & 4 \\ 1 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 8 \\ 1 & 5 \end{bmatrix}$$

Subsequently, the eigen values or characteristic roots corresponding to generated matrices (*i.e., A* and *B*) are computed. The obtained eigen values corresponding to square matrices '*A*' and '*B*' are as follows:

⇨ $det\ (\lambda I - A) = 0$

⇨ $det\left(\lambda\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \begin{bmatrix}1 & 4\\ 1 & 1\end{bmatrix}\right) = 0$

⇨ $det\left(\begin{bmatrix}\lambda - 1 & -4\\ -1 & \lambda - 1\end{bmatrix}\right) = 0$

⇨ $\lambda^2 - 2\lambda - 3 = 0$

⇨ $\lambda_1 = 3\ and\ \lambda_2 = -1$

Similarly,

⇨ $det\ (\lambda I - B) = 0$

⇨ $det\left(\lambda\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \begin{bmatrix}1 & 8\\ 1 & 5\end{bmatrix}\right) = 0$

⇨ $det\left(\begin{bmatrix}\lambda - 1 & -8\\ -1 & \lambda - 5\end{bmatrix}\right) = 0$

⇨ $\lambda^2 - 6\lambda - 3 = 0$

$\lambda_3 = 6.46\ and\ \lambda_4 = -0.46$

Further, as discussed in the overview subsection, the obtained characteristic roots corresponding to the IP vendor selected design space are encrypted using the AES-128 encryption mechanism. Fig. 5.5. Details of the AES encryption block. Next, all encrypted values are converted into binary equivalents to generate secret data corresponding to all design space characteristic roots. The final secret data corresponding to all four characteristic roots are:

$\lambda_1^{NE} = 3$

$\lambda_1^{E =}$ 917ebc40242d1c2c362b04130ae4a4a7.

λ_1^{E} = 10010001011111101011110001000000001001000010 11010001110000101100001101100010101100000100000100110000101011100100101001000100101001111.

$\lambda_2^{NE} = -1$

$\lambda_2^{E =}$ c8a3f97a72afd4dfafdfa7ea3d2ebdff.

λ_2^E = 11001000101000111111100101111010011100101010 11111101010011011111101011111101111110100111111010100011110100101110101011110111111111.

λ_3^{NE} = 6.46

λ_3^E = f88ded1ad977335078eca9d1a69b069c.

λ_3^E = 11111000100011011110110100011010110110010111 01110011001101010000011110001110110010101001110100011010011010011101100000011010011100.

λ_4^{NE} = -0.46

λ_4^E = 0eae40a1008f6b1f735c04ddb48438e9.

λ_4^E = 00001110101011100100000010100001000000001000 11110110101100011111011100110101110000000100110111011011010010000100000111000111101001.

5.2.4. Generation and embedding of final mathematical watermark

After generating the encrypted secret data corresponding to all seven characteristic parameters, the secret data is concatenated according to the IP vendor selected concatenation rule to generate a final encrypted signature. The IP vendor selected concatenation rule for the proposed approach is $Var(A_d)^E \parallel Var(L_d)^E \parallel Cov(A_d, L_d)^E \parallel \lambda_1^E \parallel \lambda_2^E \parallel \lambda_3^E \parallel \lambda_4^E$, where || is the concatenation operator. *Note:* The concatenation order can vary as per the IP vendor's choice. The final obtained encrypted signature is: "00111011001111001011111000……00011100011101001 (896-bit). Subsequently, the generated encrypted signature is transformed into covert hardware security constraints using the IP vendor chosen encoding/mapping mechanism for embedding into the design. The IP vendor chosen encoding mechanism used in the proposed approach is as follows, where C_0 and C_1 represent the covert hardware security constraints, $<L_x, L_y>$ denotes the storage variable pairs in the RAT of the design:

$$C_0 = \{(L_{2a},L_{2b}),\ a, b\ \varepsilon\ W \text{ and } (0 \le a \le m),\ (0 \le b \le n)\} \quad (5.11)$$

$$C_1 = \{(L_{2a+1},L_{2b+1}),\ a, b\ \varepsilon\ W \text{ and } (0 \le a \le s),\ (0 \le b \le t)\} \quad (5.12)$$

For incoming bit '0', covert security constraints are generated using C_0, otherwise using C_1. The upper limits of *'a'* are *'m'* and *'s'*, respectively, while the upper limits for *'b'* are *'n'* and *'t'*. The upper limit depends on the maximum number of storage variables in the target application's SDFG.

For example, using the above representation, the determined secret hardware security constraints for 8-point DCT can be obtained as follows: *<L0, L2>, <L0, L4>--<L0, L30>, <L2, L4>--<L2, L30>, <L4, L6>--<L4, L30>, <L6, L8>--<L6, L30>, <L8, L10>--<L20, L28>, <L1, L3>, <L1, L5>--<L1, L29>,<L3, L5>--<L3, L29>,<L5, L7>--<L27, L29>*. These generated security constraints are embedded into the design of the 8-point DCT application. *Note:* The generated encrypted signature is stored in a secure database, which is used to validate/verify the authenticity of the IP and original IP owner.

5.3. Demonstration

Fig. 5.6. shows the details of the proposed watermark embedding process. At first, the CDFG is scheduled using IP vendor selected resource configuration for scheduling and LIST scheduling algorithm to generate a SDFG. As discussed in earlier sections, an initial register allocation table is generated using the SDFG of the target hardware application, which is further used to

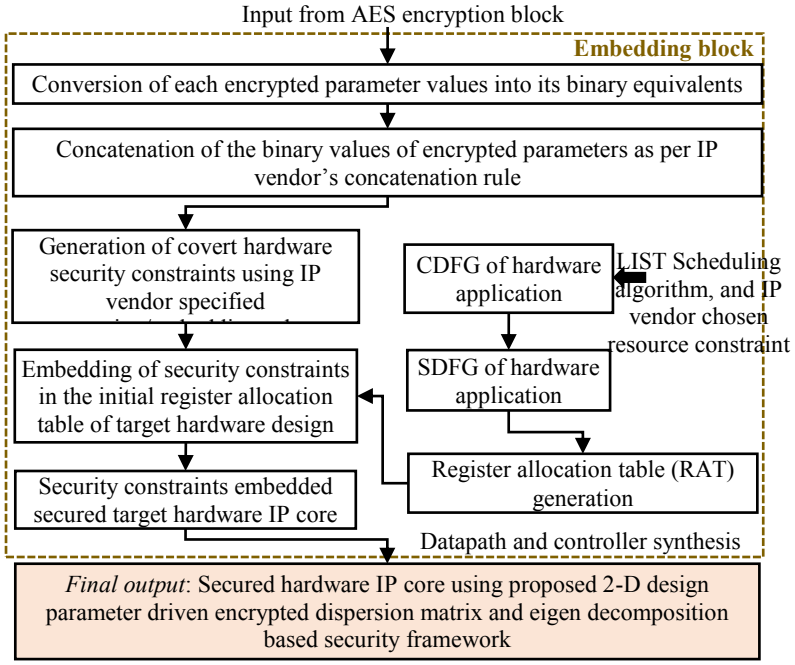


Fig. 5.6. Details of the proposed watermark embedding process

perform embedding of generated secret security constraints. The security constraints are added as additional edges in the initial RAT of the target hardware application. Embedding security constraints into the initial RAT must satisfy the security constraints obtained earlier. No two storage variables associated with the incoming security constraint can be allocated to the same register. In case of storage variable allocation conflict while embedding security constraints, either a local alteration of registers is performed, or allocation of the new register is made. Post embedding the generated hardware security constraints, the final secured RAT corresponding to the target application now contains the IP vendor's secret digital evidence. This secured RAT is further used to generate the respective secure RTL datapath. The embedding of the digital evidence acts as a detective countermeasure against IP piracy and false IP ownership by an attacker in the SoC design house. The RAT corresponding to the 8-point DCT is shown in Table 5.2 (initial and final positions of registers before and after embedding security constraints are depicted with black and red colors, respectively).

5.4. Validation and Detection

5.4.1. Validation of secured design

The proposed methodology utilizes the secret watermarking constraints of original IP vendor during the validation and detection of authentic (secured). The goal of this chapter is to provide robust validation of secured IP designs. For accomplishing this, the watermark constraints are extracted from the design-under-test (DUT) chip and matched with the originally embedded watermark security constraints of the IP design. From the extracted layout design file of the DUT chip, through reverse engineering, the IP core register transfer level (RTL) files are obtained. Finally, the watermarking constraints are extracted from the IP core RTL file (hardware description language code) for matching. In case of a complete match, validation of authentic/secured designs is complete.

5.4.2. Resolving IP ownership conflicts and detecting IP piracy

In case of IP ownership conflict, it is assumed that the attacker (in SoC house) and defender (original IP vendor) have access to the contested IP design. The

CS	Red(R)	Green (G)	Indigo (I)	Blue (BL)	Yellow (Y)	Black (B)	Violet (V)	Pink (P)	Lime (LI)	Olive (O)	Aqua (A)	Teal (T)	Grey (Gr)	Magenta (M)	Silver (S)	Khaki (K)
0	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15
1	L16/L17	L17 /L16	L2	L3	L4	L5	L6	L7	-	-	-	-	-	-	-	-
2	L24	-	L18 /L19	L19 /L18	L4	L5	L6	L7	-	L24	-	-	-	-	-	-
3	L25	-	L19	L19	L20 /L21	L21 /L20	L6	L7	L25	-	-	-	-	-	-	-
4	L26	-	-	-	L20 /L21	L21 /L20	L22 /L23	L23 /L22	-	L26	-	-	-	-	-	-
5	L27	-	-	-	L21	L21	L22 /L23	L23 /L22	-	-	L27	-	-	-	-	-
6	L28	-	-	-	-	-	L22 /L23	L23 /L22	-	L28	-	-	-	-	-	-
7	L29	-	-	-	-	-	L23	L23	-	-	-	L29	-	-	-	-
8	L30	-	-	-	-	L30	-	-	-	-	-	-	-	-	-	-

Table 5.2: Register allocation table pre and post embedding generated signature

proposed security approach facilitates a seamless IP ownership conflict resolution using the digital evidence implanted as watermarking constraints. For accomplishing this, the watermark proof is established by extracting the secret watermarking constraints from the RTL file (hardware description language code) of the IP design-under-test and matched with the original watermark constraints embedded. Only a genuine IP vendor would be able to successfully match his/her watermark security constraints with the extracted watermark constraints of the IP design to prove ownership in IP court. On the

contrary, an attacker would fail to successfully regenerate the watermark security constraints and match them with the extracted constraints to prove IP ownership in IP court. This is due to the multiple security layers, such as IP vendor selected parameters and convoluted mathematical computations in the security constraints generation process, which increases the complexity for an attacker during the watermark regeneration process. Further, to detect IP piracy, a genuine IP vendor can match the original watermark security constraints with the extracted watermark (security constraints) of the suspected chip under test. In case of a match, IP piracy is detected.

5.5. Summary

This chapter presented a 2D design parameter-based encrypted dispersion matrix, coupled with eigen decomposition security framework to secure hardware IPs against IP piracy and false IP ownership claim. The threat model assumes the IP vendor as the defender and the SoC integrator as the attacker. The methodology employs IP vendor-specific resource configurations, a dispersion matrix, eigen decomposition, and AES encryption to generate a tamper-resistant mathematical watermark signature. This watermark, embedded in hardware IP designs, provides detective control against piracy and fraudulent claim. Experimental results (discussed in Chapter 9) show the approach significantly improves security in terms of probability of coincidence, tamper tolerance, and entropy, with minimal impact on design cost overhead.

Chapter 6

Securing GLRT Cascade Hardware IP using IP Seller's Fingerprint and CIG Framework for ECG Detector

The accurate detection of cardiovascular diseases (CVDs) is critical due to the increasing incidence of heart-related issues such as arrhythmias, heart failure, etc. Timely and precise diagnosis is essential to ensure effective treatment and prevent potential complications in the future. One commonly used device for this purpose is the electrocardiogram (ECG) detector, which monitors heart activity by recording electrical signals through electrodes placed on the body. These signals, displayed as ECG waveforms, are analyzed by healthcare professionals to assess the heart's condition. The ECG detector is also a key component of cardiac pacemakers, which regulate the heart's rhythm in patients with certain cardiac conditions. For example, Table 6.1 reports the acquired ECG data of patient X from the department of non-invasive cardiology of cardiology research laboratory. *Note*: the data has been recovered ethically with the consent of the respective patient. It depicts the normal ECG parameter range [108], [109] and the acquired values. An ECG detector contains several important components that facilitate its operation, including the GLRT (generalized likelihood ratio test) unit, filtering unit, and analog-to-digital converter. Among these, the generalized likelihood ratio test (GLRT/QRS detector) unit is responsible for analyzing the QRS wave complex to estimate heart rate [110]-[113]. Due to its critical role in processing intensive computations, it is essential to design the GLRT unit as a reusable hardware IP core. HLS aids in developing the GLRT unit as a secure and efficient IP core, making it suitable for integration into SoC within ECG detector.

However, ensuring the safety and security of the GLRT hardware IP core is paramount. Pirated/unauthorized or counterfeit versions of this hardware can introduce serious risks, such as inaccurate heart data measurements or malfunctioning of the ECG device, which can have life-threatening consequences for patients. Given that the GLRT unit is also integral to the functioning of cardiac pacemakers, which remain in the body for extended

Table 6.1: Acquired ECG data of patient X, is age: 69Y 6M 3D, gender: male, report date: 05/may/2023 04:27 pm from department of non-invasive cardiology of cardiology research laboratory

ECG parameters	Normal ECG parameter range		Acquired ECG data	
	Parameter name	Value	Parameter name	Value
Heart Rate (HR)	HR	60-100 bpm	HR	75 bpm
PR Interval (PRI)	PRI	0.1 sec - 0.2 sec	PRI	0.138 sec
QRS Interval (QRSI)	QRSI	0.07 sec - 0.10 sec	QRSI	0.072 sec
QT Interval (QTI)	QTI	0.36 sec - 0.44 sec	QTI	0.34 sec
QTC Interval (QTCI)	QTCI	0.36 sec - 0.44 sec	QTCI	0.382 sec

Note: Significance of abnormal range (>upper limit): Cardiovascular disorder such as arrhythmias, atrial enlargement, Wolff-Parkinson-white syndrome), myocardial ischemia, ventricular hypertrophy, heart failure, hypertrophic cardiomyopathy long QT syndrome, etc.

periods, the importance of using secure and authenticated versions of this hardware cannot be overstated. Furthermore, counterfeit hardware can also lead to fraudulent ownership claim, further complicating the medical device industry. Therefore, securing the GLRT hardware design is critical for the safe operation of these life-saving devices.

This chapter presents a novel secure hardware IP of GLRT cascade using color interval graph (CIG) based embedded fingerprint, for ECG detector. The proposed approach discusses designing GLRT micro and GLRT cascade hardware IP core for ECG detectors for the first time in literature. The first section of the chapter outlines the overview, threat model and motivation of the proposed approach. The second section discusses the details of proposed CIG-based secure HLS flow using IP seller's fingerprint for generating secure GLRT cascade hardware IP. Following this, the third section illustrates the IP piracy detection and resolution of false IP ownership claim process. Lastly, the fourth section provides the chapter's conclusion..

6.1. Overview, Threat Model, and Motivation

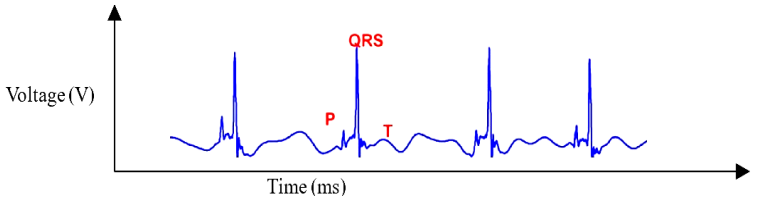


Fig. 6.1. ECG wave recorded through electrode for reference; PR interval - duration between onset of atrial depolarization and ventricular depolarization; QT interval - duration between onset of ventricular depolarization and end of ventricular repolarization

6.1.1. Overview of ECG detector (GLRT cascade)

An ECG detector consists of several key components, including a filtering unit, noise detector, GLRT cascade unit, summer, and threshold processing unit. It is primarily used to detect ECG parametric data (shown in Table 6.1), which helps evaluate heart activity by analyzing signals captured through electrode leads. The filtering unit generates both monophasic (single-direction) and biphasic (two-direction) pulse outputs, which are then sent to the GLRT unit to identify the presence of the R wave, a key part of the heart's electrical cycle. The GLRT cascade outputs are summed and compared using a threshold processing function, allowing the device to distinguish between cardiac signals and various noise interferences, such as muscle artifact, power line interference, and baseline wander. An ECG waveform, as shown in Fig. 6.1, consists of several segments: the P-wave, QRS complex (comprising Q, R, and S waves), and the T-wave. These waves, along with intervals like the PR interval (from atrial to ventricular depolarization) and the QT interval (from ventricular depolarization to repolarization), provide critical insights into heart function. Accurate detection of the QRS complex, particularly the R wave, is a challenge due to the heart's dynamic behavior and physiological variations, making the GLRT unit vital for reliable heart signal interpretation in both diagnostics and cardiac pacemaker devices.

GLRT overview: The GLRT cascade unit processes filtered signals to evaluate heart rate by analyzing outputs from a wavelet filter bank (WFB) and

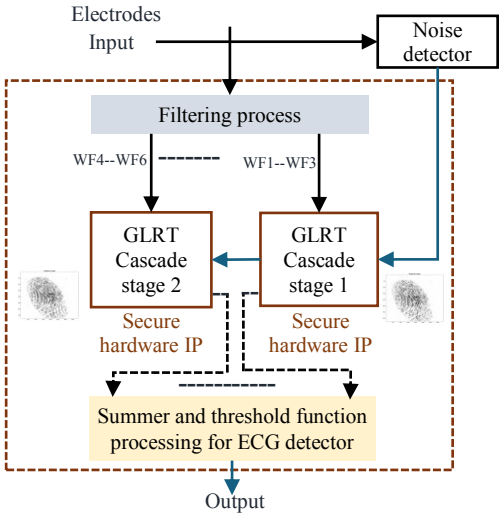


Fig. 6.2. Proposed secure hardware IP of GLRT cascade for ECG detector

detecting the presence of the QRS complex. This is achieved through maximum likelihood estimation, using hardware components like delays, multipliers, and adders. As illustrated in Fig. 6.2, each GLRT cascade unit in an ECG detector consists of two stages, each containing three GLRT micro units. The CDFG ([110]-[112]) outlines the operational flow of these units, as shown in Fig. 6.3 and Fig. 6.4. In stage 1 of the GLRT cascade, primary inputs from the filter unit (WF1, WF2, WF3) are processed by the three GLRT micro units. Each unit uses specific coefficients (C12, C22, C32 for micro unit 1; C11, C21, C31 for micro unit 2; and C13, C23, C33 for micro unit 3) to perform computations. The output of each micro unit is passed sequentially to the next unit. Similarly, stage 2 of the GLRT cascade processes inputs WF4, WF5, and WF6 in the same manner. Finally, the outputs from the GLRT cascade stages are summed and compared using a threshold processing unit,

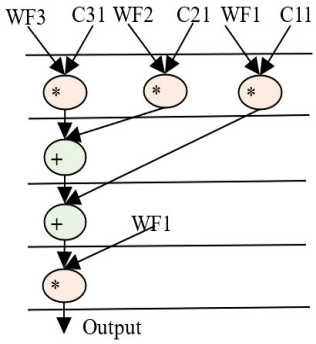


Fig. 6.3. GLRT DFG of proposed micro IP

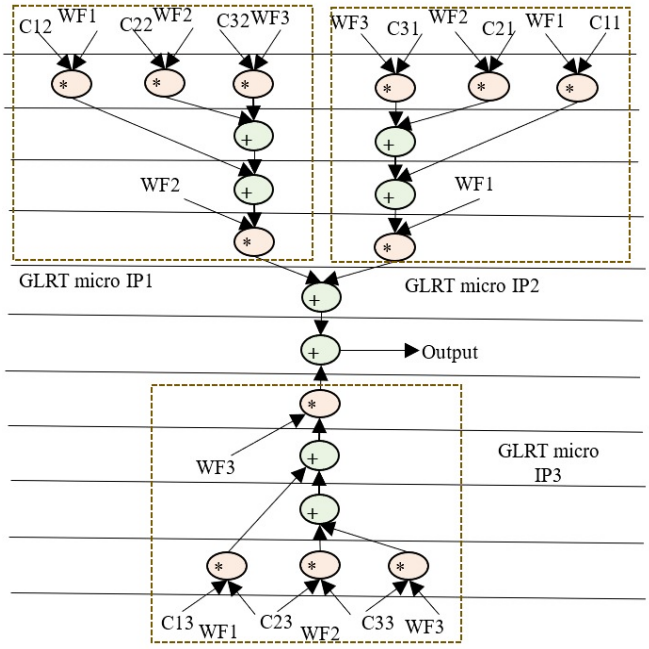


Fig. 6.4. GLRT cascade DFG of proposed macro IP

which helps differentiate between cardiac and noise signals.

6.1.2. Threat model

The proposed approach offers a security measure to detect and prevent the integration of pirated GLRT IP cores into SoC designs for ECG detectors. The presence of pirated/counterfeited GLRT hardware IP in an ECG detector can pose serious risks to patient safety, as these versions are often untested and may contain malicious logic. The attacker responsible for creating pirated IPs are typically competitors aiming to harm the reputation of the original vendor. This not only undermines the credibility of the legitimate IP vendor but also endangers patients who rely on accurate and reliable heart monitoring devices.

6.1.3. Motivation

Designing the GLRT unit of an ECG detector as a reusable hardware IP core is essential due to its role in detecting the QRS complex through computationally intensive tasks on filtered data. Further, ensuring the safe and reliable performance of the GLRT hardware IP core is crucial for accurate detection of cardiac signals and ECG parameters. Pirated GLRT hardware IPs pose significant risks, as they may contain malicious logic and are not subjected to rigorous testing, leading to potentially fatal consequences for patients. This issue arises due to the involvement of untrustworthy third-party vendors and manufacturers, increasing the risk of security breaches. Additionally, the ECG detector is a critical component in cardiac pacemakers, making the use of secure GLRT hardware IP cores essential for their proper functioning. The proposed approach provides a detective security measure, allowing only authentic IP versions to be integrated into the system.

6.2. CIG-based Secure HLS Flow Using IP Seller's Fingerprint for Generating Secure GLRT Cascade Hardware IP

6.2.1. Deriving the GLRT dataflow graph from its transfer function

The GLRT dataflow graph is initially extracted from its corresponding transfer function. The transfer function of GLRT using Mallat's algorithm is adopted from [110]-[112]:

$$Z(a) = s^T(n)H(H^TH)^{-1}H^Ts(n) \quad (6.1)$$

where, *s(n)* is the input to the filtering unit and *H* is the linear combination matrix of the representative function. Here, *H* is a 1×6 matrix, *s(n)* is a 1×6 matrix, $s^T(n)$ is a 6×1 matrix and (H^TH) is a 6×6 matrix [118]. The extracted CDFG of GLRT and its cascaded representation are illustrated in Figures 6.3 and 6.4 respectively. WF1, WF2, and WF3 are the outputs of the filtering unit, and C11-C33 is the coefficients of the linear combination matrix H. The extracted CDFG is fed as input to the scheduling allocation and binding block of HLS to generate its corresponding SDFG. The details of the proposed CIG generation from SDFG and fingerprint biometric based hardware security are discussed in the next subsection.

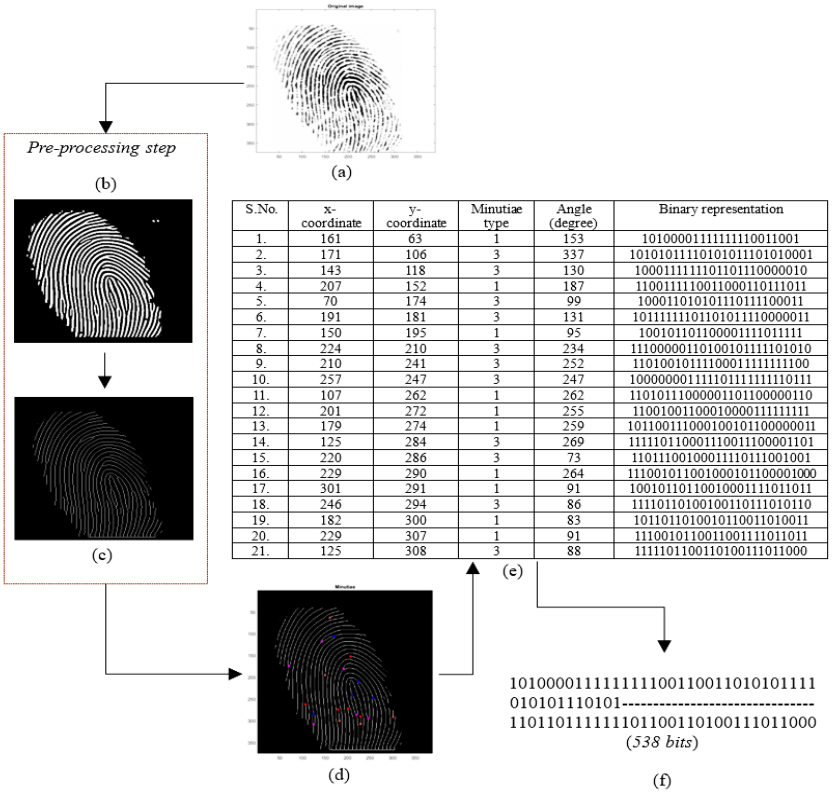
6.2.2. Creation of covert fingerprint biometric watermark signature

A fingerprint biometric-based covert signature/watermark is generated using the IP vendor's fingerprint to ensure security of GLRT IP core design. Fig. 6.5 shows the fingerprint watermark generation process. The process begins with capturing the IP vendor's fingerprint through a high-quality optical scanner in a secure, dust-free environment. The fingerprint must be clean, free from injuries, and properly scanned to capture all critical features like ridge angles and bifurcations. After the capture, the fingerprint undergoes preprocessing steps such as binarization, thinning, and Fast Fourier Transform (FFT) enhancement. These steps are crucial for improving the fingerprint image, enabling the identification of unique minutiae points—key elements that distinguish fingerprint features like ridges and valleys. Binarization converts the fingerprint image into a binary form, where pixels are assigned values of 0 or 255 based on a threshold, while thinning reduces the ridge line thickness to enhance clarity. FFT enhancement helps in reconnecting broken ridges and improving overall fingerprint structure. Once preprocessing is complete, minutiae points are generated and classified based on their features, such as bifurcation or ridge ending. The fingerprint is then placed under an IP vendor-specified grid to extract vital parameters: *x* and *y* coordinates, minutiae type, and ridge angle. The proposed approach employs a crossing number (C_r) algorithm to extract the respective minutiae points [119]. Fig. 6.6 shows the

neighboring image pixels of image pixel *I*. The crossing number corresponding to a fingerprint image pixel *I* is formulated as [119]:

$$C_r = 0.5 \sum_{k=1}^{8} |I_k - I_{k+1}| \tag{6.2}$$

where I_k is the neighborhood pixel value of pixel *I* (depicted in the above 3x3 pixel matrix). A minutiae point is classified into (a) bifurcation and (b) ridge ending. The minutiae point with crossing number 3 is a bifurcation, and crossing number 1 is a ridge ending. The four crucial parameters (*x* and *y* coordinates, minutiae type, and ridge angle) corresponding to each generated minutiae point is extracted. Each of these values is then converted into binary format and concatenated using a specific rule defined by the IP vendor (*x*-coordinate |+| *y*-coordinate |+| minutiae type number |+| ridge angle, where |+| is a concatenation operator). This concatenation of binary strings, arranged by minutiae point number, generates the final fingerprint-based digital template



S.No.	x-coordinate	y-coordinate	Minutiae type	Angle (degree)	Binary representation
1.	161	63	1	153	10100001111111110011001
2.	171	106	3	337	10101011110101011101010001
3.	143	118	3	130	10001111111011011100000010
4.	207	152	1	187	11001111100110001101111011
5.	70	174	3	99	10001101010111011100011
6.	191	181	3	131	10111111101101011110000011
7.	150	195	1	95	10010110110000111011111
8.	224	210	3	234	11100000110100101111101010
9.	210	241	3	252	11010010111100011111111100
10.	257	247	3	247	10000000111101111111110111
11.	107	262	1	262	11010111000001101100000110
12.	201	272	1	255	11001001100010000111111111
13.	179	274	1	259	10110011100010010110000011
14.	125	284	3	269	11111011000111001100001101
15.	220	286	3	73	11011100100011110111001001
16.	229	290	1	264	11100101100100010110000100
17.	301	291	1	91	10010110111001000111011011
18.	246	294	3	86	11110110100100110111010110
19.	182	300	1	83	10110110100101100110100
20.	229	307	1	91	11100101100110011111011011
21.	125	308	3	88	11111011001101001110110000

Fig. 6.5. Proposed fingerprint digital template generation process extracted from captured IP vendor's fingerprint, (a) input IP sellers fingerprint image, (b) binarized fingerprint image, (c) thinned fingerprint image, (d) minutiae points generation on fingerprint image, (e) details of generated minutiae points parameters, (f) generated fingerprint biometric based digital template.

I_4	I_3	I_2
I_5	I	I_1
I_6	I_7	I_8

Fig. 6.6. Image matrix representing neighboring pixels of image pixel I

or covert signature (shown in Fig. 6.5). The final watermark/signature serves as a unique identifier for the IP vendor, ensuring that only the authentic hardware IP core passes validation checks.

Note: The fingerprint biometric is securely stored in an encrypted format, removing the need for re-capturing during the detection and validation process. During the IP piracy detection process, evasion by an attacker is not possible as he/she is unable to regenerate the fingerprint biometric digital template for embedding into his/her fake versions, thus failing in the hardware IP core authentication process. Thus, the proposed approach effectively safeguards the GLRT hardware IP core by embedding the fingerprint biometric, ensuring that only legitimate versions are integrated, and preventing hardware IP piracy.

6.2.3. Generation and embedding of watermarking constraints

The proposed secure HLS flow uses a colored interval graph of the HLS framework and fingerprint biometric based hardware security methodology to

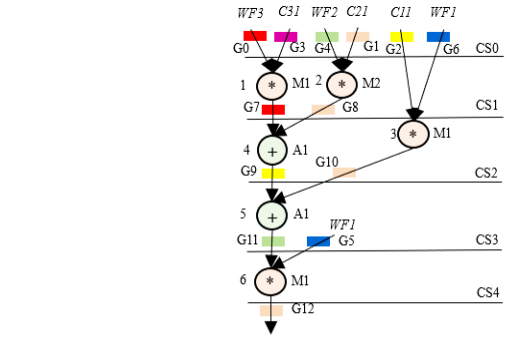


Fig. 6.7. (a) SDFG of GLRT micro IP using one adder (+) and two multipliers (*) post embedding fingerprint signature , (b) SDFG of GLRT cascade macro IP scheduled using three multipliers and two adders post embedding fingerprint

generate a secure GLRT hardware IP core for ECG detector. As discussed above in the overview subsection, the transfer functions of GLRT micro and GLRT cascade generates its corresponding CDFG. The generated CDFG is fed as the primary input to the scheduling, allocation, and binding (SAB) block of HLS with IP vendor specific resource constraints and scheduling algorithm (such as LIST scheduling) as additional inputs. The final output of the SAB block is an SDFG with registers allocated in different control steps. An SDFG corresponding to micro GLRT and GLRT cascade with register allocations is depicted in Figures 6.7 (a) and (b), respectively. Registers are storage units that store input, intermediate, and output variable values during the computation of the GLRT unit. Post SDFG creation, an initial CIG is generated using register allocation information of the created GLRT SDFG. CIG is a graphic representation of register allocation information of the GLRT SDFG. The initial CIG (i.e., pre-embedding fingerprint signature based secret security constraints) corresponding to micro GLRT and GLRT cascade are shown in Figures 6.8 (a) and (b). The obtained initial CIG is used for performing the embedding of fingerprint based secret hardware security constraints. The presence of the IP vendor's fingerprint based covert signature into the design of GLRT hardware IP core guards it against piracy and false claim of IP ownership problems. The obtained digital fingerprint template (comprising of 227 number of zeros and 311 number of ones) is initially converted into secret hardware security constraints using IP vendor specific mapping or embedding rule. The IP vendor specific embedding rule is as follows:

$$S_0 = (G(2s), G(2r)) \tag{6.3}$$

where, 2s, 2r are whole numbers and (0 ≤s ≤9), (1≤ r ≤20)

$$S_1 = (G(2s+1), G(2r+1)) \tag{6.4}$$

where (0≤ s ≤18), (1≤ r ≤19)

The symbols G(2s) and G(2r) represent the storage variables in the scheduled data flow graph. The limits of *s* and *r* depend on the maximum storage variables used in GLRT SDFG. For bit 0, covert security constraints are generated using S_0, otherwise using S_1.

The storage variables in the SDFG are sorted in ascending order and stored in a list. Post sorting, the mapping/embedding rule is applied to generate the encoded security constraints using equations (6.3) and (6.4), respectively. Therefore, the obtained secret security constraints are as follows: (G0, G2), (G0, G4)--(G0, G40), (G2, G4)--(G2, G40), (G4, G6)--(G4, G40), (G6, G8)--(G6, G40), (G6, G8)--(G6, G40), (G8, G10)--(G18, G40), (G1, G3), (G1, G5)--(G1, 39),(G3, G5)--(G3, G39),(G5, G7)--(G37, G39).

Hence, corresponding to 227 number of zeros, 227 security constraints of storage variables are obtained (ranging from (G0, G2)---(G18, G40)). Similarly, corresponding to 311 number of ones, 311 security constraints of storage variables are obtained (ranging from (G1, G3)---(G37, G39)). *Note*: As evident, these secret security constraints are extracted using the mapping/embedding rule of the IP vendor and is a function of the fingerprint signature obtained. The obtained secret hardware security constraints are embedded into the initial CIG of the micro GLRT and GLRT cascade. No

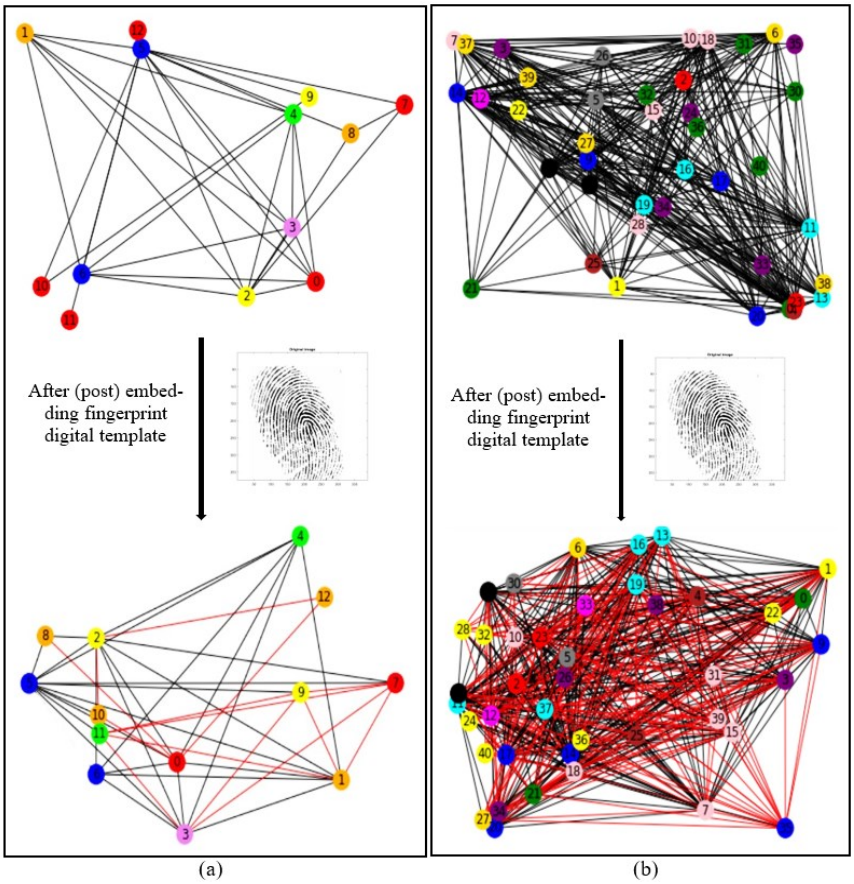


Fig. 6.8. (a) CIG (pre and post embedding fingerprint) corresponding to secure GLRT micro IP core, (b) CIG (pre and post embedding fingerprint) corresponding to secure GLRT macro IP core

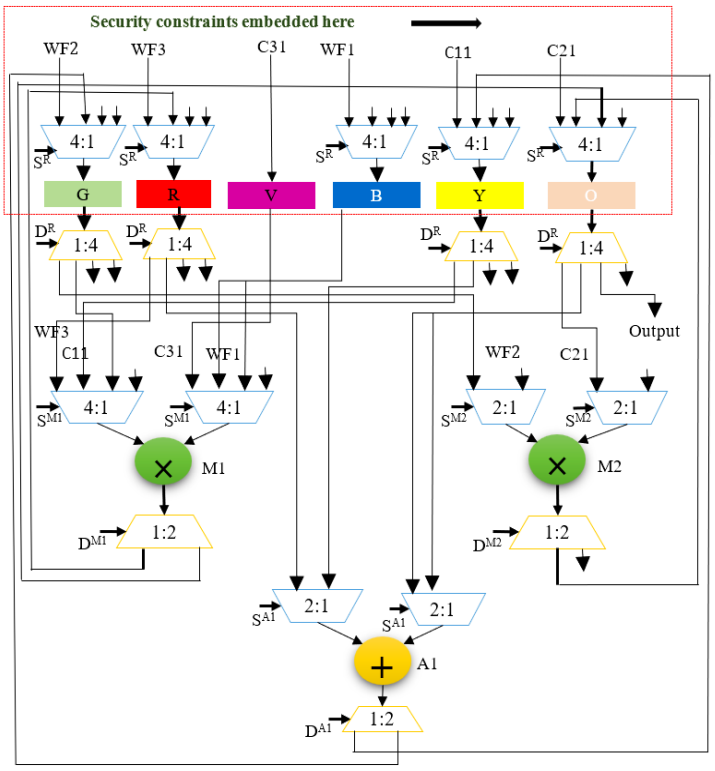


Fig. 6.9. Secure RTL design of GLRT cascade macro IP core with CIG based embedded fingerprint

additional changes are made if an edge is already present in the CIG corresponding to incoming secret security constraints. However, in the absence of the edge between storage variables of incoming hardware security constraints, an additional edge in the corresponding CIG is added (depicted in red color in Fig. 6.8 (a) and (b)). *Post embedding of an additional artificial edge, a color swapping (i.e., local alteration) between registers is performed if the additional embedded edge's storage variables are allocated on the same colored register. The storage variables of embedded security constraints cannot be allocated on the same colored register*. Moreover, a new color register is also allocated to resolve the raised conflict if no local alteration is possible. Figures 6 (a) and (b) depict the final CIG post embedding all determined security constraints corresponding to micro GLRT and GLRT cascade, respectively. As shown in Fig. 6.8 (a), storage variables G0 and G12 are allocated on the red color register. However, due to additional artificial edge (G0, G12) (*i.e.,* security constraints), the color of G12 is changed to orange. Similarly, all required alterations are performed post-embedding all security constraints in the CIG of micro GLRT and GLRT cascade. *Note*: Artificial edges (imposed security constraints) do not alter the functionality of an IP core because these artificial edges are only responsible for local

alteration of registers corresponding to storage variable assignment. These imposed security constraints do not affect the dataflow connectivity of the SDFG corresponding to the IP core functionality. Therefore, due to embedding of the security constraints, the functional units and its interconnectivity remain unaffected (only the register sharing is locally altered). Further, Figures 6.7 (a) and (b) illustrate the SDFG of micro GLRT and GLRT cascade post-embedding hardware security constraints. Moreover, Fig. 6.9 and 6.10 depict the secure RTL datapath corresponding to the micro GLRT hardware IP core and GLRT cascade macro hardware IP core for the ECG detector. The final generated RTL of GLRT hardware IP core contains security constraints in the form of altered register colors.

6.3. Identifying Pirated GLRT Hardware IP Cores for ECG Detectors

The presence of fingerprint biometric based digital evidence helps in making a clear distinction between authentic and fake (*i.e.,* pirated) versions of GLRT hardware IP cores. While conducting piracy detection, the security constraints corresponding to the IP vendor's fingerprint template are initially regenerated. The regenerated information is matched with the register allocation

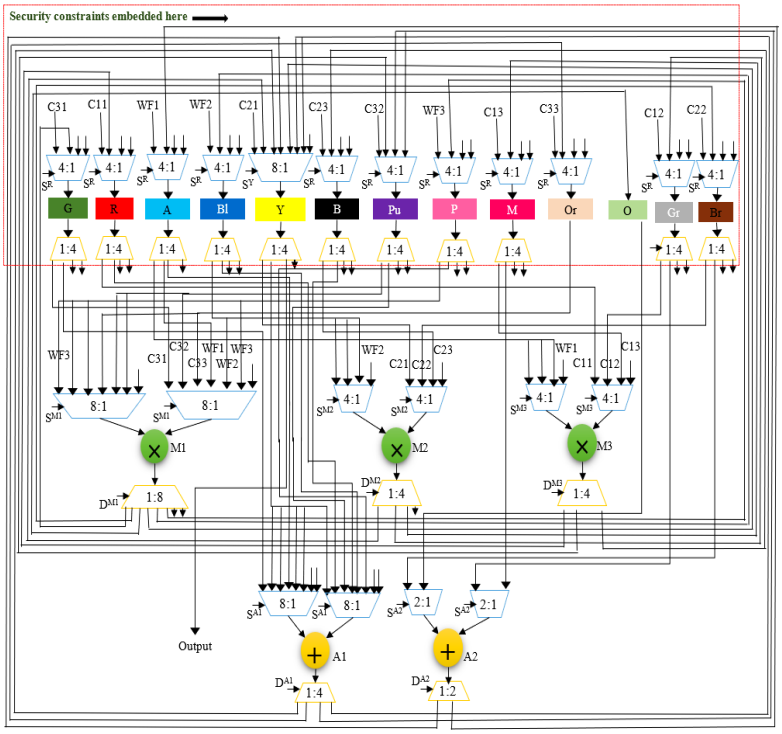


Fig. 6.10. Secure RTL design of GLRT cascade macro IP core with CIG based embedded fingerprint

information of GLRT hardware IP's RTL datapath under test. The authentic version will exhibit a complete matching of security constraints; otherwise, the version is considered pirated.

It is crucial to protect the designed GLRT hardware IP core for the ECG detector from an adversary's false claim of IP ownership problem. An adversary located at an untrustworthy offshore design or fabrication house can falsely claim hardware IP ownership right. The IP vendor's digital fingerprint template embedded in the design (*i.e.,* CIG) of GLRT hardware IP core safeguards it from the adversary's false claim of IP ownership. Authentic IP vendor can easily nullify the false claim of IP ownership by matching the embedded digital fingerprint constraints (in the GLRT RTL IP under test) with the original one (minutiae points pre-stored in a secure database in an encrypted format or can be regenerated as explained above in the previous subsection). In case of complete matching, ownership is awarded to the original IP vendor. *Note*: In the proposed approach, the embedded fingerprint biometric is stored in a secure database in an encrypted format for detection and validation process later. Therefore, recapturing of the fingerprint biometric data is not required for the detection and validation process.

6.4. Summary

This chapter presented a secure design methodology for GLRT cascade hardware IP core, integrated with CIG-based fingerprint biometric. Embedding the IP seller’s fingerprint based watermark ensures clear distinction from pirated versions, allowing only authenticated cores to be integrated into ECG detectors. This is critical for patient safety, as the use of counterfeit versions could lead to hazardous outcomes. The proposed approach ensures reliable and secure ECG detector functionality, which is essential for accurately analyzing a patient's heart condition and preventing risks associated with counterfeit components. The experimental evaluation of the proposed approach is discussed in Chapter 9.

Chapter 7

Exploiting Voice Biometric-Based Watermarking Framework for Securing Hardware IP Cores

IP cores play a crucial role in numerous industries, including consumer electronics, healthcare, information technology, military, and aerospace. The current electronic design cycle often involves offshore entities to reduce design complexity, costs, and time-to-market. However, this outsourcing increases the risk of hardware security threats, including IP piracy, counterfeiting, and fraudulent ownership claim. These threats not only pose a risk to an IP owner's revenue but also tarnish their reputation. Furthermore, counterfeit IP integrated into the supply chain can negatively impact the performance and reliability of systems, ultimately affecting end users. Hence, protecting IP cores from these hardware threats is essential.

To mitigate risks of piracy and counterfeiting, hardware watermarking [31]-[36], [42] encryption based security approaches [38], [39], [43], and steganography [37] techniques have been discussed in the literature. These techniques typically embed a vendor's signature into the IP design. However, using random binary sequences or physical parameters, like integrated circuit images, for generating signature may not accurately represent the vendor's identity, leading to potential ownership conflict. To address this, this chapter proposes a novel IP protection method utilizing the IP vendor’s voice biometric watermark signature. This approach leverages unique voice characteristics such as jitter, shimmer, pitch, and intensity to generate a distinctive signature. The voice-based signature is embedded as a hardware security constraint within the IP design, ensuring robust security (detective control) against piracy and fraudulent claim of ownership. The secure design allows for straightforward detection and verification of unauthorized IP use, with the original voice sample securely stored for future authentication. Moreover, the proposed voice biometric based security approach provide more robust security than traditional biometric based security approaches [40], [41], [44] in terms of stronger tamper tolerance and reduced probability of coincidence, discussed in Chapter 9 of the thesis.

The first section of the chapter discusses the motivation and benefits of voice biometric based watermarking framework. The second section outlines the threat model involved and overview of the proposed approach. Subsequently, the third section discusses the details of proposed voice biometric based hardware watermarking framework. Following this, the fourth section illustrates the demonstration of the proposed approach. Next, IP piracy detection and resolution of false IP ownership claim process is explained in fifth section. Finally, the sixth section provides the chapter's conclusion.

7.1. Motivation and Benefits of Voice Biometric-Based Watermarking Framework

The rise in fraud attacks, which surged by 269% over four years, and the lack of adequate security features have led to an increased focus on voice biometrics over traditional methods like fingerprints, facial recognition, and palmprints. The voice biometrics market is expanding rapidly, projected to grow from $1.1 billion in 2020 to $3.9 billion by 2026. This growth has drawn the attention of security researchers, especially in the field of IP protection [102], [121]. Voice biometrics offers several key advantages over other biometric techniques.

First, it enables the extraction of numerous features from voice signals at various timestamps, producing a vast number of hardware security constraints, which is one of the primary limitation of traditional biometric based watermarking approaches [40], [41], [44]. This makes it highly robust compared to other biometric systems like fingerprint-based approach [40], which involve complex processes such as minutiae generation and filtering. Second, voice biometrics has a lower implementation complexity. Unlike fingerprint recognition [40], which requires several preprocessing steps like binarization and image enhancement, voice biometrics involves a simpler process of feature extraction. Increasing the number of timestamps for voice feature analysis allows for a larger, more secure signature without added complexity. Third, detecting and verifying a voice signature is seamless, as it relies on a pre-stored voice sample, eliminating the need for recapturing data.

Additionally, voice biometrics meet ISO/IEC standards by supporting unique, revocable, and irreversible digital templates, ensuring secure and reliable authentication ((i) *Unlikability*- Voice biometric supports the generation of diverse (exclusive) voice signature digital templates from the same voice sample. (ii) *Revocability*- the previously generated voice biometric template can easily be replaced with a new one for the same voice sample. It is possible because of the dependence of the generated voice signature template on variation in the number of features, selected feature set, concatenation order, and mapping rule. (iii) *Irreversibility*- it is extremely challenging to recover the original voice sample data from the generated digital template as this requires complete knowledge of the concatenation rule, selected feature set, and the number of timestamps chosen to extract the features.). Lastly, voice biometrics is a contactless technology, immune to environmental factors like dirt or physical injuries that can affect other methods, such as fingerprint [40] and facial biometric [41]. While facial recognition and palmprints are also contactless, they have limitations, such as aging effects on facial features [41] and grid size requirements for palmprint [44], making voice biometrics a superior choice for secure IP protection.

7.2. Threat Model and Overview

7.2.1. Threat model: attacker's and defender's capabilities

Reusable IP cores from third-party vendors face significant hardware security risks such as counterfeiting, piracy, and fraudulent IP ownership claim, driven by the globalized nature of the design supply chain. In counterfeiting, counterfeit or substandard IPs are introduced into the supply chain under the original vendor's brand, damaging both revenue and reputation. Piracy or cloning occurs when a dishonest user, such as a SoC integrator or foundry, steals the vendor's IP and sells unauthorized copies under a different brand. Additionally, adversaries may falsely claim ownership of the IP, leading to legal and financial disputes. To combat these threats, a robust solution involving voice biometric-based watermarking is proposed to detect IP misuse and resolve ownership conflicts. Attackers, including foundries or IP brokers, may have access to the IP and could attempt piracy or fraudulent ownership claims. They might also be aware of the biometric watermarking and attempt

to spoof the voice sample or forge the signature to bypass detection. In response, the IP owner embeds an encrypted voice signature to counteract these threats. The owner is also equipped to defend against spoofing attempts or efforts to compromise the stored voice sample, ensuring robust IP protection.

7.2.2. Overview

The proposed method leverages unique features of a voice sample to generate a distinctive signature for IP core authentication and verification. The process begins by recording a voice sample and converting it into a spectrogram, from which features such as jitter, shimmer, pitch, and intensity at various timestamps are extracted. These feature values are then transformed into binary form and concatenated to create a unique voice signature template. This template is further encoded into hardware security constraints, which are integrated into the IP core design during the HLS process. The robustness of the signature can be enhanced by extracting additional pitch and intensity features from the spectrogram. For verification, a pre-stored voice template is used, ensuring that the defining characteristics—pitch, intensity, jitter, and shimmer—remain consistent with the original vendor's voice sample. The process starts by converting the target DSP application's algorithmic description, such as a transfer function, into a CDFG, which is then scheduled using LIST scheduling to produce a SDFG. Subsequently, a voice biometric signature is generated using the proposed algorithm and encoded as hardware security constraints. These constraints are embedded into the DSP design through the HLS framework, providing robust protection against threats.

7.3. Voice Biometric-Based Watermarking Framework

Fig. 7.1 highlights the detailed flow-chart of the proposed voice biometric based hardware security approach.

7.3.1. Introduction to voice biometric

Voice biometrics can be categorized into two types: text-independent, where no audio template is stored, and text-dependent, where a voice sample is stored for authentication purposes. The proposed method utilizes text-dependent voice biometrics. Each voice sample contains unique traits

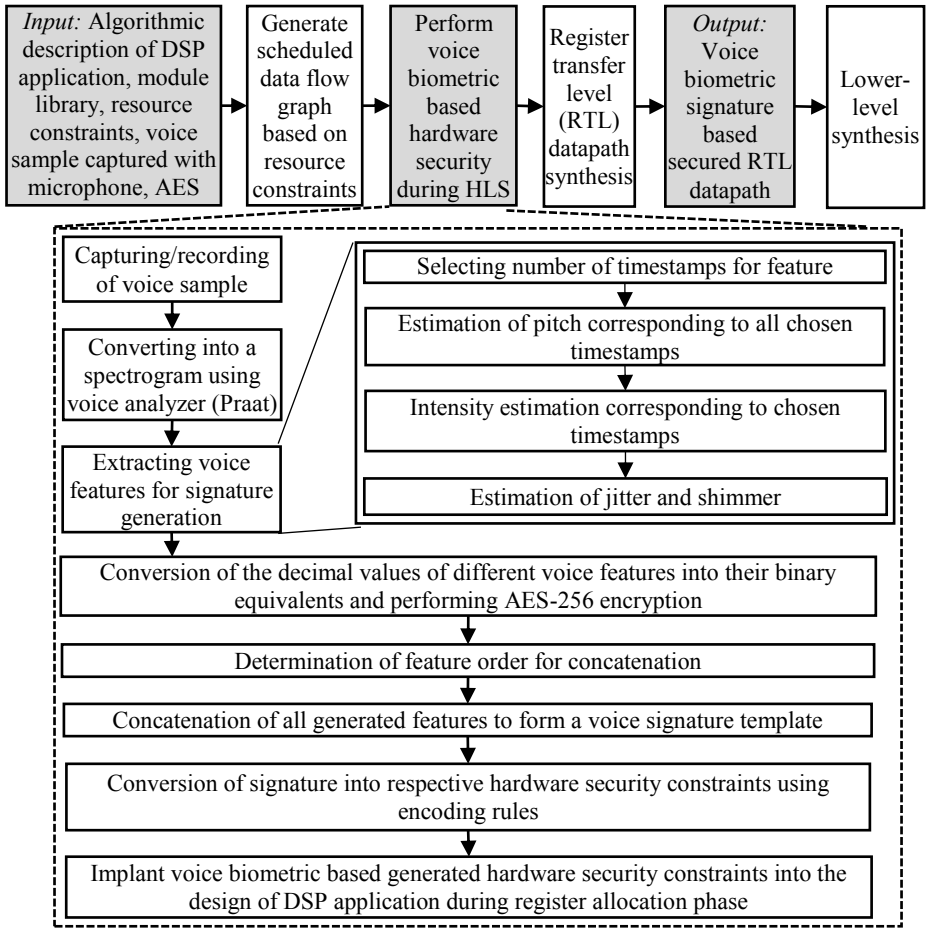


Fig. 7.1. Detailed flow-chart of the proposed voice biometric based hardware security approach

influenced by both behavioral speech patterns, such as speaking speed, and physiological factors, like the shape of the throat and mouth. Key acoustic features for voice biometrics include jitter and shimmer, which are primarily used for speaker verification. Jitter refers to the variation in pitch frequency from cycle to cycle, while shimmer relates to the variation in amplitude. These features, along with pitch, which distinguishes high and low sounds, and intensity, which measures sound energy, play a crucial role in voice-based identification [122], [123]. The proposed approach leverages these voice features—jitter, shimmer, pitch, and intensity at different timestamps—to create a unique watermark signature template for securing IP cores.

7.3.2. Inputs and Outputs

The primary inputs are: (a) transfer function of the hardware application /CDFG/C/C++ code, (b) module libraries, (c) IP vendor's voice sample. The intermediate outputs are as follows: (a) covert hardware watermarking/security constraints, and (b) a CIG (based on the RAT) of the target application. The final output is a voice biometric secure hardware IP design.

7.3.3. Capturing and pre-processing IP seller's voice biometric sample

The proposed approach begins by capturing the IP vendor's voice sample using a microphone. Once recorded, the voice sample is transformed into a spectrogram and analyzed through speech analysis software. In the proposed approach, the '*Praat application*' is used to extract biometric security features. For example, a voice sample labeled '*voice-001*', corresponding to the utterance "*Gopal*", is demonstrated. Fig. 7.2 (a) shows a spectrogram of a voice sample *'voice-001*' (corresponding to the utterance '*Gopal'*) used in the proposed approach for demonstration. Different timestamps are selected based on the required signature strength, with fewer timestamps for lower strength and more for higher strength, showing the approach's scalability for various applications. In this demonstration, 15 timestamps are used to capture pitch and intensity values from the voice sample. The goal is to extract pitch and

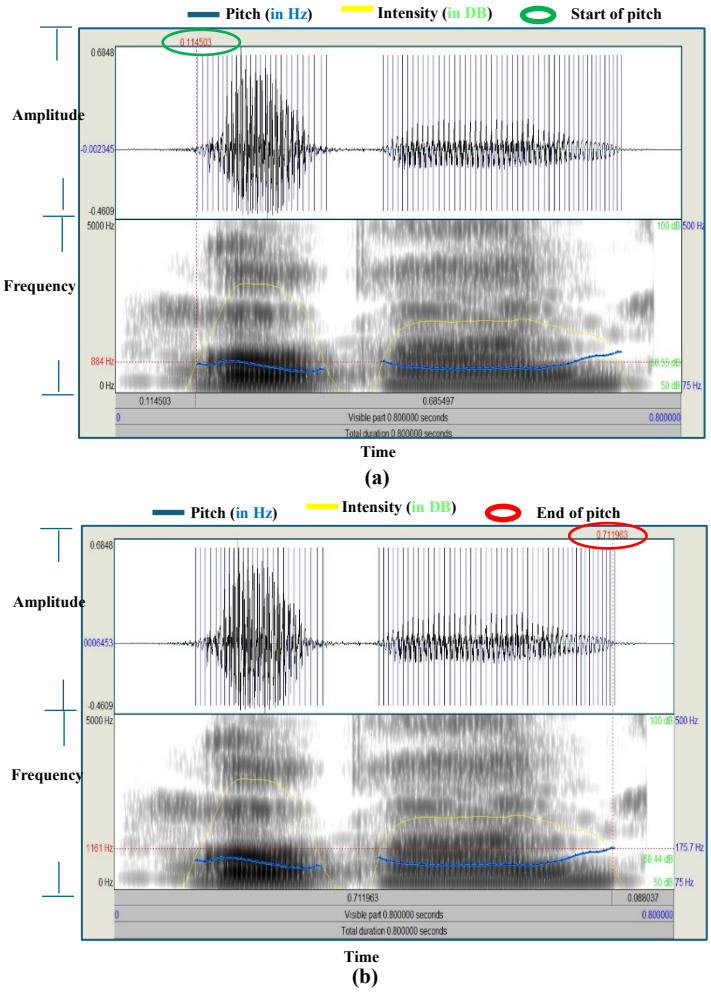


Fig. 7.2. Spectrogram of voice sample (voice-001) using a speech analyser tool showing the range of pitch and intensity with indicating (a) starting point of pitch (b) end point of pitch

intensity at each timestamp to create a unique signature. Spectrograms are preferred over Mel-frequency cepstral coefficients (MFCC) in this approach, as they more effectively represent voice data, including key features like pitch, intensity, jitter, and shimmer, making them more suitable for generating robust watermark signature.

7.3.4. Identifying distinct voice features for watermark signature creation

7.3.4.1. Timestamp analysis for pitch value determination

The proposed approach considers different features of a voice sample, *viz.* jitter and shimmer, along with pitch and intensity values at different timestamps, for corresponding voice signature template generation. The pitch and intensity values are extracted for the IP vendor selected timestamps (assumed as N_1, N_2 = 15 using the voice analyzer software (Praat) application. The blue and yellow lines in Fig. 7.2. (a) and (b) represent the pitch and intensity variation curve of the voice signal at different time instants. The initial and final position of both the pitch and intensity variation curve on the spectrogram is noted to determine different unique timestamps for the feature extraction. The pitch and intensity values at different timestamps represent the first and second subsets of features used to generate our voice signature template. Different timestamps for extracting various pitch and intensity values are determined below. In the proposed approach, pitch values are extracted at the N_1 number of timestamps (selected by the IP vendor). In order to determine the different timestamps, the following steps are performed.

(a) The total duration of voice pitch (P_T) is calculated using the following equation:

$$P_T = p_e - p_s \tag{7.1}$$

Where p_e and p_s indicate the end time and start time, respectively, on the voice pitch variation curve.

(b) We define a step size of different timestamps for pitch features extraction to be Δp which is calculated as follows.

$$\Delta p = P_T/N_1. \tag{7.2}$$

(c) The different values of timestamps are determined using the following equation.

$$t_i = V_s + i*\Delta p;\ (0< i \leq N_1\text{-}1) \quad (7.3)$$

Where V_s is the starting position of the pitch in the spectrogram shown in Fig. 7.2. (a) and (b).

Here, the IP vendor selected the number of timestamps, N_1 is $N_1=15$. The initial (p_s) and final timestamps (p_e) on which the pitch values recorded are 0.11 and 0.71 (corresponding to the voice-001 obtained from Fig. 7.2. (a) and (b), respectively). Hence, the time duration of voice pitch (P_T) is calculated to be (0.71-0.11) = 0.6, and the step size is computed to be 0.04 using (7.2).

7.3.4.2. Timestamp analysis for intensity value determination

The intensity values are extracted at the N_2 number of timestamps. In order to determine the different timestamps, the following steps are performed.

(d) The total duration of voice intensity (I_T) is calculated using the following equation:

$$I_T = I_e - I_s \quad (7.4)$$

Where I_e and I_s indicate the end time and start time of voice intensity, respectively.

(e) We define a step size of different timestamps for intensity features extraction to be Δi which is calculated as follows.

$$\Delta i = I_T/N_2. \quad (7.5)$$

(f) The different values of timestamps are determined using the following equation.

$$t_j = I_s + j*\Delta i;\ (0< j \leq N_2\text{-}1) \quad (7.6)$$

Where I_s is the starting position of intensity in the spectrograph shown in Fig. 7.2. (a) and (b).

Here, the IP vendor selected the number of timestamps, N_2 *is 15*. The initial (I_s) and final timestamps (I_e) on which the intensity values recorded are 0.09 and 0.73 (corresponding to the voice-001). The initial and final values of

intensity are determined in a similar fashion as previously determined for pitch (from Fig. 7.2 (a) and (b)). Hence, the time duration of voice intensity (I_T) is calculated to be (0.73-0.09) = 0.64, and the step size is computed to be = 0.0427 using (7.5).

In the proposed approach, values at 15 timestamps have been determined corresponding to two features, pitch and intensity. All these values will be unique as the proposed approach uses a stored voice template. For computing jitter and shimmer, the proposed approach considers a complete voice sample. We have only considered jitter local and shimmer local values. Both jitter and shimmer can be easily computed with the help of a voice report generated by a voice analyzer (*Praat*) application.

7.3.5. Feature extraction from voice template: pitch and intensity extraction

Extracting pitch and intensity features*:* In the previous subsection, we discussed how the different timestamps for extracting pitch and intensity values are determined. Further, in order to extract pitch and intensity features at the specified timestamps, the following steps are performed:

(a) The values of pitch and intensity are extracted by moving to a particular timestamp in voice spectrograph using voice analyzer (*Praat*) application.

(b) The pitch values read from the spectrograph at t_i time instant are denoted by $P(t_i)$. Table 7.1 shows the extracted pitch values corresponding to 15 different timestamps (selected by IP vendor).

(c) The intensity values read from the spectrograph at t_j time are denoted by $I(t_j)$. Table 7.2 shows the extracted intensity values corresponding to 15 different timestamps (selected by IP vendor).

Post obtaining the pitch and intensity feature values in decimal, they are converted to equivalent binary values. The timestamps where pitch and intensity are not defined are termed undefined (U) and the ASCII value of character 'U' is used as feature dimension (value). Tables 7.1 and 7.2 show the different values of pitch and intensity at different computed timestamps.

Table 7.1: Pitch values in hertz (Hz) Corresponding to Different Timestamps

Sr. No.	Timestamp (t_i)	Pitch $P(t_i)$	Magnitude	Corresponding binary equivalents
1.	0.15	$P(t_1)$	152	10011000
2.	0.19	$P(t_2)$	146	10010010
3.	0.23	$P(t_3)$	135	10000111
4.	0.27	$P(t_4)$	126	1111110
5.	0.31	$P(t_5)$	U	01010101
6.	0.35	$P(t_6)$	U	01010101
7.	0.39	$P(t_7)$	146	10010010
8.	0.43	$P(t_8)$	137	10001001
9.	0.47	$P(t_9)$	133	10000101
10.	0.51	$P(t_{10})$	134	10000110
11.	0.55	$P(t_{11})$	135	10000111
12.	0.59	$P(t_{12})$	133	10000101
13.	0.63	$P(t_{13})$	142	10001110
14.	0.67	$P(t_{14})$	162	10100010
15.	0.71	$P(t_{15})$	175	10101111

Table 7.2: Intensity values in HZ corresponding to Different Timestamps

Sr. No	Timestamp (t_i)	Intensity $I(t_i)$	Magnitude	Corresponding binary equivalents
1.	0.1327	$I(t_1)$	66	1000010
2.	0.1754	$I(t_2)$	81	1010001
3.	0.2181	$I(t_3)$	81	1010001
4.	0.2608	$I(t_4)$	75	1001011
5.	0.3035	$I(t_5)$	54	110110
6.	0.3462	$I(t_6)$	45	101101
7.	0.3889	$I(t_7)$	64	1000000
8.	0.4316	$I(t_8)$	70	1000110
9.	0.4743	$I(t_9)$	70	1000110
10.	0.517	$I(t_{10})$	70	1000110
11.	0.5597	$I(t_{11})$	70	1000110
12.	0.6024	$I(t_{12})$	69	1000101
13.	0.6451	$I(t_{13})$	67	1000011
14.	0.6878	$I(t_{14})$	65	1000001
15.	0.73	$I(t_{15})$	46	101110

Extracting jitter and shimmer features*:* The jitter (local) and shimmer (local) values are recorded from the voice report generated with the Praat voice analyzer application, corresponding to the input voice sample. Before computing jitter and shimmer (generating the voice report), we must first select the voice portion we want to consider in the spectrograph. In our experiment, we have considered the complete length of voice samples (as shown in Fig. 7.3). So, the obtained jitter (local) and shimmer (local) values (in %) and their corresponding binary equivalents for the voice-001 are:

Jitter (J_t)*= 2.782 = 10.11001000001100010 01*

Shimmer (S_h) *= 10.586 = 1010.10010110000000100001*.

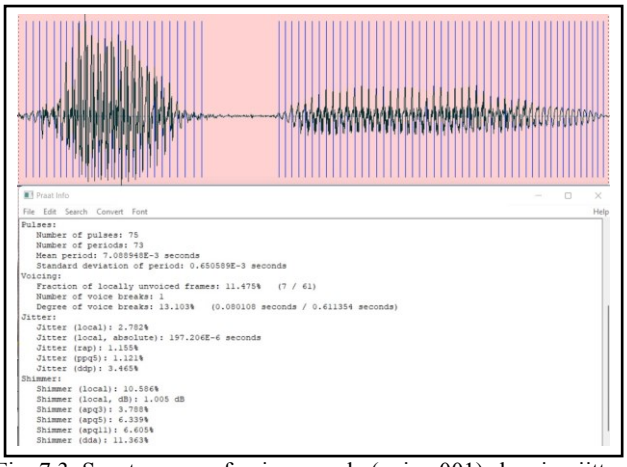


Fig. 7.3. Spectrogram of voice sample (voice-001) showing jitter (local), and shimmer (local) based on selected voice sample

7.3.6. Creating of watermark signature from extracted voice biometrics features and generation of its corresponding watermarking constraints

Once all the feature values and their equivalent binary are computed, they are encrypted using AES-256 and subsequently concatenated into a single encrypted template known as voice biometric- signature template based on the defined concatenation rule. This encrypted template is stored in a safe server for validation later. The concatenation rule is IP vendor specified. A designer's specified concatenation rule used in the proposed approach is given below.

$$S = \{\&_{i=1}^{N_1} P(t_i)\} \,\&\, \{\&_{j=1}^{N_2} I(t_j)\} \,\&\{J_t\} \,\&\{S_h\} \tag{7.7}$$

Where '&' is the concatenation operator. Based on the concatenation rule shown in (7.7), the generated voice biometric signature template is as follows.

S = [10011000100100101000011111111100101010101010110011001010001001100001011000011010000111100001011000111010100001010101111100001010100011010001100101111011010110110000001000110100011010001101000110100010110000011100000110111010110010000011000100110101001011000000100001] = 265 digits.

7.4. Demonstration: Embedding of Watermarking Constraints

Post obtaining the digital template of voice biometric signature, it is converted into equivalent hardware security constraints based on designer-selected encoding rules shown in Table 7.3. To perform the encoding, we traverse the

digital template to collect all different possible bit-variations, viz. 0→0, 0→1, 1→0 and 1→1. Further, each bit-variations has a specific encoding into a corresponding hardware security constraint, as shown in Table 7.3. For the signature template generated from voice-001, the total possible numbers of different bit-variations are as follows: (i) number of 0→0 bit-variations = 77, number of 0→1 bit-variations = 71, number of 1→0 bit-variations = 71, number of 1→1 bit-variations = 45. Hence the total number of voice signature constraints is 264.

After generating the voice biometric signature-based hardware security constraints, they are embedded in the respective IP core design using an HLS framework. A colored interval graph (CIG) of the respective DSP IP core is harnessed for embedding generated hardware security constraints. Nodes in the CIG represent storage variables corresponding to the target DSP IP core, and an edge between two nodes represents existing design constraints. The security constraints are implanted into the design in the form of extra edges in the CIG. The process of embedding addition edges (security constraints) between the same colored nodes in the CIG is achieved in two ways: *(a) local alteration*: the storage variable is allocated to some different colored register

Table 7.3: Mapping Rules (Encoding Mechanism) to Generate Hardware Security Constrains from Voice Biometric Signature

Encodings	Constraints
$E_{0\to0}$	Embed an edge between (even, even) node pair
$E_{0\to1}$	Embed an edge between (even, odd) node pair
$E_{1\to0}$	Embed an edge between (odd, odd) node pair
$E_{1\to1}$	Embed an edge between (prime, prime) node pair

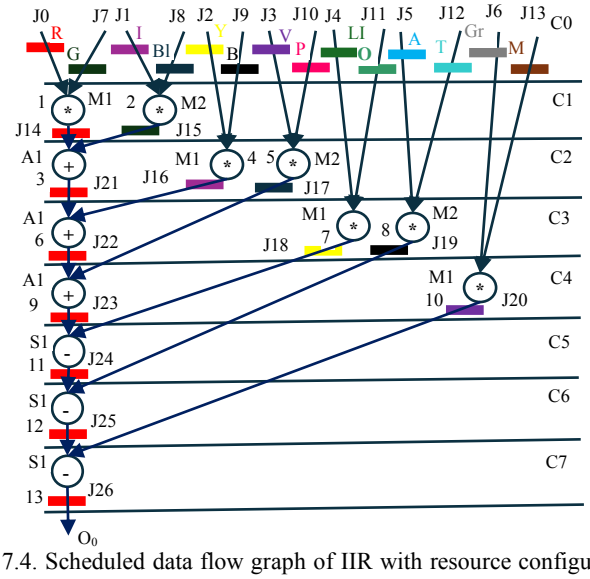


Fig. 7.4. Scheduled data flow graph of IIR with resource configuration of one adder (A1), one subtractor (S1) and two multipliers (M1 & M2)

that is not in use in that particular control step, *(b) new color register allocation*: if no any local alteration is possible while embedding of security constraints (extra edges) between same colored storage variables, then a new color (indicating a new register) is allocated. Similarly, all the voice biometric-based hardware security constraints are embedded in the design of the target DSP IP core.

Demonstration on IIR Filter Application: A DFG of IIR filter scheduled with one adder, one subtractor, two multipliers, and 27 storage variables (J0 to J26) is shown in Fig. 7.4. The 27 storage variables are executed using 14 registers highlighted using different colors. A corresponding register allocation table and CIG are shown in Table 7.4 and Fig. 7.5. (a), respectively. The hardware security constraints, corresponding to the voice signature template generated using (7.7), are embedded into the IIR filter design during the register allocation phase of the HLS process by exploiting the CIG framework. These security constraints are represented as four different sets of constraint edges ($C_{0\text{-}0}$, $C_{0\text{-}1}$, $C_{1\text{-}0}$, and $C_{1\text{-}1}$) corresponding to the four variations 0→0, 0→1, 1→0, and 1→1 in the voice signature template. The following are the sets of voice signature constraints generated for the IIR filter application.

$$C_{0-0} = \{(J_{2i}, J_{2j}),\ i,j \in W\ and\ (0 \leqslant i \leqslant 16), (1 \leqslant j \leqslant 18)\} \quad (7.8)$$

$$C_{0-1} = \{(J_{2i}, J_{2j+1}), i,j \in W\ and\ (0 \leqslant i \leqslant 16), (0 \leqslant j \leqslant 21)\} \quad (7.9)$$

$$C_{1-0} = \{(J_{2i+1}, J_{2j+1}), i,j \in W\ and\ (0 \leqslant i \leqslant 23), (1 \leqslant j \leqslant 25)\} \quad (7.10)$$

$$C_{1-1} = \{(J_i, J_j), (i \in prime\ and\ 2 \leqslant i \leqslant 19), (j \in prime\ and\ 3 \leqslant j \leqslant 23)\} \quad (7.11)$$

In the above equations (7.8) – (7.11), the limits for $'i'$ and $'j'$ depend on two factors: (i) the total number of bit-variations of a particular type and (ii) the limit of storage variables of the target application, which is J_{26} in case of IIR filter. For example, the total number of possible constraints corresponding to bit-variations (0→0) is 77. Therefore, generated security constraints corresponding to the IIR filter start from (J_0, J_2) and end with (J_{16}, J_{18}). Similarly, constraints for all remaining bit-variations are obtained.

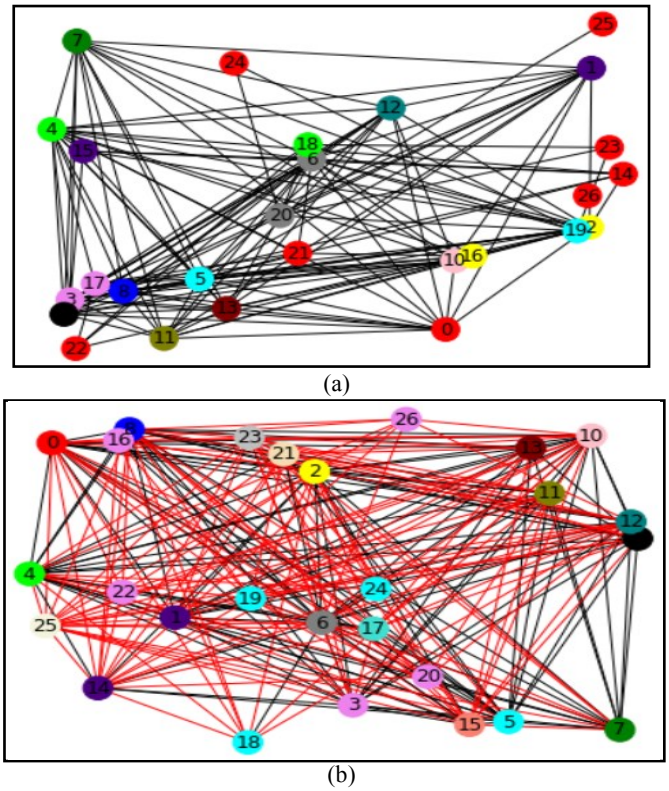


Fig. 7.5. (a). CIG of IIR filter before embedding voice biometric signature, and (b). CIG after embedding voice biometric signature, where the added voice signature constraints are highlighted using red edges

The voice signature constraints implantation process is performed by adding the aforementioned constraints edge sets to the CIG. Post implantation of constraints into the IIR filter design, the modified CIG and register allocation table are shown in Fig. 7.5 (b) and Table 7.5, respectively. The extra colors (registers) required to accommodate all the security constraints are highlighted in Table 7.5 in Black (Bold) text. Post embedding the voice signature constraints, a synthesized RTL datapath of IIR filter design is shown in Fig. 7.6. The portion of the datapath carrying the voice signature constraints is enclosed in a Red dotted rectangle.

7.5. Detection of Voice Biometric Signature

Figure 7.7 illustrates the detection process of voice biometric-based signature embedded into the hardware IP core. The detection process requires a pre-stored voice sample of the authentic IP vendor. First, the digital signature template is regenerated from the pre-stored voice sample using the proposed algorithm of signature generation. Then, the digital template is converted into corresponding hardware security constraints (representing the register allocation of different storage variables) using the predefined encoding rules.

Further, these security constraints are matched against the register allocation information extracted from the RTL datapath of the IP core (under test). If matching does not happen, then the design may be a counterfeit (i.e., absence of genuine IP vendor's voice signature). However, the complete matching implies that the design is authentic.

An adversary cannot evade the counterfeit detection process by embedding the authentic voice biometric information of the IP vendor into his/her fake design. This is because of the several security factors involved in the process of regeneration of the exact voice biometric based signature. Along with the

Table 7.4: Register allocation of IIR filter **before** embedding voice biometric based hardware security constraints

Table 7.5: Register allocation of IIR filter **after** embedding voice biometric based hardware security constraints

T	Red (R)	Indigo (I)	Yellow (Y)	Violet (V)	Lime (LI)	Aqua (A)	Gray (GY)	Green (G)	Blue (Bl)	Black (B)	Pink (P)	Olive (O)	Teal (T)	Maroon (M)	Lavender (L)	Khaki (K)	Cyan (C)	Wheat (W)	Silver (S)
0	J0	J1	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13						
1	J14	J15	J16	J17	J18	J19	J20												
2	J21																		
3	J22																		
4	J23																		
5	J24																		
6	J25																		

voice biometric of the IP vendor, an adversary also needs to decode the following security factors to forge the signature: (a) the IP vendor selected the number of timestamps for pitch determination, (b) the IP vendor selected the number of timestamps for intensity determination, (c) IP vendor selected range of voice sample for jitter and shimmer calculation (d) and concatenation sequence of the determined features (such as pitch, intensity, jitter and shimmer) to generate final voice biometric signature. Further, the voice sample used for generating hardware security constraints is stored in a tamper-proof secure database along with the IP designer's secret information, such as the number of timestamps used for features generation and the portion of voice sample used for jitter and shimmer feature estimation.

In case of IP ownership conflict, an adversary would fail to successfully claim the IP ownership as he/she is unable to match the embedded security constraints of the authentic voice biometric of the IP vendor (extracted from the final RTL datapath) with his/her signature. However, an authentic IP vendor would be successfully able to match his/her voice biometric signature with the embedded security constraints of the authentic voice biometric of IP vendor (extracted from the final RTL datapath). Therefore, IP ownership can be seamlessly awarded to the authentic IP vendor in case of ownership conflict.

7.6. Challenges and Limitations of Voice Biometrics

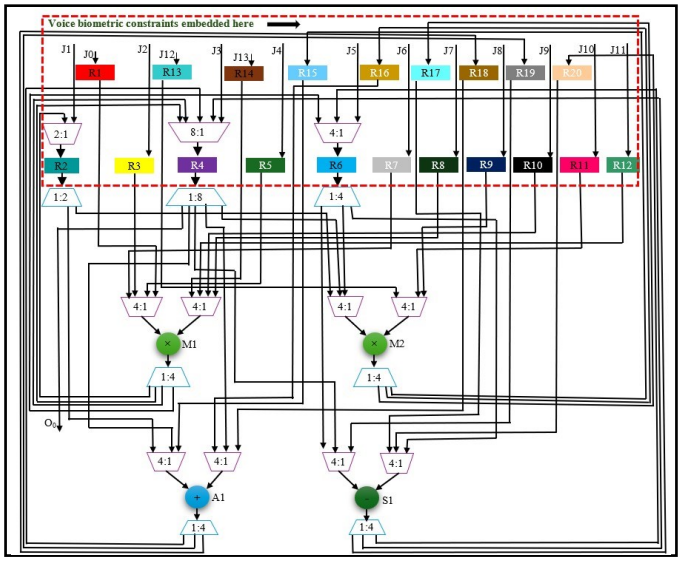


Fig. 7.6. RTL datapath of IIR filter IP core with embedded voice biometric based signature

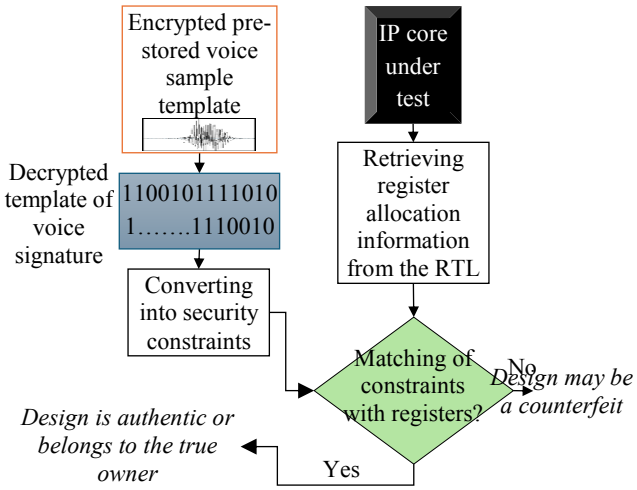


Fig. 7.7. Voice signature detection for IP authentication/ verification

Apart from the multiple advantages, voice-based biometrics pose some limitations too, such as it is not ideal to record voice sample at noisy environments. Recording of voice sample at noisy places and using poor-quality recording equipment are not favorable, as background noise present in voice sample will directly affect the feature extraction and signature generation process. How ever, standard low-pass IIR/FIR filter (LPF) can be used to eliminate the unnecessary background noise and maintain a proper signal to noise ratio. In addition, the speaker's voice is also affected by illness and aging during authentication/verification. However, in the proposed approach, recapturing of the IP vendor’s voice sample is not required as the voice sample of the original IP owner is safely pre-stored and used during verification. Further, the proposed approach requires secure storage of the voice template in encrypted format to safeguard against potential misuse.

7.7. Summary

This chapter presented a novel security methodology for securing hardware IP cores using voice biometric watermark signature. The discussed approach harnessed distinct voice biometric features, including jitter, shimmer, pitch, and intensity at various timestamps, to create a unique signature from the voice sample. This signature was then embedded into the target IP core design using the HLS framework. The proposed approach depicts robust security in terms of higher tamper tolerance and lower probability of coincidence (discussed in Chapter 9).

Chapter 8

HLS-Based Exploration of Low-Cost (Optimal) Functional Trojan-Resistant Hardware IP Designs

Various detective control mechanisms have been implemented to identify pirated/counterfeited IP cores in computing devices in the past. Techniques like hardware watermarking [31]-[36] and biometric-based methods [40], [41], [44] are commonly seen as effective in mitigating IP piracy risks. However, these methods do not address how to ensure that hardware IP core in computing application remain resistant to Trojan attack, particularly when backdoor functional Trojan are covertly inserted. An attacker could introduce a Trojan at any stage of the design process, whether at the vendor level by a malicious third-party IP provider/broker or during SoC integration, or at foundry level [125]-[129]. While some detection methods exist for identifying Trojans at the vendor or foundry stage, such as RTL simulation and side-channel analysis, none provide a comprehensive approach for designing Trojan-resistant hardware that can be applied to various computing applications [130], [131]. This chapter for the first time in literature demonstrates a complete HLS-base low-cost functional Trojan resistant design framework using distinct multi-vendor allocation policy. The proposed methodology leverages triple modular redundancy (TMR) to secure hardware SoC designs (IP cores), providing a more robust and reliable defense against functional hardware Trojan. Additionally, the approach incorporates a design space exploration framework to identify the optimal Trojan-resistant hardware architecture from a range of design possibilities.

The first section of the chapter discusses the motivation, threat model, problem formulation and advantages of designing optimal Trojan-resistant hardware IPs. The second section discuss the low-cost functional Trojan-resistant framework in detail. Following this, the third section illustrates the demonstration of the proposed approach. Next, advantages and limitations of low-cost Trojan resistant TMR framework in fifth section. Finally, the fifth section provides the chapter's conclusion.

8.1. Problem Formulation

8.1.1. Threat model, Motivation and advantages of designing optimal Trojan-resistant hardware IPs

Hardware Trojans pose a significant threat to the integrity of application-specific computing systems that rely on a variety of hardware IP cores for their functionality. These IP cores perform crucial tasks like image compression, audio equalization, and digital broadcasting in devices used across industries. For example, IP cores such as the discrete cosine transform (DCT) and haar wavelet transform (HWT) are vital for compressing audio, video, and image files, while finite and infinite impulse response (FIR/IIR) filters are essential for sound processing in audio systems. Similarly, JPEG codecs manage image and video compression in digital cameras, and fast fourier transform (FFT) is used in digital video broadcasting. Given the critical role these components, any vulnerability introduced through a hardware Trojan could have severe implications for end-users. Hardware Trojans are malicious modifications to a circuit that are covertly inserted at any stage of the design process, potentially by a rogue 3PIP vendor. These Trojans remain dormant until triggered, making them difficult to detect during standard testing procedures. Once activated, they can cause erroneous output or complete failure of the system, posing significant risks to safety, reliability, and performance. In critical infrastructure or mission-sensitive applications, such failures could be catastrophic. For instance, in image classification systems or biometric authentication processes, a Trojan could alter the output of convolution filters used in convolutional neural networks (CNNs), leading to incorrect identification or authentication results. Similarly, in medical imaging applications, a Trojan could affect the accuracy of image compression, resulting in incorrect diagnoses and potentially fatal consequences for patients.

Further, the insertion of hardware Trojans in machine learning systems also raises serious concerns. In scenarios where machine learning coprocessors are used for decision-making, a Trojan could manipulate the prediction outcomes. For example, an attacker could inject a Trojan that alters the results of a medical diagnosis model, leading to misdiagnosis and improper treatment. This not only compromises the reliability of the system but also opens avenues for adversaries to gain financial or competitive advantages. Additionally, the

Trojan could tamper with digital filter cores like the Sobel edge detector and Gaussian filters, which are widely used in image processing for both consumer and industrial applications.

To address this challenge, ensuring that IP cores are resistant to functional Trojan attack is essential. The proposed methodology aims to provide a robust defense against these threats by implementing TMR along with a distinct vendor allocation policy. This approach isolates any infected unit, preventing the Trojan from affecting the overall system. Furthermore, by distributing the design workload across multiple vendors, the probability of multiple IP cores being compromised by the same Trojan payload is significantly reduced. This isolation not only improves the security of the design but also ensures reliable functionality across various applications, from consumer electronics to critical medical and industrial systems. By safeguarding against hardware Trojans, the proposed system enhances the reliability of custom computing devices, protecting both the integrity of the design and the safety of end consumers. It also helps ensure that even if one vendor's IP core is compromised, the overall system remains functional, minimizing the risk of widespread failure or data corruption.

8.1.2. Problem formulation

The problem solved in this chapter can be formulated as follows: Designing Trojan resistant hardware IP core design while minimizing {hybrid cost (A_{TMR}, T_{TMR})} based on explored optimal resource configuration $\{S_i\}$ using design space exploration. The associated variables of the methodology are explained in nomenclature table. This chapter addresses the Trojan resistance of DSP hardware IP cores against Trojan that are capable of inducing functional error in the computed output. Trojans that are responsible for the denial of service and leakage of secret information are not targeted in the proposed approach. However, the proposed approach enables the defense against functional Trojans that may be inserted at the IP vendor/designer's level (not easily detectable during test vector analysis and normal run) and induces erroneous functional behavior. Further, the micro-IPs or modules present in the library of an HLS tool used for hardware IP core design are also susceptible to Trojan infection, including third-party IP vendors or untrustworthy entities.

Consequently, integrating such malicious hardware IPs into custom computing systems may induce abnormal functioning of the devices, causing safety hazards to the end consumer.

The proposed approach for the Trojan resistant design of the DSP hardware IP core is evaluated based on the following crucial parameters: design area, delay, design cost, and security (in terms of the number of vulnerabilities tackled):

(1) *Proposed Area Metric*: Total area covered (A_{TMR}) by the Trojan-resistant TMR design is given by (8.1).

$$A_{TMR} = \sum_{j=1}^{3} \sum_{i=1}^{3} (A({C_i}^{V_j}) * ({C_i}^{V_j})) \tag{8.1}$$

Where, $A({C_i}^{V_j})$ indicates the area of a resource type C_i corresponding to the vendor V_j and ${C_i}^{V_j}$ indicates the number of instances utilized for a resource type C_i from the vendor V_j. Further, the details of the area and delay of functional resources (adder, multiplier) corresponding to vendors are adopted from the related approach [45].

(2) *Proposed Delay Metric*: Design latency (T_{TMR}) metric involves the delay due to the number of control steps required while scheduling the design using functional resources and

delay due to respective functional components corresponding to each vendor. The delay metric can be represented as follows:

$$T_{TMR} = \sum_{v=1}^{3} ((n_m * d_m) + (n_a * d_a) \tag{8.2}$$

(3) *Design Cost Function (fitness function)*: The fitness function includes normalized area and execution time corresponding to the architectural design of Trojan-resistant TMR schedule and can be formulated as follows:

$$Design\ Cost\ (Fitness) = e_1 * \left(\frac{(A_{TMR} - A_{Con})}{A_{max}}\right) + e_2 * \left(\frac{(T_{tmr} - T_{Con})}{T_{max}}\right) \tag{8.3}$$

Where e_1 and e_2 are designer-defined weighing factors. Further, A_{MAX} and T_{MAX} represents maximum design area (computed using allocating maximum functional resources available) and delay (computed using allocating minimum functional resources) while A_{TMR} and T_{TMR} represents the computed

area and delay of the proposed Trojan resistant TMR hardware design. Further, A_{Con} and T_{Con} are IP vendor-specified area and latency constraints.

8.2. Low-Cost Functional Trojan-Resistant Framework

8.2.1. Underlying assumptions

1. The voter in the proposed approach is fault tolerant (adopted from [124]), which means it produces functionally correct output always.
2. We have considered an error detection block (EDB), which is a multi-stage setup (adopted from [124], [132]) designed to protect the Trojan-resistant design from faulty comparators.
3. The fault-tolerant voter and error detection block used in the proposed approach is considered to be Trojan-free (trustworthy). This is because these hardware modules are considered to be designed in-house (by a system integrator). In the proposed approach, the system integrator is considered to be trustworthy.
4. The information corresponding to multiple vendors is confidential and only known to the system integrator. The vendors are completely unaware of the information about their counterparts. As vendors in the proposed approach are unaware of their counterparts. Therefore, the chances of collusion between distinct unknown 3PIP vendors to achieve the same Trojan payload are very low. Henceforth, the proposed approach always, at minimum always, ensures Trojan detection [45].

8.2.2. Low-cost Trojan-resistant TMR design framework

The proposed solution for DSP hardware design with Trojan defense ability exploits an optimal design architecture. The PSO-DSE process is employed to explore an optimal resource configuration for Trojan resistant design. The overview/thematic representation of the proposed approach is shown in Fig. 8.1. The primary inputs consist of CDFG of the selected hardware application, PSO initial parameters, module library, designer's specified design constraints (A_{con}, T_{con}), and resources (such as adder, multiplier, etc.) from three distinct IP vendors. At first, the particle/swarm positions are initialized, and the

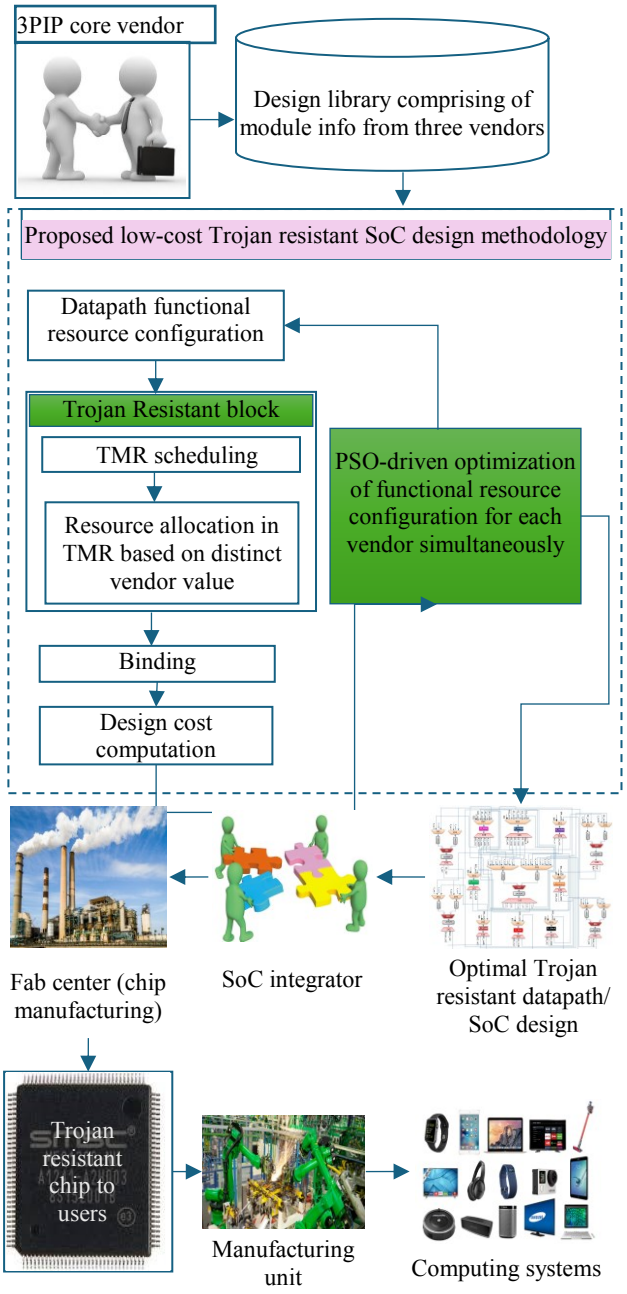


Fig. 8.1. Overview of proposed optimal Trojan defense IP core/SoC design generation process for DSP applications

corresponding TMR schedule is generated. Next, the design cost, latency and its corresponding cost are computed for each particle. Subsequently, the local best and global best particle position are updated. At last, mutation is performed on particle position to make the optimal solution search more diversified. For each resource configuration explored (using PSO-DSE) corresponding to Trojan-resistant scheduled and allocated design during HLS, the respective datapath is generated. Finally, the optimal Trojan resistant datapath is obtained at the end of DSE process. The details are discussed in the next subsections.

This chapter explicitly presents a case study on FIR IP core used in several computing applications such as convolution process in image/video processing, signal attenuation, signal filtering in audio/image applications, etc. Further, it can be converted into an optimal design with Trojan defense capability. Even if a malicious backdoor logic exists in the design, it can still produce the correct output functionally using the concept of TMR, distinct vendor allocation policy, and voter. In order to explore optimal Trojan resistant architecture corresponding to DSP hardware designs used in custom computing systems, the proposed approach accepts the following inputs: DFG of the DSP application, multivendor library, PSO-DSE parameters such as inertia weight (ω), acceleration coefficient ($b1$ and $b2$), terminating criteria (T), population size (n) with initial particle position/ functional resource configuration. The output of the proposed approach yields an optimal architectural solution (global best solution) for Trojan resistant DSP register transfer level (RTL) datapath soft IP.

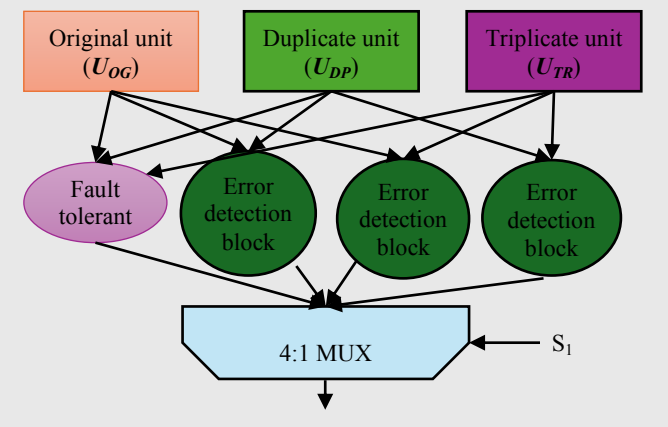


Fig. 8.2. Illustration of the trojan resistance capability of the proposed approach with the help of voter and 4:1 multiplexer

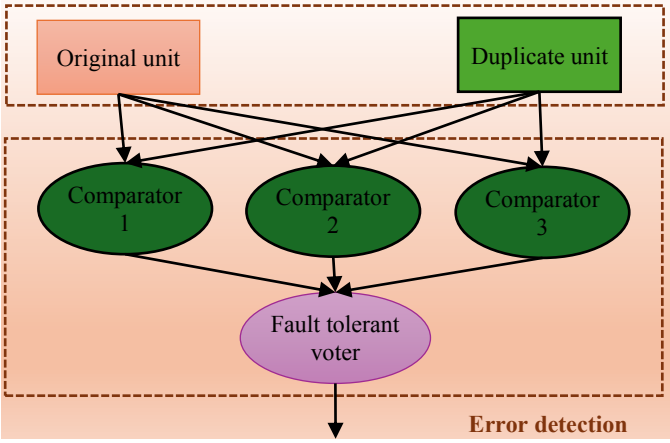


Fig. 8.3. Error detection block adopted from [124]

The proposed approach presents the TMR design framework to provide Trojan resistance against functional Trojans. The proposed approach shows greater defense in terms of providing Trojan detection and isolation, both based on the proposed setup (shown in Fig. 8.2). Fig. 8.2 illustrates the Trojan resistance capability of the proposed approach. The proposed solution provides complete Trojan resistance with the help of a TMR-based design and a fault-tolerant voter (V), adopted from [132]. The fault-tolerant voter will generate the correct output based on the majority. The output of the 4:1 multiplexer (MUX) ensures Trojan detection when select lines (S_1 S_0) chosen are 01,10,11 and provides Trojan resistance when select lines chosen are 00. In order to ensure that the proposed approach always at least provides Trojan detection, the concept of a multi-stage setup using error detection blocks (EDB) has been integrated into the proposed setup. Fig. 8.3 depicts an architecture of EDB used in the proposed approach using multiple comparators and fault tolerant voter (adopted from [124] and [132]). This multi-stage setup (or EDB) is used to handle cases of faulty comparators. Assuming fault in the comparator: in such a case, the faulty comparator out of the used three comparators inside a multi-stage setup of EDB will produce a complementary output of the remaining two. Therefore, the fault-tolerant voter [132] will produce the correct majority output generated by the remaining two correct comparators. The area overhead of used EDB (adopted from [124]) is negligible [124].

In the TMR design logic, besides the original unit, two other units corresponding to the TMR design are employed by duplicating the operations of the DFG (DSP application). The combined DFG logic with original, duplicate, and triplicate units, called TMR logic, is then scheduled using the LIST scheduling algorithm based on the hardware resources explored using PSO-DSE (discussed in the next section). LIST scheduling is a scheduler that works by aiming to schedule the maximum number of operations in a single control step, subject to resource constraints and data dependency. Therefore, it minimizes the number of control steps required to schedule the DSP application compared to traditional ones. Next, in the proposed scheduled TMR design, each of the three distinct vendors is then allocated to a dedicated unit. In the proposed approach, a distinct multivendor allocation policy has

been presented, where the resources from a particular vendor are allocated to a single unit of the TMR design. This is effective for providing Trojan resistance compared to assigning multivendor resources simultaneously within a single unit (either original, duplicate or triplicate unit) of TMR logic. It enhances the likelihood of the design being infected, thereby carrying malicious logic in each TMR unit. Therefore, multivendor resource allocation within the single unit of TMR logic fails to provide Trojan resistance against hidden functional Trojans. On the contrary, a distinct multivendor resource allocation process enables easy isolation of any Trojan-infected design. In other words, if either of the three vendors carries malicious backdoor logic in their resources (micro-IPs), then due to distinct vendor allocation to each unit of TMR logic, it would enable the remaining two units to remain Trojan-free. Thereby allowing the non-erroneous output to pass through the voter. For example, if vendor VD_1 carries a backdoor Trojan and is allocated to only the original TMR unit, the remaining two vendors, VD_2 and VD_3, are allocated to duplicate and triplicate units, respectively. Therefore, due to this distinct multivendor allocation, the voter would pass the correct majority output from duplicate and triplicate units, thus providing Trojan resistance against functional Trojans. However, it is possible that vendor assignment can be changed internally between original, duplicate, and triplicate units as long as distinctness is maintained across units. If vendor VD_1 is allocated to any one of the three units, then vendor VD_2 and VD_3 are assigned distinctly with the other two units. Further, in case if any unit is Trojan-infected, then due to a distinct vendor allocation policy, it would enable the remaining two units to be Trojan-free.

The mathematical representation of the proposed vendor allocation policy for enabling Trojan-resistance through TMR is discussed below.

$O_i \in U_{OG}, O_{i'} \in U_{DP}, O_{i''} \in U_{TR}$

Where 'i' = 1→ t, 'O_i' denotes i^{th} operations in single instance of CDFG, 't' denotes total number of operation present in single instance of CDFG and $\{U_{OG}, U_{DP}, U_{TR}\}$ are explained in nomenclature table.

$\{ U_{OG}, U_{DP}, U_{TR}\} \in SDFG^{TMR}$

$VD_i \leftarrow U_{OG}$, $VD_j \leftarrow U_{DP}$, $VD_k \leftarrow U_{TR}$, where 'VD_i' is the 'i^{th}' IP vendor and $VD_i \neq VD_j \neq VD_k$

Then, based on the majority solution (at least two of the three units producing the same output value), the fault-tolerant voter [132] will pass the functionally correct output, thereby providing Trojan resistance against functional Trojan that a rogue element could have implanted during the in-house designing process. Finally, based on the obtained scheduled and allocated Trojan-resistant TMR design, the design area using (8.1) and execution time using (8.2) are computed. Subsequently, the fitness cost for the corresponding Trojan resistant hardware design is evaluated by substituting the area and delay in (8.3) to iterate the PSO-DSE process till the stopping criterion is achieved. The overview of the PSO-DSE is discussed in the upcoming sub-section.

8.2.3. Exploration for low-cost Trojan-resistant TMR scheduling using PSO-DSE

As shown in Fig. 8.4, the PSO-driven design space exploration is responsible for generating an optimal Trojan resistant design architecture. At first, the initial position of each particle (representing initial functional resource configuration) and velocity (A_{ix}) are initialized, where the first particle's position S_1 is initialized with minimal resources such as S_1= (min (Z_1), min (Z_2) …. min (Z_n)) and second particle S_2 is initialized with maximum resources such as S_2= (max (Z_1), max (Z_2) ….max (Z_n)); while the third particle S_3 is initialized by the average of minimum and maximum resource value, where 'Z' represents the resource type (adder or multiplier). However, the rest of the particle positions (S_4 …S_n) are initialized using the following equation:

$$S_{ix} = (f + g)/2 \pm h \tag{8.4}$$

Where '*f*' is the minimum resource value, '*g*' is the maximum resource value, and '*h*' is any random number between '*f*' and '*g*'. 'S_{ix}' represents the resource configuration of i^{th} particle x^{th} dimension. Here, dimensions imply the resource types: adder and multiplier (X=2). At first, the fitness cost value (using the cost function) corresponding to each particle ('i^{th}') (as explained earlier in the above section) is computed. Subsequently, in its first iteration, the global best

particle position is determined based on each particle's initial cost value. The particle with minimal cost function value among all particles is selected as the global best resource configuration by the PSO-DSE. Subsequently, the new position of the particles is computed by adding the computed velocity/ displacement to the previous particle position [78]. Furthermore, if the computed velocity causes excessive exploration drift, then the velocity clamping is performed to keep the particle within the design space. However, if the new particle position outreaches the boundary space, adaptive end-terminal perturbation is performed to limit the particle within its valid design space [78]. This process is executed for each remaining particle selected by

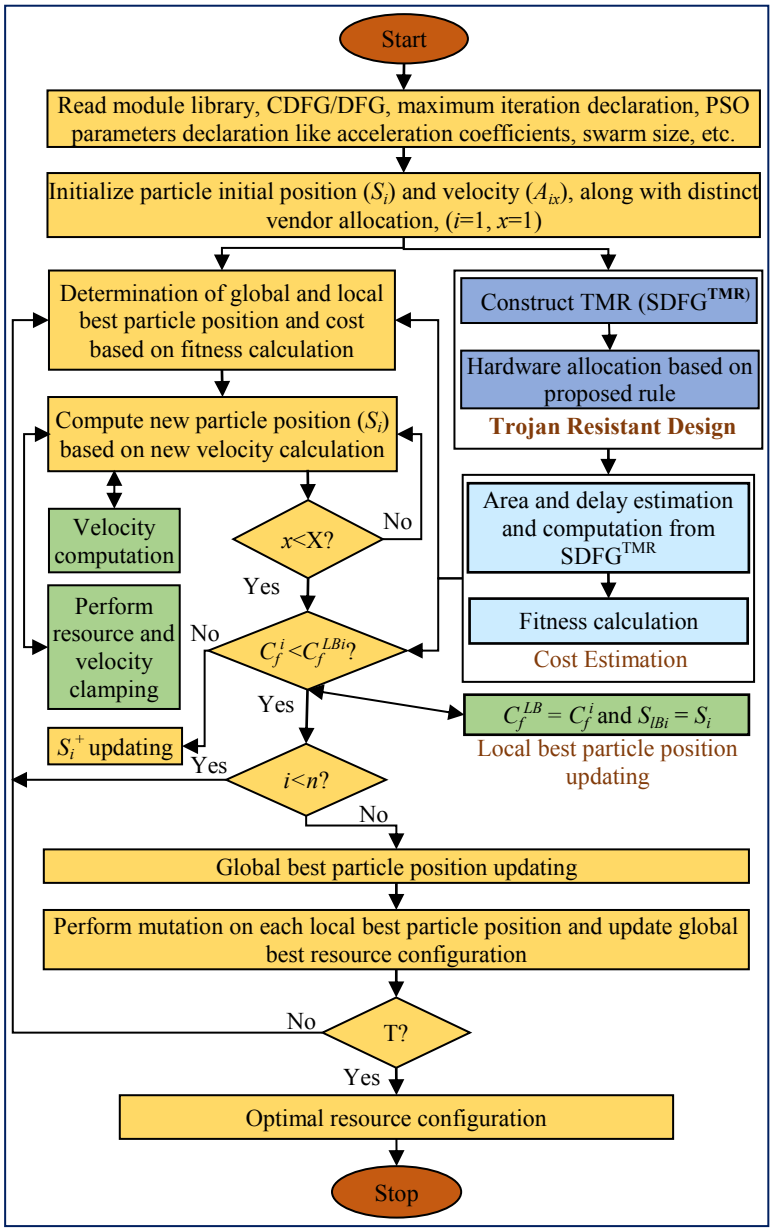


Fig. 8.4. Flow-chart of proposed optimal TMR based Trojan resistant methodology

the IP designer in the design space. Subsequently, the cost is computed for each of the remaining particles in the population (*i.e.,* for $i<n$, where 'n' is the total number of particles in the population), and if the currently computed cost is less than the cost of the particle obtained in the previous iteration, then both local best particle position, global best particle position and also the respective cost values are updated with currently computed cost. Moreover, a particle with a lesser fitness cost value is declared as the fittest among the remaining particles. In the next phase, the mutation is performed on each particle position to diversify the solution, better explore the design space and avoid getting stuck in the local minima [78]. Subsequently, each mutated particle's fitness cost is computed, and local and global best resource configurations are also updated. This process continues until the terminating condition is met. The exploration process gets terminated if the algorithm has already converged to global minima and results in no further updation/ exploration for the next ten consecutive executions (T_1= 10) or if the number of iteration counts is exhausted (T_2 ≥50). Thereby, an optimal design architecture solution for Trojan resistant TMR design is obtained using PSO-DSE.

8.3. Process Demonstration: Motivational Example

The demonstration of enabling Trojan-resistant capability of hardware IP cores used in computing systems is shown using finite impulse response filter (FIR) as follows: (i) high-level representation of FIR filter application is transformed to corresponding data flow graph (DFG). This is performed by parsing the transfer function of the FIR filter into matrix multiplication format comprising of input data samples, FIR filter coefficients and output samples. Further, the matrix multiplication format is generically represented as a mathematical function for computing the n^{th}-output data sample. Finally, this mathematical function is represented as a connected graph, where each node represents the mathematical operations and the edges represent the data dependency, (ii) subsequently, two other units (duplicate and triplicate) are created corresponding to the same application (main/ original) in regard of TMR design (iii) all the three units corresponding to TMR design are scheduled using optimal resource configuration (adders and multipliers) obtained using PSO-driven design exploration. For a single unit, explored

resource configurations are three multipliers and one adder. The first unit (original unit) is scheduled by considering the inputs from vendor VD_1. Subsequently, the second (duplicate unit) and third (triplicate unit) are scheduled using resources from vendor VD_2 and vendor VD_3, respectively. The proposed approach considers the simultaneous execution of the original unit along with two duplicate units. Further, the TMR design is scheduled using the LIST scheduling algorithm. In order to schedule the TMR design of the FIR filter, a total of nineteen control steps are required to generate the functional output value. The SDFG of the TMR design corresponding to the FIR filter using the proposed approach is shown in Fig. 8.5.

8.4. Advantages and Limitations of Low-Cost Trojan Resistant TMR Framework

Advantages:

a) The proposed approach generates a complete Trojan resistant DSP hardware design capable of providing 100 percent resistance against functional Trojans (that affects the computational outputs). The Trojan-resistant design automatically detects and isolates the malicious logic (Trojans) present in the design.

b) Additionally, the proposed approach discusses a framework of integrating the Trojan resistant design with the PSO-DSE (considering area and latency metric tradeoff) to generate low-cost optimized

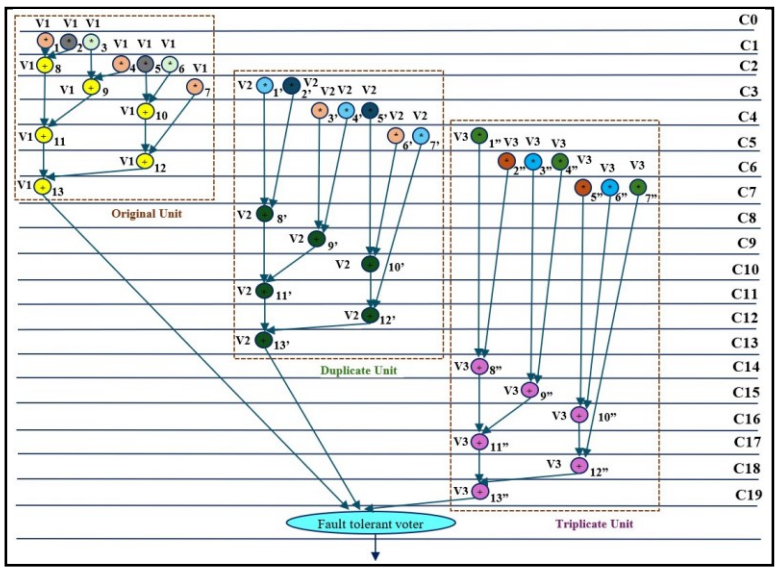


Fig. 8.5. Scheduled data flow graph of FIR filter (TMR) with 9(*), 3(+)

architecture corresponding to Trojan resistant hardware design. PSO prunes the potential architecture solutions present in the design search space to obtain a low-cost optimized solution.

Limitations:

a) The integration of PSO-DSE with the proposed Trojan resistance design is mandatory to handle the overhead caused by the replication of functional units (FUs) in the proposed approach.

b) The values of various control parameters used in PSO-DSE, such as inertia weight, random numbers, weights, social and cognitive factors, should be chosen appropriately to facilitate proper convergence of PSO-DSE to optimal solution in an acceptable time.

c) The proposed approach is based on assumptions of using fault-tolerant voter [132], error detection block [124], [132] based on multi-stage comparators to guard against faulty comparisons, and trustworthy (Trojan-free) fault-tolerant voters and comparators designed in-house.

8.5. Summary

This chapter discusses a low-cost solution for making hardware IP designs resistant to hardware Trojans for application-specific computing systems. It combines a PSO-based design space exploration technique with a TMR-based security strategy to create an optimal, low-cost SoC design that provides functional Trojan resistance in hardware applications. This approach utilizes a unique vendor allocation policy for the original, duplicate, and triplicate units within the TMR-based SoC design. Even if one of the TMR units is compromised by a functional Trojan, the system still produces correct outputs. As discussed in the Chapter 9, this Trojan-resistant method adds only a minimal design cost overhead.

Chapter 9

Results and Analysis

This chapter presents the experimental results and analysis of the proposed hardware security techniques aimed at ensuring the security and protection of data-intensive hardware IP cores. The results were obtained using various data-intensive DSP and multimedia benchmarks [83]-[86].

9.1. Experimental Results: Exploration of Low-Cost Hardware IPs during HLS using Multiphase Encryption and Crypto-Chain Signature

The experimental results of the proposed low-cost multi-phase encryption and low-cost crypto-chain signature based security methodologies (discussed in Chapter 3) are analyzed and discussed in this section. The proposed low-coat multi-phase encryption based security approach is validated on a system with a 2.30 GHz processor and 4GB RAM. The parameters for PSO-based architecture exploration are initialized as follows: acceleration coefficients (f_1 and f_2) =2, k_1 and k_2 = 1, N =3, and ω = linearly decreasing between 0.9 to 0.1 [79]. Further, the experimental analysis of the proposed FFA based crypto-chain security approach has been performed on a system with a 2.30 GHz workstation and 4GB main memory. The parametric values used for the proposed approach: firefly population size (Y) =3,5, and 7, $q1$=$q2$=0.5, β_0 = 1, γ = linearly decreasing from *0.5* to *0.1,* α_x and α_y = linearly decreasing value from the maximum value of the first dimension and maximum value of the second dimension, and *rand* = 1.5 [80]. The evaluation of area and latency corresponding to JPEG-CODEC is performed using a 15nm scale using NanGate library [86].

9.1.1. Results in terms of security, design cost, and implementation complexity analysis

(i). Security Analysis: The proposed security methodologies security is analyzed using two established security metrics (a) probability of coincidence (C_l) and (b) tamper tolerance (T_o) [25], [31], [32], [33]. The probability or likelihood of detecting identical covert security information in a baseline

design is called the probability of coincidence (PC/C*i)*. The PC/C_i is formulated as (9.1):

$$C_i = (1 - 1/b)^p \tag{9.1}$$

Where '*b*' denotes the number of registers (color) present in the SDFG of target application before embedding secret security constraints, and '*p*' denotes total embedded secret security/watermarking constraints. The robustness of the proposed security approach is inversely proportional to the value of C_i (*i.e.,* a lower value of C_i indicates a stronger security approach in terms of obtaining stronger digital evidence). The presence of a unique watermark signature inside the design of hardware application helps in the definitive and robust detection of pirated IP cores from genuine ones. Further, tamper tolerance (TT/T_o) is formulated as (9.2):

$$T_o = v^p \tag{9.2}$$

Where '*p*' denotes total embedded secret security data, and '*v*' denotes the quantity of distinctive encoding variables employed in the security approach. The greater the magnitude of T_o, the bigger the signature space (*i.e.,* stronger security). The generation of different signature combinations increases with an increase in the T_0, making it challenging for attackers to decode the precise signature blend to extract the covert constraints. Consequently, the security technique's sturdiness against tampering attacks increases with a higher value of T_o. The primary purpose of the adversary is to regenerate the exact signature (using different mechanisms such as brute force attack, etc.) so that they can easily evade the IP counterfeit detection process. Therefore, a higher value of T_o hinders the adversary from performing a tampering attack on the secured IP core design.

Proposed low-cost multiphase encryption based security methodology: Fig. 9.4 shows the comparison of PC between the proposed approach and [37]. Further, Fig. 9.5 and 9.6 report a similar comparison of PC between the proposed approach and related works [31] and [32], respectively. The proposed approach reports a lower PC than [31], [32], and [37], which means the proposed approach provides stronger digital evidence than [31], [32], and [37] due to generation of a higher number of hardware security constraints.

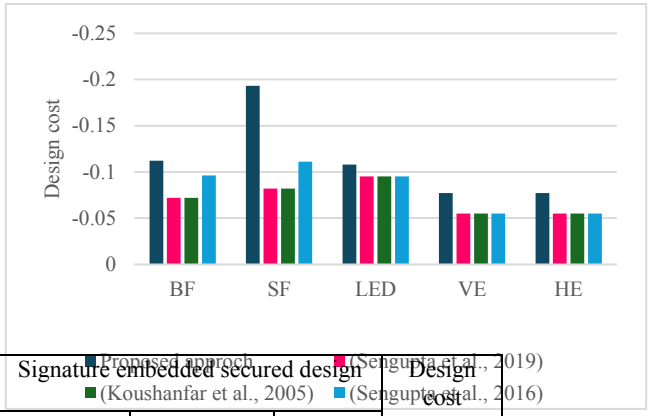


Fig. 9.1. Design cost comparison between proposed approach and (Sengupta et al., 2019) [37], (Koushanfar et al., 2005) [31], (Sengupta et al., 2016) [32]

Benchmarks	Explored resource configuration	Baseline unsecured design (before signature embedding)			Signature embedded secured design			Design overhead (%)
		Design area (um^2)	Design latency (ps)	Design cost	Design area (um^2)	Design latency (ps)	Design cost	
Blur filter (BF)	3(+), 4(*)	147.849	927.399	-0.112	147.849	927.399	-0.112	0
arpening filter (SF)	4(+), 2(*)	243.79	794.91	-0.193	243.79	794.91	-0.193	0
e edge detection filter (LED)	2(+), 2(*)	199.75	728.671	-0.108	199.75	728.671	-0.108	0
al embossment (VE)	1(+), 1(*)	99.09	596.185	-0.077	99.09	596.185	-0.077	0
ntal embossment (HE)	1(+), 1(*)	99.09	596.185	-0.077	99.09	596.185	-0.077	0

Table 9.1. Design cost, and resource configuration of proposed security methodology before and after embedding signature

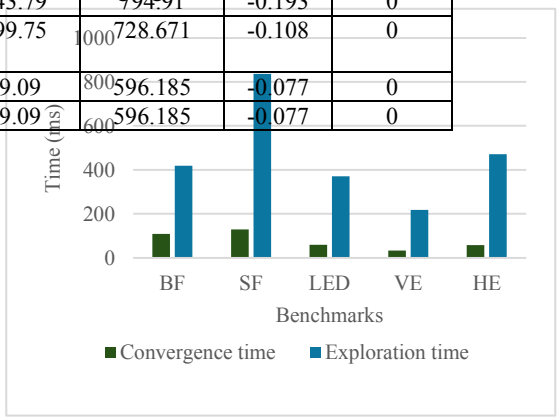


Fig. 9.2. Convergence time and exploration time for the proposed methodology

Due to embedding a higher number of generated security constraints, it becomes challenging for an adversary to detect the same security constraints in an unsecured design. Further, embedding a higher number of security constraints increases the presence of IP vendor-specific digital evidence in the design.

Next, Fig. 9.7 and 9.8 show a comparison of tamper tolerance between the proposed approach and the related works [31] and [32]. The proposed security methodology shows a significantly higher tamper tolerance ability over [31] and [32] due to generation and embedding of a higher number of hardware

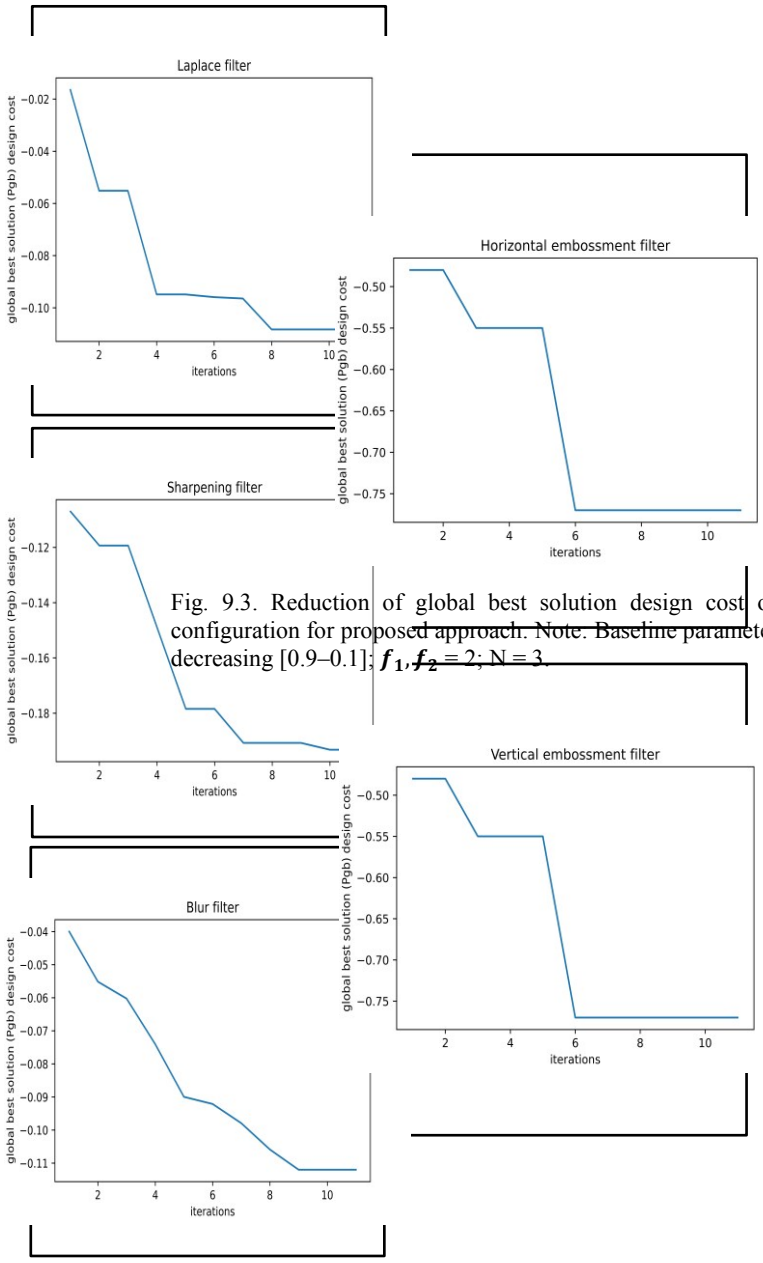


Fig. 9.3. Reduction of global best solution design cost over the iteration counts during dete
configuration for proposed approach. Note. Baseline parameters for PSO-based architecture explora
decreasing [0.9–0.1]; $f_1, f_2 = 2$; N = 3.

security constraints. Further, the comparison of TT is not reported here between the proposed and the related work [37] as security methodology [37] does not use an encoding mechanism (where encoding variables are not required) to generate the hardware security constraints. Moreover, the proposed approach is highly robust in terms of security because of the following factors: (a) generation of RAT based on low-cost resource architecture obtained through a heuristic, (b) IP vendor selected key for initial state matrix generation, (c) IP vendor selected key for row-diffusion (d) IP vendor selected key for byte concatenation, (e) IP vendor selected encoding rule (f) IP vendor selected keys for TRIFID cipher computation.

Proposed low-cost crypto-chain based security methodology: Table 9.5 illustrates a comparative study of C_i between the proposed and [44], [37], [32], [43], [31], and [107]. The proposed FFA based crypto-chain security approach surpasses all of the above-mentioned approaches with a lower value of C_i. This is because the proposed approach facilitates determining and implanting higher covert constraints (*i.e.,* security constraints that provide stronger digital evidence) than the related approaches. The production and implantation of a greater count of security information into the design make the incidence of the same security information in an unsecured design highly improbable for an

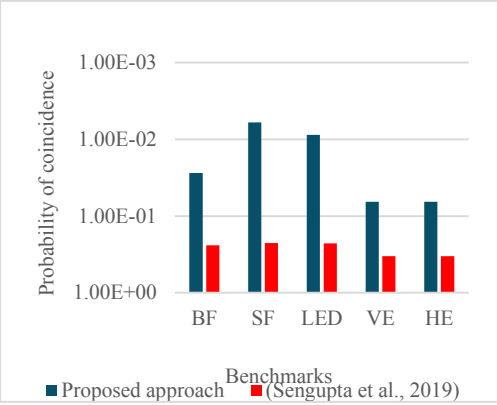


Fig. 9.4. Comparison of PC between the proposed and (Sengupta et al., 2019) [37]

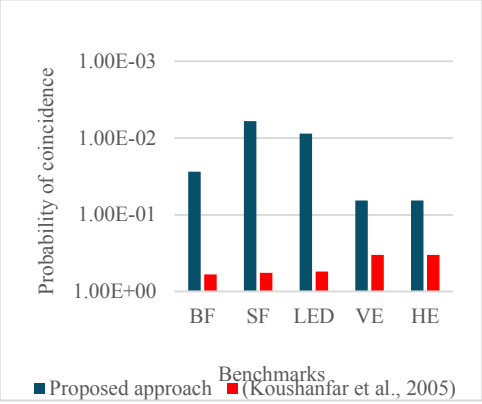


Fig. 9.5. Comparison of PC between the proposed and (Koushanfar et al., 2005) [31]

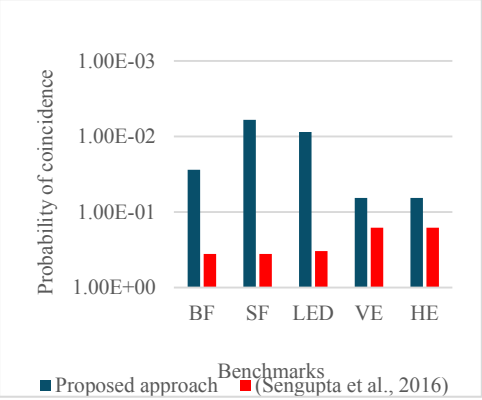


Fig. 9.6. Comparison of PC between the proposed and (Sengupta et al., 2016) [32]

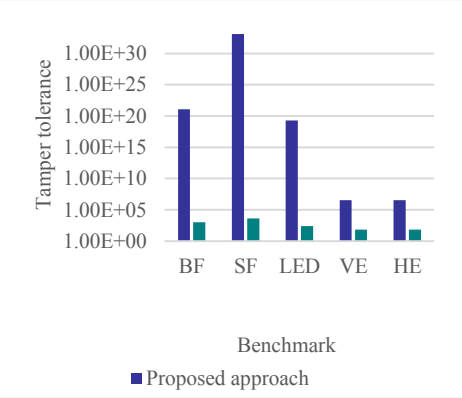


Fig. 9.7. Comparison of TT between the proposed and (Koushanfar et al., 2005) [31]

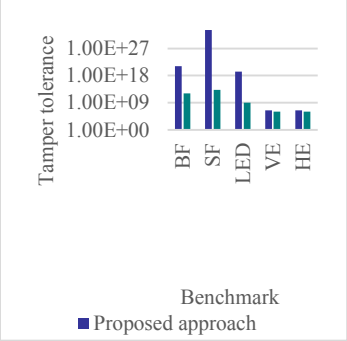


Fig. 9.8. Comparison of TT between the proposed and (Sengupta et al., 2016) [32]

Table. 9.2: Result of the proposed approach in terms of optimality

Benchmarks	GEN	Spacing (SPA)	Spread (SPD)	Weighted metric (WEM)
BF	0.00	0.111	0.372	0.186
SF	0.00	0.745	0.611	0.306
VE	0.00	0.00	0.00	0.00
HE	0.00	0.00	0.00	0.00
LED	0.00	0.5	0.167	0.084

adversary.

Next, Table 9.5 also shows a comparative investigation of T_o between the proposed and [44], [32], [43], [31], and [107]. The proposed FFA based crypto-chain security approach surpasses all of the above-mentioned approaches with a higher value of T_o. Extracting the exact security constraints from the derived crypto-chain signature combination amongst the innumerable combinations in the signature space becomes impossible when the tamper tolerance value is large, which is the case for the proposed approach. On the contrary, the related approaches achieve lesser tamper tolerance magnitude, as evident from Table 9.5. This indicates that the proposed approach is more secure and robust than all the related approaches. Further, T_o corresponding to [37] is not reported as this security mechanism does not employ signature encoding to generate security constraints. Additionally, Table 9.6 reports variations in the values of C_i and T_o *w.r.t.* embedded crypto-chain signature bits into the design for JPEG-CODEC. The value of C_i decreases with an increase in the embedded signature bits, while the value of T_o increases. Furthermore, the presented FFA based security technique provides more sturdy security due to the following reasons: (a) IP vendor specified encoding rules, (b) IP vendor specified keys to drive crypto-chain based security methodology, (c) IP vendor specified bit padding and embedding rules, (d) IP vendor specified truncation length, and (e) scheduling information obtained through transformed JPEG-CODEC SDFG.

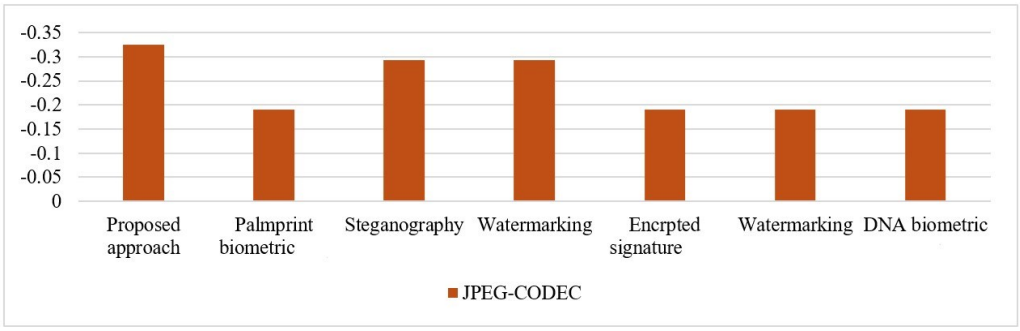
(ii). Design Cost:

Proposed low-cost multiphase encryption based security methodology: The design cost for the proposed approach is evaluated with the help of the design cost function explained in equation (3.9). The evaluated design cost before and after embedding the hardware security constraints for different image processing filter benchmarks is reported in Table 9.1. Further, Table 9.1 reports the low-cost resource architecture explored with PSO-based architecture exploration corresponding to the secured target image processing filter IP core. Table 9.1 shows that the proposed approach obtains a secured image processing filter IP core at zero design cost overhead. Moreover, Fig. 9.1. shows design cost comparison between the proposed approach and a

Table. 9.3: Results of presented techn

Benchmarks	FFA explor architec
JPEG-CODEC	6(+), 8

Design cost



Steganog
etric [10

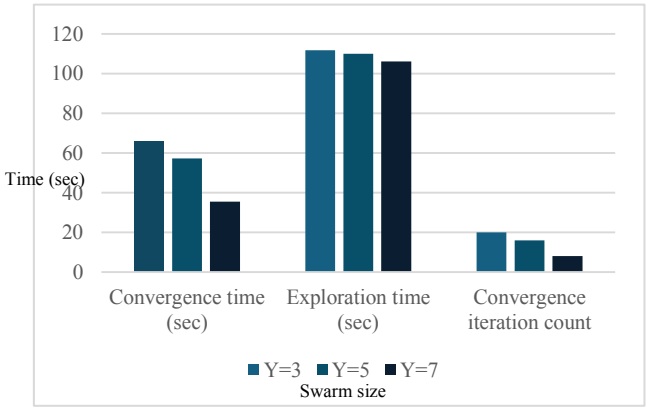
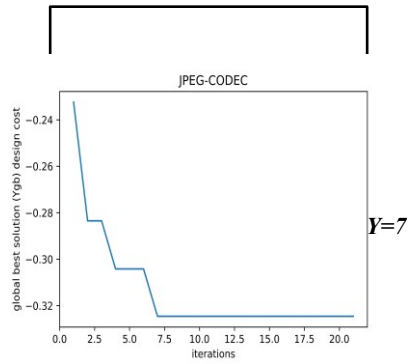
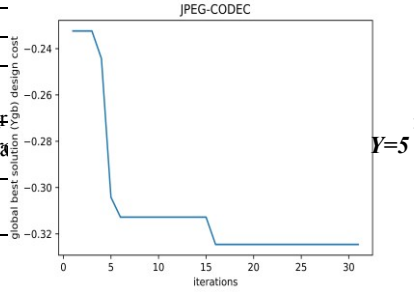


Fig. 9.10. Comparison of convergence time, exploration time and convergence iteration count corresponding to the swarm (population) sizes (Y) = 3, 5, and 7 for the proposed FFA-based security approach for secured JPEG-CODEC hardware IP core design

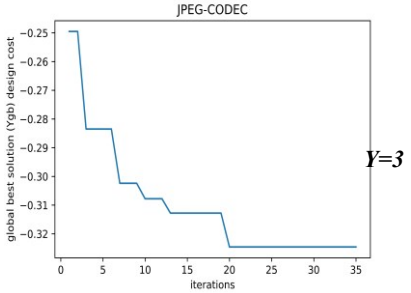
recent approach [31], [32], and [37]. As evident, the proposed approach offers significant improvement in quality of results (QoR). It is evident from Fig. 9.1 that the proposed approach achieves a significantly better QoR as compared to



Y=7



Y=5



Y=3

Proposed approach			
Swarm size	*Y=3*	*Y=5*	*Y=7*
Convergence time (sec)	65.80	57.22	35.47
Exploration time (sec)	111.73	109.84	106.11
Convergence iteration	20	16	7

Table. 9.4: Comparison of convergence and exploration time between pr ions corresponding to different swarm s for generating low-cost optimized architectural solution against benchmarks

[31], and [37] with an average design cost reduction of 56.92 percent for all benchmarks. Further, the average design cost reduction for the proposed approach compared to [32] is 36.84 percent for all the benchmarks, indicating a better QoR for the proposed approach than [32]. The integration of PSO-based architecture exploration with the proposed multi-phase encryption algorithm helps to determine low-cost optimal secured image processing filter IP core datapath with significantly lower design cost (higher QoR). Further, the convergence and exploration time of the proposed approach to obtain secured target filter IP cores are reported in Fig. 9.2. The proposed algorithm reports an average convergence time of 77.2ms and an average exploration time of 462.8ms. This shows that the proposed low-cost multi-phase

encryption algorithm converges to an optimal solution in an acceptable time. Moreover, the global best design cost reduction graph corresponding to different image processing filter benchmarks for PSO based architecture exploration is reported in Fig. 9.3. The graph depicts the reduction in the global best solution design cost over various iterations obtained using PSO based architecture exploration in order to determine low-cost final resource architecture. As evident from Fig. 9.3, the proposed algorithm converges to the global optimal solution in fewer iterations for smaller image processing filter applications such as vertical and horizontal embossment. However, it consumes 2 or 3 additional iterations for slightly larger applications, such as blur and sharpening filters. It is evident that the proposed algorithm can provide robust security with zero overhead in final design cost (i.e., no extra register is required).

Proposed low-cost crypto-chain based security methodology: The proposed FFA-based crypto-chain based security approach's design cost is computed using area and latency-based design cost function shown in eqn. (3.9). Table 9.3 reports the computed design cost corresponding to low-cost JPEG-CODEC IP core datapath before and after embedding the secret security constraints. Further, Table 9.3 highlights the FFA-DSE based low-cost optimized architectural solution corresponding to secured JPEG-CODEC hardware IP core, design area, and design latency. It is clearly evident from Table 9.3 that the proposed FFA-DSE based security incurs zero design cost overhead while securing the JPEG-CODEC hardware IP core. Next, a comparative study of design cost between the presented technique and [31], [32], [37], [43], [44], and [107] is illustrated in Fig. 9.9. As apparent from Fig. 9.9, the proposed methodology offers an average design cost saving of 71.11% in comparison to [44], [43], [33], and [107]. Further, it reports an average design cost saving of 11.08 % compared to [32] and [37]. This indicates that the presented approach provides a notable advancement in the quality of results (QoR). The incorporation of FFA based resource exploration with the presented key-driven crypto-chain based security algorithm facilitates the production of a low-overhead optimized secured JPEG-CODEC hardware IP core with a higher QoR (*i.e.,* lower design cost).

Table. 9.5: Comparison of probability of coincidence (C_i) and tamper tolerance (T_O) between the proposed approach, [44], [37], [32], [43], [31], and [107] corresponding to JPEG CODEC

Security approach		Security parameters	
	Embedded security constraints	(C_i)	(T_O)
Proposed approach	512	1.86E-02	1.34E+154
Palmprint biometric [44]	231	1.65E-01	3.45E+69
Steganography[37]	100	4.59E-01	NA
Watermarking [32]	240	1.54E-01	3.12E+144
Encrypted signature [43]	160	2.87E-01	1.46E+48
Watermarking [31]	240	1.54E-01	1.76E+72
DNA biometric [107]	128	3.69E-01	3.40E+38

Table. 9.6: Variation of probability of coincidence and tamper tolerance corresponding to proposed approach *w.r.t.* embedded crypto-chain signature bits into the design for JPEG-CODEC

Signature bits (embedded into the design)	Probability of coincidence *(C_i)*	Tamper tolerance *(T_o)*
32	7.79E-01	4.29E+09
64	6.07E-01	1.84E+19
128	3.69E-01	3.40E+38
256	1.36E-01	1.15E+77
512	1.86E-02	1.34E+154

Table. 9.7: Comparison of entropy between the proposed approach, [44], [32], [43], [31], and [107] corresponding to JPEG-CODEC

Benchmarks	Proposed approach
Proposed approach	5.61E-177
Palmprint biometric [44]	2.38E-87
Watermarking [32]	5.65E-73
Encrypted signature [43]	2.01E-87
Watermarking [31]	1.66E-111
DNA biometric [107]	2.9E-39

Table. 9.8: Optimality analysis of proposed technique for JPEG-CODEC

Parameters	Values
Spacing (SPA)	0.476
Generational distance (GEN)	0
Weighted metric (WEM)	0.232
Spread (SPD)	0.463

Next, Fig. 9.10 reports the exploration and convergence time of the presented approach to determine the secured JPEG-CODEC IP core datapath. The average exploration and convergence times are 109.26 sec and 52.83 sec, respectively. Additionally, Table 9.4 also reports the exploration time, convergence time, and convergence iteration corresponding to different population sizes (*i.e.,* 3, 5and 7). The convergence iteration count (*i.e.,* the iteration required by the DSE algorithm to converge to the global best architecture solution) decreases with an increase in the population size of fireflies. Table 9.4 establishes that the probability of convergence toward the global best solution increases with the increase in the population size. Further, Table 9.4 compares the proposed approach (that uses FFA-driven DSE) with

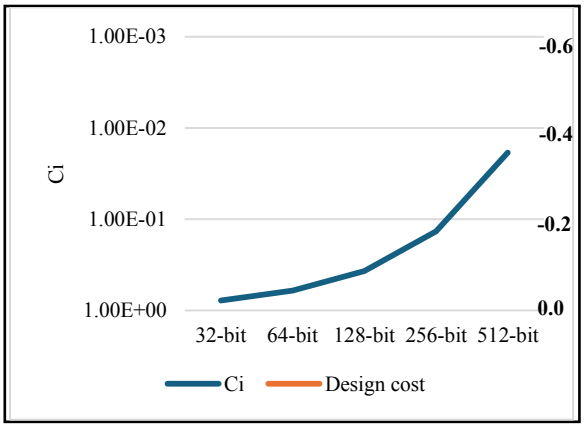


Fig. 9.12. Design cost vs probability of coincidence trade-off for proposed approach corresponding to varying signature sizes

PSO-driven DSE [78] regarding exploration time, convergence time, and convergence iteration corresponding to optimal secured JPEG-CODEC IP core datapath. Table 9.4 indicates that the presented technique supersedes [78] in terms of lower exploration and convergence time values to achieve the global optimal solution. Moreover, [78] report the advancement of PSO-driven DSE over other meta-heuristic based DSE approaches such as GA, ACO, etc.

Further, Fig. 9.11 depicts the design cost reduction graph while achieving the global best configuration corresponding to JPEG-CODEC hardware IP core for FFA-based architecture exploration. The graph illustrates the design cost reduction while reaching the global best solution over various iterations. As explained above in this sub-section, the FFA architecture exploration process converges in fewer iterations with an increase in firefly population size (shown in Fig. 9.11). Additionally, a design cost *vs.* probability of coincidence tradeoff for the proposed approach corresponding to varying signature sizes is shown in Fig. 9.12. As evidenced from Fig. 9.12, the proposed approach incurs zero design cot overhead with significantly lower value of C_i on increasing the embedded signature bits.

(iii). Entropy analysis: Entropy is described as the effort required by an adversary and uncertainty encountered in decoding the embedded hidden information inside the IP design [134]. The proposed approach's entropy is estimated using E_T:

$$E_T = ((1/2^Z)*(1/E_n)*(1/R)*(1/2^{64}))) \quad (9.3)$$

Where 'z' is the magnitude of generated signature, 'E_n' is IP vendor specified encoding rules, 'R' is the round computation's maximum value, and $(1/2^{64})$ is

the probability of finding the exact key hash buffer initialized value in SHA-512 cryptographic module (each hash buffer is initialized with pre-defined 64-bit value). Table 9.7 compares the entropy of the proposed approach with similar approaches [31], [32], [43], [44], and [107]. The proposed approach offers a stronger entropy value (lower probability value) compared to the prior state-of-the-arts.

9.1.2. Optimality analysis

The optimality assessment of the proposed security methodologies with respect to the determination of the explored architectural solution for secured image filters and JPEG-CODEC IP core is performed using the following optimality metrics: (a) spacing (SPA), (b) Generational distance (GEN), (c) weighted sum (WEM), and (d) spreading (SPD). Table 9.2 and 9.8 highlight the computed values corresponding to all optimality metrics for the proposed low-cost multiphase encryption and low-cost crypto-chain based security methodologies. A zero value of the 'G' shows that the list of obtained solutions using the presented approach lies on the true Pareto front. Likewise, a zero value (or marginally higher than zero) for the spacing parameter designates the even scattering of Pareto points on the curve. Next, the spread metric computes how comprehensively the true Pareto front is covered. The obtained lower value (*i.e.,* near to zero) corresponding to both spreading and spacing metric indicates the scattering evenness of the obtained solutions along with its extreme covering of true Pareto front.

9.2. Experimental Results: Enhanced Security for Hardware IPs Using IP Seller's Protein Molecular Biometrics and Facial Biometric-based Encryption Key

The experimental results of the proposed encrypted protein molecular biometric based security methodology (discussed in Chapter 4) are analyzed and discussed in this section.

9.2.1. Experimental setup and benchmarks

The specification of the system used to implement the proposed approach is processor-intel core2 duo, 2.10GHz RAM -3GB. A 15nm open-cell library

[86] is used to determine both the latency and area of a DSP hardware design, corresponding to resource constraints.

9.2.2. Security analysis: Analysis of PC and TT

The probability of coincidentally detecting embedded security constraints in an unsecured design is evaluated using PC metric. It is a measure of the false positive of the methodology. The probability of coincidence, that the same design with the security constraints (example, watermark) is produced by any other authors (IP vendors) must be reduced. The probability is proportional to the probability that any specific design is produced by a synthesis tool or by a manual design. It also is an indicator of the presence of digital evidence inside a secured design that can be used as a digital proof to handle IP piracy and verify true IP vendor ownership. A lower value of PC indicates a more robust security methodology with a higher value of digital evidence. Moreover, a lower value of PC helps in the generation of signatures with greater uniqueness (digital proof), which provides a smooth, definite, and robust differentiation between authentic and pirated IP cores during the detection process. Further, security against tampering attack is evaluated using the tamper tolerance ability (TT) of the design. The larger is the key-space, the harder it is for an attacker to find the exact embedded encrypted protein molecular signature to tamper. The formulas of PC and TT are already discussed in the previous section of this chapter.

The PC attained using proposed approach corresponding to DSP IP cores has been reported for varying amino acid sequence length and encrypted protein molecular signature size, as shown in Table 9.9. As evident from Table 9.9, with the increase in the number of amino acids in the chain, the encrypted protein molecular size increases, thereby resulting into larger number of security constraints producing lower PC value. The proposed protein molecular signature methodology is also compared with recent state-of-the art hardware security approaches based on fingerprint biometric [40] and chromosomal DNA [107]. The comparison of PC of proposed approach with [40] and [107] are reported in Table 9.10 and 9.11, respectively. As evident, the proposed approach attains lesser PC (higher strength of ownership proof) compared to both [40] and [107]. Further, PC comparison has also been

#Amino acids	Encrypted protein molecular signature size	FIR	DFT	4point DCT	8-point DCT	JPEG
		PC				
		Fingerprint image	# of embedded security constraints of fingerprint	PC of fingerprint approach [40]	Proposed amino acid chain of protein sequence	# of embedded security constraints of proposed approach	PC of proposed approach
50	200	2.5E-12	9.8E-3	5.6E-3	5.4E-7	6.3E-2	
		200	30	18	108	200	
150	599	Image:101_1	30	18	108	150	599	599	2.5E-4
		Image:101_2	315	350	8.0E-3				
250	990	Image:101_1	3.4E-13	9.8E-3	5.6E-3	5.4E-7	200	1.1E-6	799	1.6E-5
		Image:101_2	285	418	3.1E-3				
		Image:101_1	285	526	7.0E-4	108	250	990	990	1.1E-6
350	1382	Image:102_1	3.4E-13	9.8E-3	5.6E-3	5.4E-7	300	5.2E9	1184	8.0E-8
		Image:102_2	285	558	5.9E-4					
		Image:103_1	285	535	4.7E-4	108	350	1382	1382	5.2E-9
		Image:103_2	30	18						

Table. 9.9: Variation in PC Table. 9.10: Comparison of security in terms of PC for different DSP cores using proposed approach and IP fingerprinting [40]

reported for two different amino acid chains (protein sequence-1 and sequence-2) for different DSP cores, as shown in Table 9.12. Moreover, the PC comparison with conventional approaches [31], [36], [39], [43] has been shown in Table 9.13. As evident the proposed approach attains lower PC value (desirable) than the conventional approaches [31], [36], [39], [43]. This is because the proposed protein molecular signature capacitates the IP vendor to generate more secret security constraints as compared to generated using related approaches.

Table. 9.11: Comparison of PC w.r.t. related work [107]

Bench-marks [84]	Proposed		Related work [107]	
	Max. constraints	Pc	Max. constraints	Pc
FIR	225	0.9E-13	128	3.7E-8
ARF	306	1.79E-18	128	3.7E-8
DWT	110	2.1E-11	92	1.2E-9
JPEG	1408	3.6E-9	128	1.7E-1
MESA	1408	1.3E-13	128	3.7E-8

Next, the comparison of tamper tolerance of the proposed approach with [40] and [107] is shown in Table 9.14 and Table 9.15, respectively. As evident, the tamper tolerance of the proposed approach is significantly higher due to generation of more security constraints using protein molecular biometric signature than [40], [107]. Further, TT has also been compared corresponding to two different amino acid chains using proposed approach. As evident from Table 9.16, TT is higher for larger chain of amino acids (protein sequence-1). Moreover, the TT comparison with conventional approaches [31], [36], [39], [43] has been shown in Table 9.17. As evident the proposed approach attains higher TT value (desirable) than the conventional approaches [31], [36], [39], [43]. Thus, the proposed approach offers robust security than contemporary approaches against piracy and fraudulent ownership claim.

9.2.3. Design cost analysis

The design cost DC, pre and post-embedding of generated secret hardware security constraints corresponding to encrypted protein molecular signature is evaluated using the following design metric [32]:

$$DC = e_1 \frac{T_d}{T_m} + e_2 \frac{A_d}{A_m} \tag{9.4}$$

Where, P_i denotes the resource constraints of the design, T_d and A_d signify the security constraint embedded design latency and design area respectively, A_m and L_m denote the maximum possible area and maximum possible latency of the design. e_1 and e_2 denote weights of latency and area in the normalized cost function. The used design cost function is similar to design cost function discussed in previous section. Table 9.18 presents the design cost of proposed security approach post embedding security constraints corresponding to two different encrypted protein molecular signature extracted from two different protein sequences. The proposed approach incurs negligible design cost overhead post implanting facial biometric encrypted protein molecular

Amino acid chain sequence	#Amino acids	Fingerprint image Benchmarks	#Constraints Proposed approach	# of embedded FIR constraints of fingerprint [39]	security DFT approach [40]	TT of fingerprint DCT [36]	Proposed amino acid chain DCT protein sequence [41]	# of embedded security constraints of proposed approach	TT of proposed approach	% pr
Sequence-1	350	Image:101_1	350	9.82E-5	2.29E-5	2.26E-14	5.61E-7	5.12E-9	599	2.07E+180	
Sequence-2	51	Image:101_2	418	9.82E-5	6.76E-5	6.61E-7	4.24E-7	1.6E-16E-2	799	3.33E+240	4
		Image:101_8	526		2.19E+158		250		990	1.04E+298	4
		Image:102_3	538		8.99E+161		300		1184	~1.0E+358	
		Image:103_8	555		1.17E+167		350		1382	~1.0E+417	

Table 9.11 Comparison of TT of proposed approach and IP fingerprinting [40]

2: Variation in PC for two different protein sequences with varying strength of embedded security constraints for proposed approach

signature corresponding to different protein sequence of varying length. Further, a particular molecular signature of appropriate signature length can be chosen by considering the security-design cost tradeoff. The security (Pc)-design cost tradeoff for two different protein sequences (sequence-1 and sequence-2) corresponding to the varying strength of embedded security constraints is shown in Fig. 9.13. As evident from Fig. 9.13, the lesser value of probability of coincidence is achieved (desirable) with an increase in the number of embedded security constraints for both the protein sequences (1 and

Table. 9.15: Comparison of tamper tolerance (TT) w.r.t. related work [107]

Bench-marks [84]	Proposed		Related work [107]	
	Max. constraints	TT	Max. constraints	TT
FIR	225	5.39E+67	128	3.40E+38
ARF	306	1.30E+92	128	3.40E+38
DWT	110	1.29E+33	92	4.95E+27
JPEG	1408	1.0E+421	128	3.40E+38
MESA	1408	1.0E+421	128	3.40E+38

Table. 9.16: Variation in TT for two different encrypted protein sequence of varying length using proposed approach

Amino acid chain sequence	#Amino acids	# Constraints (z)	TT
Sequence-1	350	1382	~1.0E+417
Sequence-2	51	197	2.0E+59

Table. 9.17: Comparison of TT with related approaches

Benchmarks	Proposed approach	[39]	[36]	[31]	[43]
FIR	5.39E+67	1.32E+32	7.5E+38	7.0E+13	1.0E+3
ARF	1.30E+92	1020E+24	7.5E+38	7.0E+13	1.0E+3
1D-DWT	1.29E+33	1.20E+24	7.9E+16	1.0E+6	1.0E+3

Table. 9.18: Design cost of embedding encrypted protein molecular signature

Benchmarks [84]	Design cost of encrypted protein molecular signature implanted design corresponding to Sequence-1 (1408 digits)	Design cost of encrypted protein molecular signature implanted design corresponding to Sequence-2 (128 digits)
8-point DCT	0.473	0.473
FIR	0.569	0.567
ARF	0.476	0.473
DWT	0.615	0.617
JPEG	0.214	0.214
MESA	0.280	0.280

2). This is because embedding a higher number of security constraints makes it more challenging for an adversary to detect the authentic signature in an unsecured design version.

A lower value of PC depicts the presence of stronger digital evidence (definitive proof of ownership). Further, embedding more security constraints may impact the design cost. This is because embedding a higher number of security constraints into a smaller IP design may require extra storage element (registers) for accommodating storage variables post performing the local alteration as per the constraints embedding rule (a single register cannot be assigned with two storage variables at the same control step). Therefore, sometimes, new registers may be required to accommodate all the generated hardware security constraints into the design, which in turn may increase the design cost of the secured IP. However, in the proposed methodology an IP

Table. 9.19: Entropy of the proposed approach

Facial features (m)	Facial encryption key size (k)	Generated security constraints (s)	Entropy
5	32	1382	2.27E-452
9	64	1382	5.28E-462
11	83	1382	1.01E-467

Table. 9.20: Comparison of entropy between proposed approach and RNG techniques (crypto key based and SSL TRNG)

Proposed approach	Key based RNG techniques [31], [39]	SSL TRNG [133]
1.01E-467	2.98E-39	9.33E-302

Table. 9.21: Total computational time of the proposed approach with and without facial encryption mechanism

Benchmarks	Computational time without facial encryption (msec)	Computational time with facial encryption (msec)
FIR	744	824
ARF	842	922
DWT	840	920
JPEG	1598	1678
MESA	1152	1232

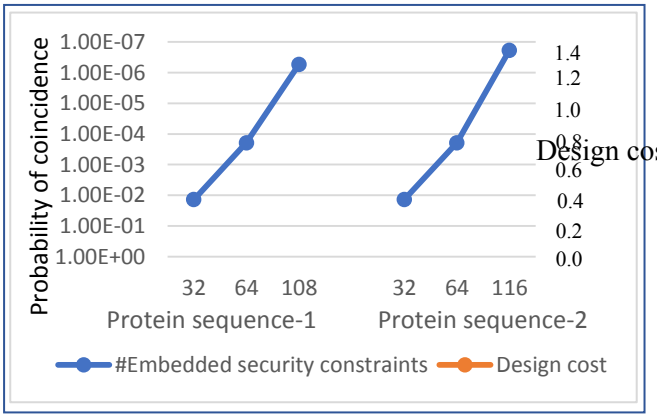


Fig. 9.13. Security-design cost trade-off for 8-point DCT corresponding to two different protein sequences for varying security constraints

vendor can choose the strength of hardware security constraints for embedding depending upon the size of application and desirable security strength.

9.2.4. Entropy analysis

The effort required to guess the exact protein molecular signature embedded can be quantified as: $e = 1 \div 2^{s}$, where $'s'$ stands for the number of generated protein molecular signature bits. For example, if the value of 's' is 1382 bits (protein molecular sequence-1), then the value of 'e' is 9.4742751E-417. This is an estimation of the entropy of the proposed approach in terms of the hardness of the adversarial guessing and effort. In the proposed approach, the order in which the amino acid is sequenced in polypeptide chain to obtain the protein molecular signature is unknown to an adversary. In the proposed approach, 20 different amino acid elements can be concatenated to generate a

robust protein molecular signature. This results in 20! (Factorial 20) permutations. Further, 11 different facial features can also be concatenated to generate facial encryption key. Therefore, the total effort *(T_E)* required from an adversarial perspective:

$$Entropy = 1/n! \times 1/2^s \times 1/m! \times 1/2^k \quad (9.5)$$

Where '*n*' is the number of the different amino acid elements used for the signature bit generation, '*m*' is the number of facial features for generating the encryption key and '*k*' is the size of facial encryption key. For example, when *n*=20, *s*=1382, m=11 and *k*=83 then T_E= 1.01E-467. The larger the value of *n, s, m* and *k*, the higher the effort required from an adversarial perspective. The entropy in terms of the hardness of the adversarial guessing and effort of the proposed approach corresponding to varying encryption key size (based on different facial features), is shown in Table 9.19. Table 9.19 presents the entropy corresponding to varying size of encryption key and embedded security constraints generated through proposed protein molecular signature. Additionally, the comparison of entropy of the generated signature using proposed approach with crypto key and semiconductor superlattice true random number generator (SSL-TRNG) based approach, is shown in Table 9.20. The proposed approach depicts improved entropy (lesser probability value) than techniques [31], [39] and [133]. Further, the computational time of the proposed approach corresponding to different benchmarks in case of 'with and without facial encryption module' is shown in Table 9.21. The computational time corresponding to facial encryption key generation block is adopted from [41].

9.3. Experimental Results: Securing Hardware IPs by Exploiting Statistical Watermarking Using Encrypted Dispersion Matrix and Eigen Decomposition Framework

The experimental results of the proposed statistical watermarking based security methodology (discussed in Chapter 5) are analyzed and discussed in this section.

9.3.1. Experimental setup and benchmarks

The experimental assessment of the proposed approach has been performed on a system with a 2.30 GHz processor and 4 GB RAM. A 15 nm technology scale based on the NanGate library [86] is used in the proposed approach to evaluate design area and latency corresponding to IP vendor selected resource configurations. The benchmarks used in the experimental analysis and their details (including their CDFGs, scheduling, register count, etc.) are available in [40],[41],[84]. The framework/tool used for characterizing area/latency is adopted from [41] and is publicly available at [85]. The maximum design space capacity (in terms of exhaustive resource configurations available) is as follows: 8-point DCT – 8; FIR – 64; ARF- 32; DWT- 20; JPEG- 2048 [46].

9.3.2. Analysis of attack scenarios

(i). Security Against Forgery Attack: In the proposed approach, it is not mandatory to store the embedded signature. The authentic IP vendor can easily perform IP ownership resolution by matching the security constraints with the embedded security constraints of the IP design. On the contrary, regeneration of original encrypted security constraints is impossible for the adversary as the regeneration process requires decoding of several security parameters such as (a) IP vendor selected p-bit key for resource configuration generation, (b) IP vendor chosen AES encryption key, (c) IP vendor chosen characteristic security parameters such as: *var* (A_d), *var* (L_d), *(cov* (A_d, L_d), eigen roots, and the number of resource configuration chosen, (d) concatenation rule for appending the encrypted characteristic secret data to generate the final encrypted signature, and (e) IP vendor specified encoding rule to convert the encrypted signature into security constraints. Therefore, the proposed approach is resilient against possible forgery attack by an SoC integrator. The used acronyms are explained in Chapter 5.

Benchmarks		Proposed approach		[41]	Proposed approach		[43]	[31]		[40]	
	Register count before embedding security constraints (c)	Embedded security constraints	(C_i)	Register count before embedding security constraints (c)	Embedded security constraints	(C_i)	Embedded security constraints	(C_i)	Embedded security constraints	(C_i)	Embedded security constraints	(C_i)
8-point DCT	16	81	1.00E-06	16	81	2.14E-03	5.00E-09	2.32E-09	1.28E-05	2.58E-04	199	2.64E-06
FIR	16	343	2.43E-10	16	81	3.45E-03	2.43E-10	1.12E-09	1.28E-05	2.58E-04	199	2.64E-06
ARF	16	441	4.35E-13	16	81	4.44E-03	4.35E-13	2.01E-09	1.28E-05	2.58E-04	199	2.64E-06
DWT	10	164	3.13E-08	10	81	1.06E-04	3.13E-08	1.45E-10	1.28E-08	1.39E-06	164	3.13E-08
JPEG-CODEC	137	896	1.41E-03	137	81	8.96E-01	1.41E-03	6.52E-09	1.28E-01	3.91E-01	199	2.32E-01

Table 9.23: Comparison of probability of constraints (C_i) between the proposed approach, [41], [43], [31], and [40]

(ii). Security Against Brute Force Attack: The proposed approach is capable of providing resistance against brute force attack due to its very high tamper tolerance ability. Due to greater size of the watermark signature generated, the signature space of the proposed approach is extremely large. Therefore, from an attacker's perspective, the probability of finding the exact signature combination used for embedding security constraints is extremely low. Henceforth, the proposed approach provides sturdy resilience against brute force attack used for tampering and/or removal of the embedded watermark.

(iii). Security against False Positive and Ghost Signature Search Attack: The implanted secret mark (watermark) should be seamlessly detectable to

Table. 9.25: Comparison of probability of coincidence (C_i) and tamper tolerance (Z_t) with variation in signature strength corresponding to JPEG-CODEC IP for the proposed approach

p-bit key based resource configuration	Variation in signature	C_i	Z_t
[1,4], [1,1], and E_K =128 bit	512-bit	2.34E-02	1.34E+154
[1,4], [1,1], [1,8], [1,5], and E_K =128 bit	896-bit	1.41E-03	5.28E+269
[1,4], [1,1], [1,8], [1,5], and E_K =256 bit	1792-bit	1.98E-06	2.79E+539

establish the proof of authorship. This indicates the credibility of the embedded watermark. No third party (*i.e.,* other than the IP owner) should be able to claim the watermark by chance. The probability of coincidence serves as a metric to assess the likelihood of coincidently detecting the exact security constraints within an unsecured IP design (false positive). The likelihood of a successful ghost signature search attack is the same as the probability of coincidence. In the proposed approach, the credibility of the embedded watermark is extremely high due to its lower probability of coincidence. This is because the proposed approach is capable of generating/embedding much larger number of secret watermarking constraints into the IP design. Therefore, the likelihood of launching successful ghost signature search attack is extremely low.

9.3.3. Security analysis: PC, TT and entropy analysis

Tables 9.22 and 9.23 report the comparison of the probability of coincidence between the proposed approach, [31], [40], [41], and [43]. In case of the proposed approach, the security constraints that can be generated and embedded are larger in size, therefore the maximum embedding possible (as reflected in tables 9.22 and 9.23) corresponding to different applications are higher and different, than prior approaches. On the other hand, for prior approaches [40], [41], and [43], the security constraints in Tables 9.22 and 9.23 are same, because the maximum possible generation of security constraints for embedding corresponding to different applications is exactly same. Further, Table 9.25 shows the comparison of the probability of coincidence with variation in signature strength corresponding to the JPEG-CODEC IP for the proposed approach. The proposed approach depicts a lower value of C_i as compared to the prior approaches [31], [40], [41], and [43], indicating stronger digital evidence due to the generation and embedding of a larger number of hardware security constraints. Table 9.25 shows, the C_i of the

proposed approach decreases with the increase in signature strength. This shows that the proposed approach carries the capability to generate and embed larger signatures for achieving a lower C_i value (*Note: the same resource configuration and encryption key have been used for proposed approach and prior works*). The proposed approach is capable of producing larger number of watermarking constraints for embedding as digital evidence due to the following security variables in the framework: (a) creation of mathematical watermark by extracting variance and covariance of IP vendor's chosen design space parameters, (b) creation of mathematical watermark by capturing variance and eigen roots of the IP vendor's chosen design space parameters (such as resource configurations, area, latency, etc. corresponding to the hardware application), (c) inherent capability of the proposed framework to extract secret design parametric information for creation of watermarking constraints. The above mentioned blocks exploited as watermarking framework, has not been used in any prior approaches for generation/embedding of security constraints. Therefore, the proposed approach is more robust against standard attacks and is capable of offering stronger digital evidence than prior approaches.

Next, Table 9.24 reports comparison of tamper tolerance between the proposed approach, [31], [40], [41], and [43]. The proposed approach depicts a higher value of TT/Z_t as compared to the prior approaches [31], [40], [41], and [43] due to the generation of a larger number of hardware security constraints and a greater signature space. Further, Table 9.25 compares the tamper tolerance with variation in signature strength corresponding to the JPEG-CODEC IP for the proposed approach. As evident from the Table 9.25, the proposed approach's tamper tolerance ability increases with an increase in signature strength.

Further, the entropy of the proposed approach can be represented as follows:

$$E_T = 1/2^d * 1/2^k * 1/2^p * (1/(A_M * M_M))^t \quad (9.6)$$

Where 'd' is the length of the final generated encrypted signature, 'k' is the size of the encryption key, 'p' is the size of the secret key used by the IP vendor for deciding the input resource configuration, 'A_M' is the maximum possible value

Table. 9.26: Results of proposed approach pre and post implanting the developed signature

Benchmarks	IP vendor selected resource configuration for scheduling	Initial design (*i.e.,* pre signature implanted unsecured design)				Final secured signature implanted design				Design cost overhead %
		Area (um^2)	Latency (ps)	Register count	Design cost	Area (um^2)	Latency (ps)	Design cost	Register count	
8-point DCT	1(+), 2(*)	182.45	1324.86	16	0.46	182.45	1324.86	0.46	16	0
FIR	1(+), 2(*)	106.95	2583.46	16	0.56	109.31	2583.46	0.57	19	0.17
ARF	1(+), 2(*)	182.45	2450.98	16	0.41	187.95	2450.98	0.41	23	0.72
DWT	2(+), 3(*)	272.10	1722.31	10	0.65	275.25	1722.31	0.65	14	0.15
JPEG-CODEC	6(+), 8(*)	824.96	3245.89	137	0.15	824.96	3245.89	0.15	137	0

Table. 9.27: Comparison of entropy between the proposed approach, [41], [43], [31], and [40]

Benchmarks	Proposed approach	[41]	[43]	[31]	[40]
8-point DCT	1.09E-44	1.03E-32	2.01E-87	8.63E-78	3.08E-65
FIR	3.35E-49	1.03E-32	2.01E-87	8.63E-78	3.08E-65
ARF	1.06E-47	1.03E-32	2.01E-87	8.63E-78	3.08E-65
DWT	6.99E-47	1.03E-32	2.01E-87	8.63E-78	3.08E-65
JPEG	9.94E-77	1.03E-32	2.01E-87	8.63E-78	3.08E-65

for adder, 'M_M' is the maximum possible value for the multiplier corresponding to the targeted application, and 't' is the IP vendor chosen number of resource configurations from the design space. Table 9.27 reports the comparison of entropy between the proposed approach, [31], [40], [41], and [43]. Further, as evident from Table 9.27, the proposed approach offers a stronger entropy value (*i.e.,* a lower probability value) compared to prior approaches [31], [40], [41], and [43]. Moreover, Table 9.28 illustrates a comparison between the entropy of the proposed approach, cryptographic key random number generator (RNG) [31] and a semiconductor superlattice true random number generator (SSL-TRNG) [133] based approaches. The proposed approach

Table. 9.28: Comparison of Entropy between Proposed Approach and RNG Techniques (Crypto Key based) for JPEG-CODEC

Proposed approach	Key based RNG [31]	SSL TRNG [133]
9.94E-577	8.63E-78	9.33E-302

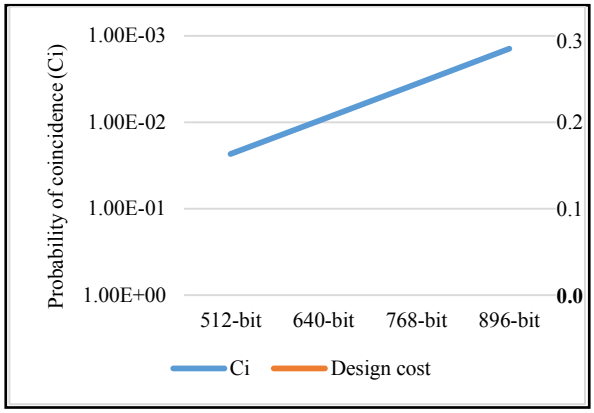


Fig 9.14. Design cost vs probability of coincidence trade-off for proposed approach for varying IP vendor signature sizes corresponding to JPEG-CODEC IP

demonstrates enhanced entropy, indicated by lower probability values compared to [31] and [133]. This is because the proposed approach offers more resistance and uncertainty to an attacker than other watermarking approaches [31], [40], [41], and [43], and [133]. More explicitly, an attacker needs to decode the following resistive parameters to overcome the uncertainty in regenerating or forging the exact security constraints, which offers stronger entropy than existing works: (a) The different p-bit key values for deciding the number of input resource configurations, (b) Design parameters chosen for dispersion matrix and Eigen matrix, (c) Encrypted signature strength, (d) Size of the encryption key (k), (e) Maximum possible value ('A_M', 'M_M') of the resources used, such as adders, multipliers, etc.

9.3.4. Design cost analysis

The design cost function used here is same as the design cost function discussed in prior sections of this chapter. Table 9.26 reports the IP vendor's chosen resource configuration for scheduling, its design area, latency, and cost corresponding to the design. As evident from Table 9.26, the proposed approach provides robust security at a negligible design overhead of 0.2 %. Further, Fig. 9.14 illustrates the design cost *vs.* probability of coincidence tradeoff for the proposed approach for varying IP vendor signature sizes. The value of C_i decreases with an increase in signature size at a constant value of design cost for JPEG-CODEC IP.

9.4. Experimental Results: Securing GLRT Cascade Hardware IP using IP Seller's Fingerprint and CIG Framework for ECG Detector

The experimental results of the proposed secure GLRT cascade hardware IP design framework (discussed in Chapter 6) are analyzed and discussed in this section. The experimental assessment of the proposed secure GLRT hardware IP core design for the ECG detector has been performed on a system with a 2.30 GHz processor and 4 GB RAM. A 15 nm technology scale based on the NanGate library [86] is used in the proposed approach to evaluate design area and latency corresponding to secure GLRT hardware IP core. The proposed design in this paper is a simulated version of the secure GLRT hardware IP at register transfer level (RTL). In case fabricated version of the design RTL is intended using the layout level information, standard CAD tool based design synthesis steps at lower levels can be employed to generate the layout level representation of the designed secure GLRT hardware IP (at RTL).

9.4.1. Analysis of attack scenarios

(i). Security against Forgery and Spoofing Attack: Forgery and spoofing are not feasible in the case of the proposed approach. This is because the biometric fingerprint minutiae points are pre-stored in an encrypted format in a safe database for validation/detection later. Any attacker endeavoring to forge the stored encrypted biometric fingerprint template would be unsuccessful in using it since he/she does not have the knowledge of the advanced encryption standard (AES) private key needed for decryption. Furthermore, a spoofing attack is not applicable in the case of IP piracy detection. This is because the attacker's goal is to evade/escape IP piracy detection by re-moving/tampering with the original embedded secret signature (security constraints). However, an attacker may attempt to launch spoofing to falsely claim IP ownership, which is not possible as spoofing of encrypted biometric fingerprint template requires forgery of the pre-stored encrypted biometric fingerprint template, which is not useful until an attacker is capable of successfully decrypting the encrypted template using the AES private key. Besides, an attacker also needs to decode the following security parameters:

(a) the number of features used in each template for fingerprint, (b) number of minutiae points and their exact 4-dimensional coordinates used for template regeneration, and (c) concatenation order of minutiae points used for regenerating the fingerprint template, for performing successful and accurate spoofing to falsely claim IP ownership. Therefore, forgery and spoofing attack is not possible in the proposed approach.

(ii). Security against Side Channel Attack (SCA) and Machine Learning (ML)-based Attack: The proposed security methodology stands strong against SCA and ML-based attacks, in contrast to PUF-based techniques. This resilience is attributed to the fact that the proposed security approach incurs a zero impact on the overall design cost of GLRT IP. Hence, the secret biometric fingerprint watermark embedded design does not leak significant side-channel information (such as delay, power, etc.). In the proposed approach, the biometric fingerprint watermark constraints (digital evidence) are embedded solely by locally modifying the register assignments (through swapping). Consequently, there is no noticeable impact on side channel parameters from an attacker's perspective. Furthermore, ML attacks are not applicable to a design with an embedded watermark (in case of proposed approach), as it does not rely on challenge-response pairs, which are prime targets for adversarial/modelling attacks, contrary to PUF-based systems.

(iii). Security against Brute-Force Attack (Tamper Tolerance): An attacker may attempt to perform a brute-force attack to remove/tamper the original embedded secret watermark (fingerprint security constraints). Tamper tolerance measures security in terms of the brute force attempts by adversaries to tamper the design or guess the exact signature combination. A higher T_Z value is desirable as it indicates a larger signature space, resulting in huge possible signature combinations. A higher T_Z value increases the complexity for attackers in their attempts to discover the exact watermark signature combination (security constraints). In the case of the proposed approach, T_Z is extremely high when launching a brute-force attack. Therefore, removing the embedded biometric fingerprint based watermark (security constraints) of the proposed approach is highly challenging.

Table. 9.29: Comparison of probability of coincidence (C_i) between the proposed fingerprint embedded secure GLRT cascade IP with facial biometric [40] embedded IP design and digital signature embedded IP design [39]

Proposed secure GLRT IP with fingerprint		Design with facial constraints [40]		Design with digital signature [39]	
Security constraints	C_i	Security constraints	C_i	Security constraints	C_i
250	3.57E-10	16	2.48E-01	16	2.48E-01
275	4.05E-11	32	6.17E-02	32	7.71E-02
300	4.60E-12	64	3.81E-03	64	3.81E-03
346	8.41E-14	81	8.69E-04	128	1.45E-05

Table. 9.30: Comparison of probability of coincidence (C_i) between the proposed fingerprint embedded secure GLRT cascade IP with encrypted signature embedded IP design [43] and hardware watermarking embedded IP design [31]

Proposed secure GLRT IP with fingerprint		Design with encrypted signature [43]		Design with watermark [31]	
Security constraints	C_i	Security constraints	C_i	Security constraints	C_i
250	3.57E-10	32	6.17E-02	32	6.17E-02
275	4.05E-11	64	3.81E-03	64	3.81E-03
300	4.60E-12	128	1.45E-05	128	1.45E-05
346	8.41E-14	160	8.99E-07	240	8.52E-10

Table. 9.31: Comparison of tamper tolerance (T_i) between the proposed fingerprint embedded secure GLRT cascade IP with facial biometric [40] embedded IP design and digital signature embedded IP design [39]

Proposed secure GLRT IP with fingerprint		Design with facial constraints [40]		Design with digital signature [39]	
Security constraints	T_i	Security constraints	T_i	Security constraints	T_i
250	1.80E+75	16	6.55E+04	16	6.55E+04
275	6.07E+82	32	4.29E+09	32	4.29E+09
300	2.03E+90	64	1.84E+19	64	1.84E+19
346	1.43E+104	81	2.41E+24	128	3.40E+38

Table. 9.32: Comparison of tamper tolerance (T_i) between the proposed fingerprint embedded secure GLRT cascade IP with encrypted signature embedded IP design [43] and hardware watermarking embedded IP design [31]

Proposed secure GLRT IP with fingerprint		Design with encrypted signature [43]		Design with watermark [31]	
Security constraints	T_i	Security constraints	T_i	Security constraints	T_i
250	1.80E+75	32	4.29E+09	32	4.29E+09
275	6.07E+82	64	1.84E+19	64	1.84E+19
300	2.03E+90	128	3.40E+38	128	3.40E+38
346	1.43E+104	160	1.46E+48	240	1.76E+72

(iv). Security against Ghost Signature Search Attack and False Positive/Watermark Collision (Probability of Coincidence): The credibility of the embedded secret watermark should be seamlessly detectable for the evidence of authorship. No third party (*i.e.,* other than the IP owner) should be able to claim the watermark by chance (in order words watermark collision should be as low as possible). The probability of coincidence serves as a metric to assess the likelihood of coincidently detecting the exact security constraints within an unsecured IP design (false positive). The likelihood of a successful ghost signature search attack is the same as the probability of

coincidence. Thus, a lower C_i value signifies more robust security and stronger credibility, indicating higher level of security.

9.4.2. Security analysis

Embedding the IP vendor's digital fingerprint template provides robust security to the designed GLRT hardware IP core of the ECG detector. This is because of the following reasons: (a) IP vendor's fingerprint digital template facilitates the integration of a unique natural identity with the design synthesis flow that increases the robustness of the proposed approach against IP piracy and false claim of IP ownership, and (b) the inclusion of several IP vendor specific parameters and rules such as concatenation rule, mapping rule, truncation length, etc. hinders the adversary from exactly regenerating the digital fingerprint template. The security analysis of the proposed secure GLRT hardware IP core is performed using established security metrics in the literature [25], [31], [32], [33] such as (a) probability of coincidence (false positive), (b) tamper tolerance. These metrics are already defined in section 9.1.1 of this chapter.

Tables 9.29 and 9.30 report the comparison of C_i between the proposed secure GLRT cascade hardware IP with embedded fingerprint and secure GLRT cascade hardware IP with facial biometric [40], digital signature [39], encrypted signature [43] and hardware watermarking [31]. The proposed secure GLRT cascade hardware IP core with embedded fingerprint surpasses [40], [39], [43], and [31], as clear from Tables 9.29 and 9.30. The determination of the larger number of security constraints in the proposed approach helps in achieving a lower value of C_i than [40], [39], [43], and [31]. Embedding a larger number of security constraints (*i.e.,* the presence of greater digital evidence in the design) increases the attacker's effort to locate the same security constraints in an unsecured GLRT hardware IP design. Next, Tables 9.31 and 9.32 show the comparison of T_i between the proposed secure GLRT cascade hardware IP with embedded fingerprint and secure GLRT cascade hardware IP with facial biometric [40], digital signature [39], encrypted signature [43] and hardware watermarking [31]. The proposed approach supersedes [40], [39], [43], and [31], as clear from Tables 9.31 and 9.32, due to the determination of larger security constraints. A higher value of

Table. 9.33: Design latency, area, and resource configuration of proposed secure GLRT IP before and after embedding fingerprint signature

Application	Resource configuration	Unsecured design (before fingerprint embedding)		Proposed fingerprint embedded secure design	
		Design area (um^2)	Design latency (ps)	Design area (um^2)	Design latency (ps)
GLRT cascade hardware IP core	2(+), 3(*)	273.67	1656.07	273.67	1656.07

Table. 9.34: Design cost, leakage power, register count and resource configuration of proposed secure GLRT hardware IP before and after embedding fingerprint signature

Application	Resource configuration and registers	Unsecured design (before fingerprint embedding)		Proposed fingerprint embedded secure design	
		Design cost	Leakage power	Design cost	Leakage power
GLRT cascade hardware IP core	2(+), 3(*), and 13 registers	0.43	8.57 μw	0.43	8.57 μw

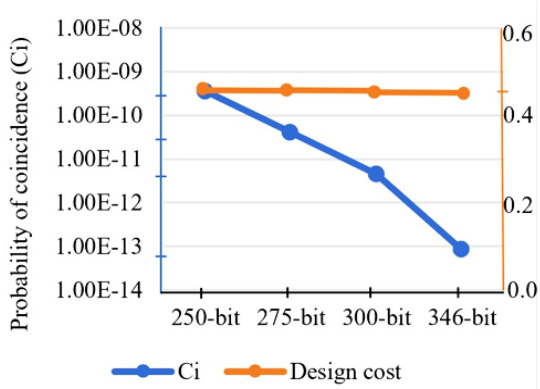


Fig. 9.15. Design cost vs probability of coincidence trade-off for proposed secure GLRT cascade for varying fingerprint signature sizes

T_i signifies a larger signature space because of greater signature combinations. This makes it significantly harder by increasing the attacker's effort/time to guess the exact embedded signature combination from larger signature space.

9.4.3. Design cost analysis

The design cost is computed using equation (9.7):

$$D_c = \left(e_1\left(\frac{L_T}{L_{max}}\right)\right) + \left(e_2\left(\frac{A_R}{A_{max}}\right)\right) \tag{9.7}$$

where, $e_1 = e_2 = 0.5$ for giving equal weightage to design latency and area, L_T and A_R are design latency and area corresponding to GLRT hardware IP. Further, L_{max} and A_{max} are their corresponding maximum latency and area, respectively. Table 9.33 reports the design latency, area, and IP vendor specified resource configuration of the proposed secure GLRT hardware IP for the ECG detector before and after embedding the fingerprint template. As evident from Table 6, the proposed secure GLRT hardware IP core with

fingerprint biometric provides robust security at zero design cost overhead (*i.e.,* no extra registers are required). As evident from Table 9.34, the power overhead of the proposed approach is 0%, as the post fingerprint embedded design does not incur any extra functional units or registers. Table 9.34 also reports the leakage power value of pre-embedded and post fingerprint embedded GLRT IP core. Therefore, the proposed secure GLRT IP core produces reliable designs as low-power designs result into lesser heat dissipation.

Further, Fig. 9.15 highlights the design cost *vs.* probability of coincidence tradeoff for the proposed secure GLRT cascade for varying fingerprint signature sizes. The value of C_i decreases with an increase in signature size at constant value of design cost.

9.5. Experimental Results: Exploiting Voice Biometric-Based Watermarking Framework for Securing Hardware IP Cores

The experimental results of the proposed voice biometric based security methodology (discussed in Chapter 7) are analyzed and discussed in this section.

9.5.1. Experimental setup and benchmarks

The experimental validation and analysis of the proposed approach are performed on Intel(R) Core (TM) i7-9700 CPU @ 3.00GHz and 4GB RAM. We analyze the impact of a varying number of selected voice biometric features on the signature template size. Further, the security of the proposed approach has been analyzed in terms of (i) strength of IP ownership proof using the probability of coincidence metric, (ii) tamper tolerance of the proposed voice signature, and (iii) security against forgery attack. Additionally, we evaluated the impact of voice biometric-based hardware security on design metrics such as area and latency (delay) and the final design cost for various DSP benchmarks [84]. A 15nm technology scale-based Nan Gate library [86] has been used to compute the design metrics.

9.5.2. Effect of feature selection on voice signature size and sensitivity analysis

The impact of varying number of selected features, such as pitch and intensity, on voice signature size has been analyzed for different benchmark applications in Table 9.35. This table compares different signature strengths generated from a different number of selected timestamps (Ts) or pitch and intensity features (such as 10, 12, 14, and 15 Ts). As shown, the signature size increases with the increase in Ts or the number of features extracted. Hence, a larger number of Ts can be chosen to generate a higher size signature for higher security. Further, a slight variation in the timestamp does not affect the signature size but affects the voice features, *viz.* pitch, intensity, Jitter and Shimmer. This results in a completely different signature template. It is noteworthy that there is an infinite number of timestamps possible for extracting pitch and intensity values for a particular voice sample. This may lead to an infinite number of possible signatures through variations in timestamps or features. However, the genuine IP vendor being aware of the selected value of the timestamp, can generate the same signature template for verification. On the other hand, the variation in signature with timestamps thwarts an adversary from reproducing the same signature template for misuse or during verification.

***Scalability of the proposed approach*:** The proposed approach can generate very long size signatures by extracting the features (pitch and intensity) at large number of timestamps. Further, the strength of the signature generated from a voice sample also depends on the size of the targeted IP core. Therefore, the proposed technique is scalable in nature as it provides the ability to accommodate more constraints in big size applications like moving picture expert group (MPEG) IP for securing them.

9.5.3. Security evaluation and comparison with prior watermarking techniques

The security evaluation of the proposed approach is performed using probability of coincidence and tamper tolerance security metric. The probability of coincidence is the measurement of the strength of IP ownership

Table. 9.35: Variation in Signature Size (in bits) with variation in selected number of Timestamps (Ts) or pitch and intensity values for Voice-001

Variation in number of timestamps	Signature size variation				
	IIR	IDCT	FIR	MPEG	4point DCT
10 Ts	191	191	191	191	77
12 Ts	205	220	220	220	77
14 Ts	227	249	249	249	77
15 Ts	234	262	264	264	77

Table. 9.36: Comparison of PC value with variation in signature size for varying # of timestamps or pitch and intensity values

#Ts variation	IIR	8-point IDCT	FIR	MPEG	4point DCT
10 Ts	7.12E-7	4.43E-6	4.43E-6	9.62E-4	3.42E-5
12 Ts	2.52E-7	6.81E-7	6.81E-7	3.35E-4	3.42E-5
14 Ts	4.94E-8	1.04E-7	1.04E-7	1.64E-4	3.42E-5
15 Ts	2.94E-8	4.53E-8	3.98E-8	6.76E-5	3.42E-5

Table. 9.37: Comparison of probability of coincidence of proposed with other watermarking techniques

Water-marking techniques	Pc			
	4-point DCT	8-point IDCT	FIR	MPEG
Proposed	3.42E-5	4.53E-8	3.94E-8	6.76E-5
[36]	2.08-2	7.8E-2	7.8E-2	7.27E-1
[37]	3.2E-3	1.9e-6	4.9E-4	1.3E-2
[40]	3.54E-2	2.36E-4	4.93E-7	2.79E-4
[41]	2.0E-5	5.36E-3	5.36E-3	5.25E-2
[44]	2.71E-2	3.13E-4	3.35E-7	2.24E-4
[31]	2.63E-1	3.79E-1	3.79E-1	5.79E-1
[43]	2.63E-1	5.24E-1	5.24E-1	6.95E-1

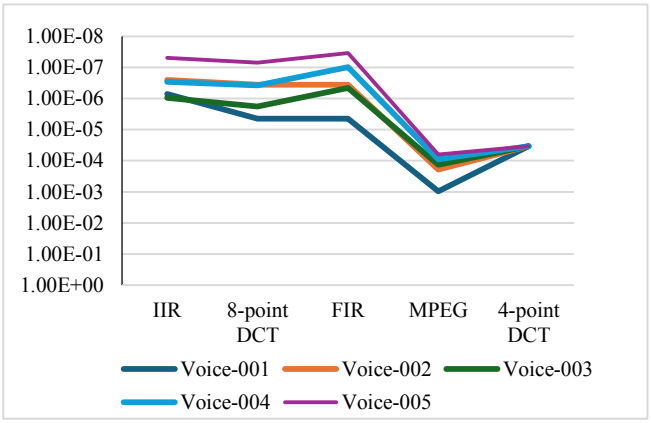


Fig. 9.16. Comparison of P_C of the proposed approach among 5 different voice samples (15 timestamps)

proof. The lower the probability of coincidence, the higher the strength of IP ownership proof will be. The PC value corresponding to the target applications decreases with the increase in signature size (or timestamps) as shown in Table 9.36. Further, the PC achieved using the proposed methodology is compared with different state-of-the-art watermarking techniques [31], [36], [37], [40], [41], [43] and [44] in Table 9.37. As shown, the proposed approach offers a lower probability of coincidence as compared to state-of-the-arts [31], [36], [37], [40], [41], [43] and [44], indicating higher strength of IP ownership.

Table. 9.38: Comparison of tamper tolerance with variation in signature size for varying timestamps or pitch and intensity values

#Ts	8-point IDCT	FIR	MPEG	4point DCT
10 Ts	9.8E+114	9.8E+114	9.8E+114	2.3E+46
12 Ts	6.8E+132	6.8E+132	6.8E+132	2.3E+46
14 Ts	8.2E+149	8.2E+149	8.2E+149	2.3E+46
15 Ts	5.5E+157	8.8E+158	8.8E+158	2.3E+46

Table. 9.39: Comparison of tamper tolerance of proposed with other watermarking techniques [36],[40], [41], [44], [31], and [43]

Water-marking techniques	Pc			
	4-point DCT	8-point IDCT	FIR	MPEG
Proposed	2.28E+46	5.49E+157	8.78E+158	8.78E+158
[36]	5.08E+50	4.1E+67	4.1E+67	4.1E+67
[40]	8.47E+11	5.39E+57	2.25E+107	2.25E+107
[41]	2.41E+24	2.41E+24	2.41E+24	2.41E+24
[44]	7.62E+12	4.36E+59	1.64E+110	1.64E+110
[31]	1.02E+3	3.27E+4	3.27E+4	3.27E+4
[43]	1.04E+6	1.04E+6	1.04E+6	1.04E+6

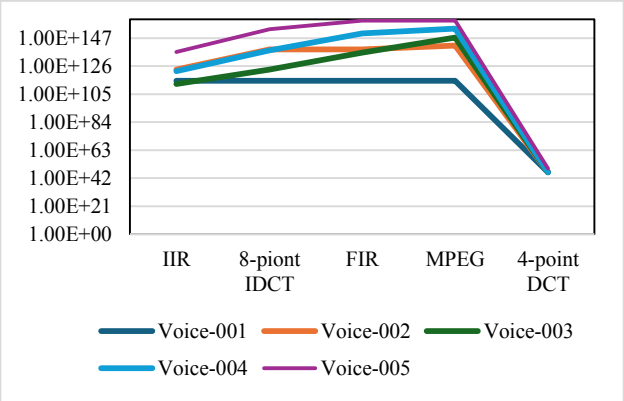


Fig. 9.17. Comparison of TT of the proposed approach among 5 different voice samples (15 timestamps)

The higher PC is achieved due to the ability to generate a larger voice signature template, embedding a higher number of hardware security constraints. Further, the PC value for five different voice samples is reported in Fig. 9.16.

Next, the TT value of the proposed voice signature increases with an increase in the signature strength as shown in Table 9.38. The higher tamper tolerance helps preserve the author's signature for seamless verification of IP. Further, Table 9.39 shows that the proposed approach has a higher value of TT as compared to state-of-the-art [31], [36], [40], [41], [43] and [44]. The high value of TT is achieved due to the larger size of the signature and the multiple encoding used. Further, the TT value for five different voice samples is also reported in Fig. 9.17.

9.5.4. Design cost analysis and security trade-offs

Benchmarks	Baseline design (before signature embedding)			Voice biometric signature embedded design			Design cost overhead %				
	Design area	Design latency	Design cost	Design area	Design latency	Design cost	proposed	[36]	[37]	[44]	Signature size (256-bits)
8-point IDCT	182.7	2244.346	0.447	182.7	2244.346	0.447	0.0	0.1	0.0	0.1	0.449
FIR	106.954	2583.469	0.569	106.954	2583.469	0.569	0.0	2.3	0.0	0.8	0.569
MPEG	305.135	1391.099	0.436	305.135	1391.099	0.436	0.0	7.8	0.0	0.0	0.436
4-point DCT	176.161	662.428	0.563	178.52	662.428	0.565	0.3	5.5	0.1	0.2	0.565

Table. 9.40: Area(um), latency (ns) and design cost analysis of proposed approach and comparison with related works

The design cost function used in the proposed approach has already been discussed in prior sections of this chapter. Table 9.40 illustrates the design area, execution latency or delay, and design cost before and after embedding the voice signature constraints for different benchmarks. Further, it reports the incurred design cost overhead due to the embedding of voice biometric-based signature into the designs and compares it with the most recent watermarking/steganography approaches [36], [37], [44]. The average design cost overhead of the proposed technique is 0.18% which is lesser than the related approaches [36], [44]. However, the design cost overhead of [37] is slightly lower as it embeds a lesser number of constraints than the proposed approach. The overhead of PUF based techniques [135], [136] are reported as

follows. In the case of [135], area overhead of 52% in LUT, 55% in slices count and delay overhead of 17%; and in the case of [136], 5.16% delay overhead has been reported previously [136]. Hence, this implies that the proposed watermarking is more design cost-efficient than the PUF based techniques. Further, Table 9.41 presents a security cost tradeoff analysis of the proposed approach. It is evident from the table that embedding a smaller signature (such as 32 bits) and a larger one (such as 256 bits) has almost a similar effect on design cost with minimal overhead. Hence, the proposed approach is capable of offering higher security using larger signature strength without significantly affecting design cost.

9.6. Experimental Results: HLS-Based Exploration of Low-Cost (Optimal) Functional Trojan-Resistant Hardware IP Designs

The experimental results of the proposed HLS based Trojan resistant approach (discussed in Chapter 8) are analyzed and discussed in this section

9.6.1. Experimental setup and benchmarks

The proposed approach, [45], [46] and [53] have been implemented using a system with 2.30 GHz processor with 4 GB RAM. Further, ten runs have been performed to obtain the final result, and the average value has been reported. We have given equal weightage (e_1 = e_2 = 0.5) to both delay and area objective to the proposed PSO-DSE-based optimal Trojan resistant hardware design approach. Providing equal weightage to the design cost function (discussed in Chapter 8) for evaluating fitness ensures both design area and schedule delay are given equal priority. This is because, from an SoC integrator perspective, designing Trojan resistant may cause area and delay overhead concurrently. Therefore, it is necessary for the SoC integrator to provide equal preference to both the design parameters during fitness evaluation. On the contrary, providing unequal weightage to e_1 and e_2 in the context of area and delay shall provide an imbalanced fitness evaluation, causing exploration of results that are not truly optimal. Henceforth, providing equal weightage to e_1 and e_2 = 0.5 in the context of normalized design cost function has been established practice for design space exploration in HLS

[45], [78]. However, the proposed approach is scalable, *i.e.,* capable of handling small and large hardware applications in minimal exploration time. The PSO-DSE settings used for generating results (based on empirical analysis) for proposed framework are acceleration coefficient (b_1 and b_2) = 2; inertia weight (ω) = linearly decreasing between 0.9 to 0.1; swarm size (n) = 3 or 5 or 7; random numbers (r_1 and r_2) = 1; stopping criterion = T_1 or T_2 [79].

9.6.2. Security evaluation and comparison with prior techniques

The proposed PSO-driven TMR-based approach has been compared with the state-of-the-art methodology proposed by [45], [46], [53]. The proposed work and [45], [46], [53] deal specifically with Trojans that affect the computational output. The cost function used in both approaches above considers the complete SoC design area, including all types of functional resources used and the required execution time. The area and the execution time have equal weightage in the design cost function. PSO-DSE module integrated with the proposed Trojan resistance TMR logic is used for generating optimal Trojan resistance architecture of hardware IP cores, while in [45], [46] PSO-DSE module is used to generate low-cost Trojan detectable architectures. The PSO-DSE module in the proposed approach accepts input from the TMR schedule (allocated with three IP vendors), while in [45], [46], PSO-DSE accepts inputs from the DMR schedule (allocated with two IP vendors). Further, the PSO-DSE module in the proposed approach generates outputs in the form of Trojan-resistant TMR schedule latency and area for iteratively pruning the search space of Trojan-resistant architectures, while in [45], [46], PSO-DSE produces output in the form of Trojan detectable schedule latency and area. Moreover, in the proposed approach, the distinct vendor allocation policy deployed inside PSO-DSE based scheduling and allocation differs from the one used in [45], [46]. Finally, the particle configurations used during the initialization process in the proposed approach are different than [45], [46].

Further, Table 9.45 compares the proposed approach with [45] and [46] in terms of design cost and overhead. The proposed TMR-based design is more robust than [45] and [46] in terms of security. This is because the proposed approach uses three distinct vendors to implement the TMR-based design, thus providing greater defense to the hardware IP core in terms of Trojan detection

Table. 9.42: Comparison of convergence time (msec) for generating trojan resistant hardware designs *w.r.t.* swarm size '*n*'

S. No.	Benchmark	*n*=3	*n*=5	*n*=7
1.	4-point DCT	16	24	27
2.	FIR	196	200	200
3.	ARF	32	57	96
4.	JPEG	44	48	93
5.	DWT	65	68	68

Table. 9.43: Comparison of exploration time (msec) for generating trojan resistant hardware designs *w.r.t.* swarm size '*n*'

S. No.	Benchmark	*n*=3	*n*=5	*n*=7
1.	4-point DCT	96	130	190
2.	FIR	674	867	973
3.	ARF	231	416	868
4.	JPEG	299	485	1048
5.	DWT	267	281	353

Table. 9.44: Area, cost, and time of proposed TMR based design

S. No	DSP IP	Global optima	T_{TMR} (µs)	A_{TMR} (au)	Design cost
1.	4-pointDCT	3(+), 9(*)	45.635	25808	-0.120
2.	FIR	3(+), 9(*)	79.77	28272	-0.165
3.	ARF	3(+), 6(*)	264.1	20880	-0.173
4.	JPEG	3(+), 3(*)	88.76	13488	-0.059
5.	DWT	6(+), 9(*)	112.37	31904	-0.091

ability and Trojan resistance compared to the [45] and [46], which provide only Trojan detection. From Table 9.45, we can observe that with a minimal average overhead of 4.6%, the proposed IP core design can provide Trojan resistance (isolation) compared to [45] and [46], which only provide Trojan detection. Therefore, the proposed approach enables the Trojan resistance capability of hardware IP/SoC design with minimal design overhead and ensures correct output functionality through a distinct multivendor allocation policy. However, the probability of obtaining identical wrong outputs from any two of the TMR units is improbable. Further, Table 9.46 compares the proposed approach with [53] in terms of design cost overhead while handling Trojans. The proposed approach provides Trojan defense (isolation) at an average design cost overhead of 2.5 % compared to [53], which only provides Trojan detection.

Further, Table 9.47 provides a comparison of Trojan defense capability between the proposed approach and [45], [46] in terms of the respective output generated from the Trojan detection DMR unit in [45], [46] and Trojan resistant TMR unit in the proposed approach, corresponding to test vectors, random sequences, malicious vendor (assuming VD_1 has Trojan (*i.e.,* vendor VD_1 inserted with Trojan logic); or assuming VD_2 has Trojan (i.e., vendor VD_2 inserted with Trojan logic)) and its defense status. Since the proposed

Table. 9.46: Comparison of the proposed approach with [53]

S. No.	Benchmark	Final architecture solution for proposed approach	Cost of the final solution for proposed approach	Cost of the final solution [53]	% Change (overhead)	Cost of the final solution [53]	% Change (overhead)
1.	4-point DCT	3(+), 9(*)	-0.120	2(+), 6(*)	-0.121	-0.120	0.82
2.	FIR	3(+), 9(*)	-0.165	8(+), 8(*)	-0.176	-0.165	6.25
3.	ARF	3(+), 6(*)	-0.173	2(+), 4(*)	-0.187	-0.173	7.48
4.	JPEG Sample	4(+), 3(*)	-0.059	8(+), 4(*)	-0.062	-0.059	4.8
5.	DWT	6(+), 9(*)	-0.091	4(+), 6(*)	-0.095	-0.091	4.09

Table. 9.46: Comparison of the proposed approach with [45], [133]

S. No.	Benchmark	Final architecture solution for proposed approach	Final architecture solution [45], [133]	Cost of the final solution for proposed approach	Cost of the final solution [53]	Cost of the final solution [45], [133]	% Change (overhead)	Cost of the final solution [53]	% Cha (overhe
1.	4-point DCT	3(+), 9(*)	2(+), 6(*), 9(*)	-0.120	2(+), 6(*)	-0.121	-0.120	0.82	-0.121	0.82
2.	FIR	3(+), 9(*)	2(+), 6(*), 9(*)	-0.165	8(+), 8(*)	-0.176	-0.165	6.25	-0.152	0
3.	ARF	3(+), 6(*)	2(+), 4(*), 6(*)	-0.173	2(+), 4(*)	-0.187	-0.173	7.48	-0.187	7.48
4.	JPEG Sample	4(+), 3(*)	2(+), 2(*), 3(*)	-0.059	8(+), 4(*)	-0.062	-0.059	4.8	-0.055	0
5.	DWT	6(+), 9(*)	4(+), 6(*), 9(*)	-0.091	4(+), 6(*)	-0.095	-0.091	4.09	-0.095	4.09

approach handles only IPs where the functionality of the third-party modules (IPs) is changed, hence, the process of Trojan insertion in the HDL codes of IP modules (such as adders, subtractors, etc.) was imitated by functionality altering the hardware operation through insertion of Trojan logic in the HDL code as follows: '+' function was modified to '□' function, '-' function to '+' function and '*' function to '+' function. As evident from this table, the proposed approach is capable of providing Trojan detection and isolation (Trojan resistance) with the aid of TMR logic, distinct multivendor allocation policy and voter, which [45], [46] are unable to provide. The greater the number of computations in the proposed design, the greater the vulnerability,

Table. 9.47: Comparison of Trojan defense capability of proposed approach and [45], [46]

Benchmarks	Test Vectors for U_{OG}, U_{DP} and U_{TR}	Random sequences (Samples for demonstration)	Malicious Site (vendor)	Outputs [45], [46]		Defense Status [45], [46]	Outputs [Proposed]				Defense Status [Proposed]	Defense capability in percentage
				O_{OG} (VD_1)	O_{DP} (VD_2)		O_{OG} (VD_1)	O_{DP} (VD_2)	O_{TR} (VD_3)	Voter output (isolated true value)		
IIR filter	$T_m = \{i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}\}$, where $[i_n]_1^{128}$	{1, 2, 1, 2, 1, 2, 1, 2, 1, 2}	V_1	243	10	Trojan Detected but not isolated	243	10	10	10	Trojan Detected and isolated (providing Trojan resistance)	100%
FIR filter	$T_m = \{i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}, i_{11}, i_{12}, i_{13}, i_{14}, i_{15}, i_{16}\}$, where $[i_n]_1^{128}$	{1, 2, 1, 2, 1, 2, 1, 2, 1, 2}	V_1	65536	74	Trojan Detected but not isolated	65536	74	74	74	Trojan Detected and isolated (providing Trojan resistance)	100%
4-point DCT	$[i_n]_1^{128}$ $T_m = \{i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8\}$, where $[i_n]_1^{128}$	{1, 2, 1, 2, 1, 2, 1, 2}	V_2	8	81	Trojan Detected but not isolated	8	81	8	8	Trojan Detected and isolated (providing Trojan resistance)	100

Note: T_m is the set of the random input test vectors corresponding to a benchmark; 'n' is the number of primary inputs required for an application; 'm' is the number of possible vectors sets for an application; 'i_n' is the n^{th} primary input

as each untrustworthy 3PIP core used during the hardware design is considered a potential vulnerability. In the proposed approach, each such potential vulnerability has been addressed by using individual distinct vendor allocation policy to both original, duplicate, and triplicate units of the design of TMR logic. The number of potential vulnerabilities the proposed TMR-based approach handles for various applications is shown in Fig. 9.18.

9.6.3. Design cost and optimality analysis

Table 9.42 compares convergence time (in msec) for generating Trojan resistant hardware design *w.r.t.* swarm size (n) 3, 5, and 7. Further, Table 9.42 dictates that the convergence time of the proposed method to find the optimal Trojan-resistant DSP core architectural solution nominally increases with the increase in swarm size, while the same final solution is obtained in each case. The increase in the convergence time *w.r.t.* swarm size is because the time required for the cost computation increases with the rise in the number of

Table. 9.48: Optimality Analysis of the proposed approach

Benchmarks	GEN	Spacing (SPA)	Spread (SPD)	Weighted metric (WEM)
DCT-4	0.00	0.00	0.00	0.00
FIR	0.00	1.16	0.54	0.27
ARF	0.00	1.41	0.66	0.132
JPEG-sample	0.00	0.00	0.00	0.00
DWT	0.00	0.47	0.34	0.17

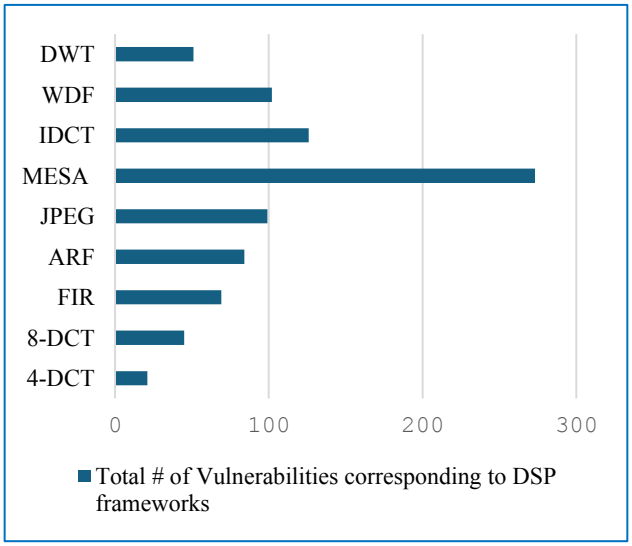


Fig. 9.18. Number of potential untrustworthy 3PIPs vulnerabilities handled using proposed approach

swarm positions (n) or resource architectures. A similar pattern can also be observed in the case of Table 9.43 (showing exploration time (in msec) for generating Trojan-resistant hardware design *w.r.t.* swarm size (n) = 3, 5, and 7). Because of this, the computation time increases with the rise in the number of swarm positions while yielding the exact optimal solution in each case.

Table 9.44 shows the global best resource architecture obtained for Trojan resistant hardware IP core using PSO-DSE. For instance, if we take the example of the FIR IP core, then three adders and nine multipliers (from three different vendors) are required to design the proposed TMR-based Trojan-resistant logic. Further, Table 9.44 depicts the hardware area (A_{TMR}), corresponding execution time (T_{TMR}), the global best resource architecture, and the design cost for the respective TMR-based Trojan-resistant IP core. The benchmarks (adopted from [84]) have been evaluated for design area and the latency (delay). Further, the total design area and latency are computed using (8.1) and (8.2), respectively. Here, the normalized design cost is computed by providing equal weightage to both hardware area and execution time, as shown in (8.3). For instance, for the FIR IP core, the execution time

required for the scheduling Trojan-resistant design with explored 3(+) and 9(*) resources is 79.77μs, while the area required and design cost are 28272au and 0.41, respectively.

Further, in order to determine the quality of the Trojan-resistant solution explored using PSO, it is essential to analyze the optimality of the proposed approach using various key metrics, such as generational distance (GEN), spacing (SPA), spreading (SPD) and weighted sum (WEM). The optimality analysis for the proposed strategy is presented in Table 9.48. A zero value for the GEN parameter indicates that the solutions obtained using the proposed methodology are on the true Pareto front. Similar to the GEN parameter, a value of zero (or slightly greater than zero) for spacing denotes a uniform distribution of Pareto points along the curve. Additionally, the spread metric measures the extent to which the true Pareto front is covered. It is clear from Table 9.48 that the proposed approach is capable of achieving a lower value (*i.e.,* either zero or close to zero) for both spacing and spreading metrics, which shows that the achieved solutions cover the extremes of the true Pareto Front in addition to their uniform distribution on the curve.

Chapter 10

Conclusion and Future work

10.1 Conclusion

The rise of DSP, multimedia, machine learning, and healthcare applications has become central to modern electronics ecosystem. Designing secure hardware IP cores for these SoC is therefore vital. Given the global nature of SoC design, where multiple design houses collaborate from different regions, it is crucial to establish trust in the hardware design before integrating third-party IP cores. This necessitates the development of robust security measures to counter external hardware threats, which can negatively impact not only the end-user but also the system itself and the IP vendors/designers. This thesis introduces various innovative security techniques for securing IP cores in computing and consumer electronics systems. The objectives achieved are the following:

- Proposed two novel security techniques were proposed: (a) PSO-driven multi-phase encryption and (b) firefly algorithm-based, low-cost crypto-chain frameworks for designing secure IP cores in image processing and JPEG-CODEC applications. The PSO-driven multi-phase encryption employs a series of strong security layers including bit manipulation, row diffusion, TRIFID cipher computation, alphabetic substitution, and byte concatenation. These layers work together to form a highly resilient and tamper-resistant signature aimed at countering IP piracy and false ownership claim. The threat model assumes that the IP vendor is defending against attacks from attackers in SoC integrators or foundries. In the second approach, the low-cost crypto-chain method incorporates an encoding mechanism specified by the IP vendor, combined with cryptographic keys, SHA-512 hash slices, and mapping rules to produce security constraints. These are embedded into the hardware IP core, which has been optimized using the firefly algorithm, ensuring that the IP cores remain protected against piracy with minimal design overhead.

- Proposed a protein molecular biometric signature based watermarking approach, derived from a human body sample, with a facial biometric encryption key unique to the IP vendor. During the HLS process, an encrypted version of this protein molecular signature is embedded into the design, preventing counterfeit IPs and false IP ownership claim. This dual-layer security approach, combining molecular and facial biometrics, offers stronger tamper resistance and a lower probability of coincidence than current state-of-the-art methods.

- Proposed a statistical modelling based hardware watermarking approach using 2D encrypted dispersion matrix combined with an eigen decomposition security framework. This approach secures IP cores from piracy and false IP ownership claim by embedding a tamper-resistant mathematical watermark signature within the hardware design. This framework relies on the IP vendor's specific resource configurations, combined with AES, to ensure robust protection. The embedded watermark is highly resistant to tampering (as it is generation from the hardware design space parameters) and facilitates the detection of pirated versions with minimal design cost overhead.

- Proposed a novel secure GLRT cascade IP cores with embedded fingerprint biometric-based watermarking constraints, specifically for medical applications like ECG detectors. The fingerprint watermark is unique to the IP seller and ensures that only authentic GLRT IP cores are integrated into medical devices. This approach not only prevents counterfeit IPs from entering sensitive medical systems but also guarantees the safety and accuracy of critical medical devices like ECG detectors and cardiac pacemakers. The inclusion of authentic GLRT IP in ECG SoCs ensures that only verified cores are used, thereby safeguarding patient health from the risks associated with counterfeit components.

- Proposed a security/watermarking technique using voice biometrics, which captures unique features such as jitter, shimmer, pitch, and

intensity at different timestamps of IP seller's voice sample to create a distinct watermark signature. This signature is embedded into the target IP core design during the HLS process. This approach provides robust security with enhanced tamper tolerance and a lower probability of coincidence, making it a robust method for securing IP cores against piracy and false ownership claim.

- Proposed a low-cost solution to protect hardware IP designs from functional hardware Trojans. This method combines PSO-based design space exploration with triple modular redundancy (TMR) to create a secure, low-cost SoC. The system employs a distinct multi-vendor-specific allocation policy for original, duplicate, and triplicate units. Even if one unit is compromised by a Trojan, the system continues to function correctly. Further, the integration of PSO based design space exploration module leads to the generation of optimized Trojan resistant design. This methodology provides comprehensive security against functional Trojans with minimal design overhead, ensuring that malicious blocks are isolated from the rest of the system.

10.2 Future work

This thesis has presented various alternative paradigms hardware security techniques for generating low-cost secure hardware IP cores/designs corresponding to different data intensive applications from the various domains such as DSP, electronic, multimedia, healthcare applications, etc. Future research in the security of hardware IP cores can focus on multiple promising directions to address evolving challenges.

- One key area involves developing *security-aware synthesis flows using HLS and physical design* methodologies. Security features, such as watermarks, can be covertly embedded during the design process to detect and deter IP piracy with greater resistance to tampering and minimal false positives. By incorporating these security constraints in both the HLS and physical design phases (*e.g.,* floorplanning and routing), we can ensure that they impose minimal overhead on the final hardware layout.

- Another important focus is creating *hybrid security solutions for IP cores* to handle a wider range of hardware security threats, particularly for fields like medical devices and the Internet of Things (IoT). Additionally, more robust alternative paradigms for securing hardware IP design can be explored than the proposed ones (such as protein molecular, statistical modelling, and voice biometric based hardware watermarking).

- To explore beyond traditional watermarking techniques, such as using *molecular and cognitive biometrics* (*e.g.,* DNA or cognitive data) for IP authentication. These techniques offer a stronger defense against piracy and replication, with the potential to provide more distinct and robust watermark signature.

- Combining watermarking with logic encryption can offer *dual protection against IP piracy and reverse engineering* for data-intensive hardware applications, such as multimedia and healthcare devices. Further, integrating fault tolerance mechanisms alongside piracy detection systems would further enhance security.

- Future work should also aim to overcome the limitations of current systems, such as improving the *generation of optimal datapath for loop-based applications* (such as FIR, etc.) through advanced optimization algorithms. This would enable efficient resource configuration with reduced costs in terms of area and latency. Additionally, *exploration of optimal watermarking constraints*, so that they do not incur any design cost overhead is also crucial from future research perspective.

- Furthermore, *investigating hardware Trojan (HT) attacks introduced via compromised computer-aided design (CAD) tools*, including HLS tools, is another critical area. Rogue insiders or external attackers could potentially insert malicious code during the design cycle. Detecting such Trojans at the RTL level, especially attacks that degrade performance or exhaust system resources, will be crucial for ensuring hardware security in future designs.

REFERENCES

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

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

[3] 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.

[4] J. A. Roy, F. Koushanfar and I. L. Markov (2008), "EPIC: ending piracy of integrated circuits," in *Proc. DATE*, Munich, pp. 1069-1074.

[5] F. Koushanfar *et al*. (2012), "Can EDA combat the rise of electronic counterfeiting?," in *Proc. DAC*, San Francisco, CA, pp. 133-138.

[6] B. Colombier and L. Bossuet (2015), "Survey of hardware protection of design data for integrated circuits and intellectual properties," *IET Computers & Digital Techniques*, vol. 8, no. 6, pp. 274-287.

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

[8] C. Pilato, S. Garg, K. Wu, R. Karri and F. Regazzoni (2018), "Securing hardware accelerators: a new challenge for high-level synthesis," *IEEE Embedded Syst. Lett.*, vol. 10, no. 3, pp. 77-80.

[9] G. He, C. Dong, Y. Liu and X. Fan (2020), "IPlock: An Effective Hybrid Encryption for Neuromorphic Systems IP Core Protection," *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 612-616.

[10] Y. Xie, C. Bao and A. Srivastava (2017), "Security-Aware 2.5D Integrated Circuit Design Flow Against Hardware IP Piracy," *in Computer*, vol. 50, no. 5, pp. 62-71.

[11] X. Wang, Y. Zheng, A. Basak and S. Bhunia (2015), "IIPS: Infrastructure IP for Secure SoC Design," *IEEE Trans.Comput*., vol. 64, no. 8, pp. 2226-2238.

[12] A. Sengupta, S. P.Mohanty (2016), “High-Level Synthesis of Digital Circuits in the Nanoscale Mobile Electronics Era”, *IET Book: Nano-CMOS and Post-CMOS Electronics: Circuits and Design*, pp: 219 - 261.

[13] B. K. Mohanty and P. K. Meher (2016), “A High-Performance FIR Filter Architecture for Fixed and Reconfigurable Applications,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 2, pp. 444-452.

[14] S. Sitjongsataporn, A. Thitinaruemit and S. Prongnuch (2021), “Implementation of High Level Synthesis for Adaptive FIR Filtering on Embedded System,” *2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST)*, pp. 257-260.

[15] S. Chen, J. Jung, P. Song, K. Chakrabarty and G. -J. Nam (2020), “BISTLock: Efficient IP Piracy Protection using BIST,” *2020 IEEE International Test Conference (ITC)*, pp. 1-5.

[16] M. T. Arafin, A. Stanley and P. Sharma (2017), “Hardware-based anti-counterfeiting techniques for safeguarding supply chain integrity,” *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-4.

[17] M. Yasin, J. J. Rajendran, O. Sinanoglu and R. Karri (2016), “On improving the security of logic locking,” *IEEE Trans. on CAD of Integr. Circuits Syst.*, vol. 35, no. 9, pp. 1411-1424.

[18] S. A. Islam, L. K. Sah, and S. Katkoori. 2020. High-Level Synthesis of Key-Obfuscated RTL IP with Design Lockout and Camouflaging. *ACM Trans. Des. Autom. Electron. Syst*. 26, 1, Article 6 (January 2021), 35 pages.

[19] S. M. Saeed, A. Zulehner, R. Wille, R. Drechsler and R. Karri (2019), “Reversible Circuits: IC/IP Piracy Attacks and Countermeasures,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 11, pp. 2523-2535.

[20] A. Anshul and A. Sengupta, "A Survey of High Level Synthesis Based Hardware Security Approaches for Reusable IP Cores [Feature]," in *IEEE Circuits and Systems Magazine*, vol. 23, no. 4, pp. 44-62, Fourthquarter 2023.

[21] M. Rostami, F. Koushanfar and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," in *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283-1295, Aug. 2014.

[22] D. Mouris, C. Gouert and N. G. Tsoutsos (2022), "Privacy-Preserving IP Verification," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 7, pp. 2010-2023.

[23] A. Hroub and M. E. S. Elrabaa (2022), "SecSoC: A Secure System on Chip Architecture for IoT Devices," *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2022, pp. 41-44.

[24] M. Rathor and A. Sengupta (2021), "Signature Biometric based Authentication of IP Cores for Secure Electronic Systems," *2021 IEEE International Symposium on Smart Electronic Systems (iSES)*, 2021, pp. 384-388.

[25] W. Hu, C. -H. Chang, A. Sengupta, S. Bhunia, R. Kastner and H. Li (2021), "An Overview of Hardware Security and Trust: Threats, Countermeasures, and Design Tools," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 6, pp. 1010-1038.

[26] M. Srivastava, P. SLPSK, I. Roy, C. Rebeiro, A. Hazra and S. Bhunia, "SOLOMON: An Automated Framework for Detecting Fault Attack Vulnerabilities in Hardware," *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, 2020, pp. 310-313.

[27] D. Ziener and J. Teich (2008), "Power signature watermarking of IP cores for FPGAs," *J. Signal Process. Syst*., vol. 51, no. 1, pp. 123–136.

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

[29] M. Ni and Z. Gao (2005), "Detector-based watermarking technique for soft IP core protection in high synthesis design level," *Proc. CCS*, Hong Kong, pp. 1348–1352.

[30] M. Lewandowski and S. Katkoori, "A Darwinian Genetic Algorithm for State Encoding Based Finite State Machine Watermarking," *20th*

International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 2019, pp. 210-215.

[31] F. Koushanfar, I. Hong, and M. Potkonjak (2005), "Behavioral synthesis techniques for intellectual property protection," *ACM Trans. Des. Autom. Electron. Syst*., vol. 10, no. 3, pp. 523–545.

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

[33] M. Potkonjak and I. Hong (1999), "Behavioral Synthesis Techniques for Intellectual Property Protection," *IEEE/ACM Design Automation Conference*, New Orleans, Louisiana, United States, pp. 849-854.

[34] B. Le Gal and L. Bossuet (2012), "Automatic low-cost IP watermarking technique based on output mark insertions," *Design Autom. Embedded Syst*., vol. 16, no. 2, pp. 71–92.

[35] R. Karmakar and S. Chattopadhyay (2020), "Hardware IP Protection Using Logic Encryption and Watermarking," *2020 IEEE International Test Conference (ITC)*, 2020, pp. 1-10.

[36] A. Sengupta, D. Roy and S. P. Mohanty (2018), "Triple-Phase Watermarking for Reusable IP Core Protection During Architecture Synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 4, pp. 742-755.

[37] A. Sengupta and M. Rathor (2019), "IP core steganography for protecting DSP kernels used in CE systems," *IEEE Trans. Consum. Electron.*, vol. 65, no. 4, pp. 506-515.

[38] T. Yu and Y. Zhu, "A new watermarking method for soft IP protection," *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Xianning, China, 2011, pp. 3839-3842.

[39] A. Sengupta, E. R. Kumar and N. P. Chandra (2019), "Embedding digital signature using encrypted-hashing for protection of DSP cores in CE," *IEEE Trans. Consum. Electron*., vol. 65, no. 3, pp. 398-407.

[40] A. Sengupta and M. Rathor (2020), "Securing hardware accelerators for CE systems using biometric fingerprinting," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 9, pp. 1979-1992.

[41] A. Sengupta and M. Rathor (2021), “Facial Biometric for Securing Hardware Accelerators,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 1, pp. 112 – 123.

[42] J. Chen and B. C. Schafer, "Watermarking of Behavioral IPs: A Practical Approach," *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, 2021, pp. 1266-1271.

[43] E. Castillo, L. Parrilla, A. Garcia, U. Meyer-Baese, G. Botella and A. Lloris, "Automated Signature Insertion in Combinational Logic Patterns for HDL IP Core Protection," *2008 4th Southern Conference on Programmable Logic*, Bariloche, Argentina, 2008, pp. 183-186.

[44] A. Sengupta, R. Chaurasia and T. Reddy, "Contact-Less Palmprint Biometric for Securing DSP Coprocessors Used in CE Systems," *IEEE Transactions on Consumer Electronics*, vol. 67, no. 3, pp. 202-213, Aug. 2021.

[45] A. Sengupta, S. Bhadauria and S. P. Mohanty, "TL-HLS: Methodology for Low Cost Hardware Trojan Security Aware Scheduling With Optimal Loop Unrolling Factor During High Level Synthesis," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 4, pp. 655-668, April 2017.

[46] A. Sengupta, S. Mohanty, Trojan security aware DSP IP core and integrated circuits, in: IET Book: IP Core Protection And Hardware-Assisted Security For Consumer Electronics, 2019. Book ISBN: 978-1-78561-799-7.

[47] A. J. Hu, "Formal hardware verification with BDDs: an introduction," *1997 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, PACRIM. 10 Years Networking the Pacific Rim, 1987-1997*, Victoria, BC, Canada, 1997, pp. 677-682 vol.2.

[48] M. Bushnell, V. Agrawal, Essentials of electronic testing for digital, memory, and mixed-signal VLSI circuits, IEEE Circuits and Devices Mag 17, 2001, pp. 39–40. July.

[49] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," in *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10-25, Jan.-Feb. 2010.

[50] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi and B. Sunar, "Trojan Detection using IC Fingerprinting," *2007 IEEE Symposium on Security and Privacy (SP '07)*, Berkeley, CA, USA, 2007, pp. 296-310.

[51] S. Narasimhan *et al*., "Multiple-parameter side-channel analysis: A non-invasive hardware Trojan detection approach," *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Anaheim, CA, USA, 2010, pp. 13-18.

[52] X. Wang, H. Salmani, M. Tehranipoor and J. Plusquellic, "Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis," *2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, Cambridge, MA, USA, 2008, pp. 87-95.

[53] J. J. Rajendran, O. Sinanoglu and R. Karri, "Building Trustworthy Systems Using Untrusted Components: A High-Level Synthesis Approach," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 9, pp. 2946-2959, Sept. 2016.

[54] H. Martin, L. Entrena, S. Dupuis and G. Di Natale, "A Novel Use of Approximate Circuits to Thwart Hardware Trojan Insertion and Provide Obfuscation," *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, Platja d'Aro, Spain, 2018, pp. 41-42.

[55] M. M. Siboni and S. Mohammadhossein Shekarian, "An Effective Application of Obfuscated TMR to Tackle Hardware Trojans," *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, Iran, 2019, pp. 12-17.

[56] N. B. Gunti and K. Lingasubramanian, "Neutralization of the Effect of Hardware Trojan in SCADA System Using Selectively Placed TMR," *2017 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, Bhopal, India, 2017, pp. 99-104.

[57] H. Li, A. Abdelhadi, R. Shi, J. Zhang and Q. Liu, "Adversarial Hardware With Functional and Topological Camouflage," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 5, pp. 1685-1689, May 2021.

[58] M. Abderehman, R. Gupta, R. R. Theegala and C. Karfa, "BLAST: Belling the Black-Hat High-Level Synthesis Tool," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3661-3672, Nov. 2022.

[59] A. Kulkarni, Y. Pino and T. Mohsenin, "Adaptive real-time Trojan detection framework through machine learning," *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, McLean, VA, USA, 2016, pp. 120-123.

[60] P. Fezzardi, F. Ferrandi and C. Pilato, "Enabling Automated Bug Detection for IP-Based Designs Using High-Level Synthesis," *IEEE Design & Test*, vol. 35, no. 5, pp. 54-62, Oct. 2018.

[61] M. Abderehman, T. Rakesh Reddy and C. Karfa, "DEEQ: Data-driven End-to-End EQuivalence Checking of High-level Synthesis," *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 2022, pp. 64-70.

[62] N. Fern and K. -T. Cheng, "Evaluating Assertion Set Completeness to Expose Hardware Trojans and Verification Blindspots," *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Florence, Italy, 2019, pp. 402-407.

[63] U. Alsaiari and F. Gebali, "Hardware Trojan Detection Using Reconfigurable Assertion Checkers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 7, pp. 1575-1586, July 2019.

[64] S. Sadangi, S. Baraha, D. K. Satpathy and P. K. Biswal, "FPGA implementation of spatial filtering techniques for 2D images," *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, India, 2017, pp. 1213-1217.

[65] D. Tsiktsiris, D. Ziouzios and M. Dasygenis, "A portable image processing accelerator using FPGA," *2018 7th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, Thessaloniki, Greece, 2018, pp. 1-4.

[66] J. V. Vourvoulakis, J. Lygouras and J. A. Kalomiros, "Acceleration of Image Processing Algorithms Using Minimal Resources of Custom

Reconfigurable Hardware," *2012 16th Panhellenic Conference on Informatics*, Piraeus, Greece, 2012, pp. 68-73.

[67] D. Davalle, B. Carnevale, S. Saponara, L. Fanucci and P. Terreni, "Hardware accelerator for fast image/video thinning," *2014 IEEE International Conference on Imaging Systems and Techniques (IST) Proceedings*, Santorini, Greece, 2014, pp. 64-67.

[68] A. Ansari, K. Gunnam and T. Ogunfunmi, "An efficient reconfigurable hardware accelerator for convolutional neural networks," *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, 2017, pp. 1337-1341.

[69] M. -C. Chang, Z. -G. Pan and J. -L. Chen, "Hardware accelerator for boosting convolution computation in image classification applications," *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)*, Nagoya, Japan, 2017, pp. 1-2.

[70] S. B. Choi, S. S. Lee and S. J. Jang, "CNN inference simulator for accurate and efficient accelerator design," *2019 International SoC Design Conference (ISOCC)*, Jeju, Korea (South), 2019, pp. 283-284.

[71] C. Shu, W. Pang, H. Liu and S. Lu, "High Energy Efficiency FPGA-Based Accelerator for Convolutional Neural Networks Using Weight Combination," *2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)*, Wuxi, China, 2019, pp. 578-582.

[72] Koff DA, Shulman H. An overview of digital compression of medical images: can we use lossy image compression in radiology? Can Assoc Radiol J. 2006 Oct;57(4):211-7.

[73] S. B. Gokturk, C. Tomasi, B. Girod and C. Beaulieu, "Medical image compression based on region of interest, with application to colon CT images," *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Istanbul, Turkey, 2001, pp. 2453-2456 vol.3.

[74] Chen YY. Medical image compression using DCT-based subband decomposition and modified SPIHT data organization. Int J Med Inform. 2007 Oct;76(10):717-25.

[75] R. H. Alshammari *et al*., "A Survey On Image Filtring Algorithms," *2020 3rd International Conference on Computer*

Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 2020, pp. 1-5.

[76] T. Huang, J. Burnett and A. Deczky, "The importance of phase in image processing filters," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 6, pp. 529-542, December 1975.

[77] J. Yang, G. Zhu and Y. -Q. Shi, "Analyzing the Effect of JPEG Compression on Local Variance of Image Intensity," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2647-2656, June 2016.

[78] V. Mishra and A. Sengupta, "MO-PSE: Adaptive Multi Objective Particle Swarm Optimization Based Design Space Exploration in Architectural Synthesis for Application Specific Processor Design", *Elsevier Journal on Adv. in Eng. Softw.*, Vol. 67, pp. 111-124, 2014.

[79] I. C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters, Vol 85, Issue 6, 2003, pp 317-325.

[80] P. Sarkar, A. Sengupta, S. Rathlavat and M. K. Naskar, "Designing Low-Cost Hardware Accelerators for CE Devices [Hardware Matters]," *IEEE Consumer Electronics Magazine*, vol. 6, no. 4, pp. 140-149, Oct. 2017.

[81] Sait, S. M., & Youssef, H. (1999). *VLSI physical design automation: theory and practice* (Vol. 6). World Scientific Publishing Company.

[82] X.-S. Yang, X. He, "Firefly algorithm: Recent advances and applications," *Int. J. Swarm Intell.*, vol. 1, no. 1, 2013, pp. 36–50.

[83] 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.

[84] Express benchmark suite, University of California San Diego, 2016, https://www.ece.ucsb.edu/EXPRESS/benchmark/.

[85] CAD for Assurance, IEEE Hardware Security and Trust Technical Committee, https://cadforassurance.org/tools/ip-ic-protection/faciometric-hardware-security-tool/, accessed on Jan 2022.

[86] 15 nm open cell library. [Online]. Available: https://si2.org/open-cell-library/, last accessed on Jan. 2020.

[87] G. Martin and G. Smith (2009), "High-Level Synthesis: Past, Present, and Future," in IEEE Design & Test of Computers, vol. 26, no. 4, pp. 18-25.

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

[89] 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.

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

[91] 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.

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

[93] D. Roy and A. Sengupta (2019), "Multilevel Watermark for Protecting DSP Kernel in CE Systems, "*IEEE Consum. Electron. Mag.*, vol. 8, no. 2, pp. 100-102.

[94] X.-S. Yang, "Firefly algorithms for multimodal optimization," in Proc. *5th Int. Conf. Stochastic Algorithms: Foundations and Applications (SAGA'09)*, 2009, pp. 169–178.

[95] A. Sengupta, R. Sedaghat, "A Multi Structure Genetic Algorithm for Integrated Design Space Exploration of Scheduling and Allocation in High Level Synthesis for DSP Kernels", Elsevier Journal of Swarm and Evolutionary Computation, Vol. 7, pp. 35-46, 2012.

[96] V. Krishnan and S. Katkoori, "A genetic algorithm for the design space exploration of datapaths during high-level synthesis," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 213-229, June 2006.

[97] A. Sengupta and S. Bhadauria, "Automated exploration of datapath in high level synthesis using temperature dependent bacterial foraging optimization algorithm," *2014 IEEE 27th Canadian Conference on*

Electrical and Computer Engineering (CCECE), Toronto, ON, Canada, 2014, pp. 1-5.

[98] Y. Kao, M. Chen, & Y. Huang, A Hybrid Algorithm Based on ACO and PSO for Capacitated Vehicle Routing Problems. *Mathematical Problems in Engineering*, 2012.

[99] Z. Zukhri, U. Islam, & Z. Zukhri, A Hybrid Optimization Algorithm based on GeneticAlgorithm and Ant Colony Optimization. *International Journal of Artificial Intelligence and Application*, Vol. 4, pp. 63–75, 2013.

[100] F. Wihartiko,H. Wijayanti, and F. Virgantari, "Performance Comparison of Genetic Algorithms and Particle Swarm Optimization for Model Integer Programming Bus Timetabling Problem," *In: IOP Conference Series: Materials Science and Engineering*, vol. 332, pp. 012020, 2018.

[101] B. Ge, Y. Han, & C. Bian, Hybrid Ant Colony Optimization Algorithm for Solving the OpenVehicle Routing Problem. *Journal of Computers*, 27(4), pp. 41–54, 2016.

[102] Duong, V.-A.; Park, J.-M.; Lim, H.-J.; Lee, H. "Proteomics in Forensic Analysis: Applications for Human Samples". Appl. Sci. 2021, 11, 3393.

[103] E.D. Merkley, D.S. Wunschel, K.L. Wahl, K.H. Jarman, "Applications and challenges of forensic proteomics," Forensic Science International, Volume 297,2019, Pages 350-363, ISSN 0379-0738.

[104] Parker GJ, Leppert T, Anex DS, Hilmer JK, Matsunami N, Baird L, et al. (2016) "Demonstration of Protein-Based Human Identification Using the Hair Shaft Proteome," PLoS ONE 11(9): e0160653.

[105] Chen, Lorenzo & White, Camille & Babbitt, Patricia & Mcleish, Michael & Kenyon, George. (2000), "A Comparative Study of Human Muscle and Brain Creatine Kinases Expressed in Escherichia coli," Journal of protein chemistry, 19. 59-66. 10.1023/A:1007047026691.

[106] K. Steendam, M. Ceuleneer, M. Dhaenens, et al. "Mass spectrometry-based proteomics as a tool to identify biological matrices in forensic science," Int J Legal Med 127, 287–298 (2013).

[107] A. Sengupta and R. Chaurasia, “Securing IP Cores for DSP Applications Using Structural Obfuscation and Chromosomal DNA Impression,” *IEEE Access*, vol. 10, pp. 50903-50913, 2022.

[108] RR. Chamley *et al*, ECG interpretation. *European Heart Journal*; 40: 32, 2019, pp. 2663-2666.

[109] S. Jarvis, S. Saman, Cardiac system 1: anatomy and physiology. *Nursing Times* [online]; 114: 2, 2018), pp. 34-37.

[110] Kirti, H. Sohal and S. Jain, "Interpretation of Cardio Vascular Diseases using Electrocardiogram: A Study," *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, Solan, India, 2018, pp. 159-164.

[111] A. Kumar, R. Komaragiri, & M. Kumar, From Pacemaker to Wearable: Techniques for ECG Detection Systems. *J Med Syst* **42**, 34 (2018).

[112] A. Uma, P. Kalpana. (2021). Area efficient folded undecimator based ECG detector. *Sci Rep* 11,3756.

[113] J. A. Roy, F. Koushanfar and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," *2008 Design, Automation and Test in Europe*, Munich, Germany, 2008, pp. 1069-1074.

[114] U. Satija, B. Ramkumar and M. S. Manikandan, "A New Automated Signal Quality-Aware ECG Beat Classification Method for Unsupervised ECG Diagnosis Environments," in *IEEE Sensors Journal*, vol. 19, no. 1, pp. 277-286, 1 Jan.1, 2019.

[115] U. Satija, B. Ramkumar and M. S. Manikandan, "A Review of Signal Processing Techniques for Electrocardiogram Signal Quality Assessment," in *IEEE Reviews in Biomedical Engineering*, vol. 11, pp. 36-52, 2018.

[116] National Library of Medicine, https://www.ncbi.nlm.nih.gov/books/NBK2214/#:~:text=At%20every%20beat%2C%20the%20heart,skin%20and%20displays%20it%20graphically, accessed on 5 May 2023.

[117] S. Kadambe, R. Murray and G. F. Boudreaux-Bartels, "Wavelet transform-based QRS complex detector," in *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 7, pp. 838-848, July 1999.

[118] J. N. Rodrigues, T. Olsson, L. Sornmo and V. Owall, "Digital implementation of a wavelet-based event detector for cardiac pacemakers," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 12, pp. 2686-2698, Dec. 2005.

[119] F. Zhao, X. Tang, Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction, *Pattern Recognition*, Volume 40, Issue 4, 2007, Pages 1270-1281.

[120] Voice Biometrics Market- Growth, "Trends, COVID-19 impact, and forecasts (2022- 2027)," Jun. 2022. Accessed: Aug. 2022.

[121] Voice Biometrics Market by Component, "Type, application, authen tication process, organization size, deployment mode, vertical, and region- global forecast to 2026," Rep. No. TC7070,May2021.Accessed: Aug. 2022.

[122] Mireia Farrús and Javier Hernando, "Using jitter and shimmer in speaker verification," *IET Signal Process*., vol. 3, pp. 247–257, 2009.

[123] I. Daly, M. Novotny, Z. Hajaiej and A. Garsallah, "Accuracy of jitter and shimmer measurements for speaker in the database TIMIT and NTIMIT," *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Prague, Czech Republic, 2016, pp. 1-5.

[124] Deepak Kachave, Anirban Sengupta, "Integrating Physical Level Design and High Level Synthesis for Simultaneous Multi-Cycle Transient and Multiple Transient Fault Resiliency of Application Specific Datapath Processors", *Elsevier Journal on Microelectronics Reliability*, Volume 60, Pages 141-152, May 2016.

[125] A. Nejat, Z. Kazemi, V. Beroulle, D. Hely and M. Fazeli, "Restricting Switching Activity Using Logic Locking to Improve Power Analysis-Based Trojan Detection," *2019 IEEE 4th International Verification and Security Workshop (IVSW)*, Rhodes, Greece, 2019, pp. 49-54.

[126] S. Mobaraki, A. Amirkhani and R. E. Atani, "A Novel PUF based Logic Encryption Technique to Prevent SAT Attacks and Trojan Insertion," *2018 9th International Symposium on Telecommunications (IST)*, Tehran, Iran, 2018, pp. 507-513.

[127] S. Moein, T. A. Gulliver, F. Gebali and A. Alkandari, "A New Characterization of Hardware Trojans," in *IEEE Access*, vol. 4, pp. 2721-2731, 2016.

[128] S. Hazra, J. S. Sattenapalli, A. Roy and M. Dalui, "Evaluation and Detection of Hardware Trojan for Real-Time Many-Core Systems," *2018 8th International Symposium on Embedded Computing and System Design (ISED)*, Cochin, India, 2018, pp. 31-36.

[129] M. Fyrbiak *et al*., "HAL—The Missing Piece of the Puzzle for Hardware Reverse Engineering, Trojan Detection and Insertion," in *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp. 498-510, 1 May-June 2019.

[130] A. Sengupta, "Hardware Vulnerabilities and Their Effects on CE Devices: Design for Security Against Trojans [Hardware Matters]," in *IEEE Consumer Electronics Magazine*, vol. 6, no. 3, pp. 126-133, July 2017.

[131] A. Sengupta, "Hardware Security of CE Devices [Hardware Matters]," in *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 130-133, Jan. 2017.

[132] R.V. Kshirsagar, R.M. Patrikar, Design of a novel fault-tolerant voter circuit for TMR implementation to improve reliability in digital circuits, *Microelectronics Reliability*, Volume 49, Issue 12, 2009,Pages 1573-1577.

[133] J. Liu, J. Xie, L. Chao, H. Wu, P. Ding, X. Chen, and H. Feng, ‘‘Min entropy estimation for semiconductor superlattice true random number generators,’’ *Sci. Rep.,* vol. 12, no. 1, p. 2948, Feb. 2022.

[134] NIST Computer Security Resource Center, Glossary, https://csrc.nist.gov/glossary/term/entropy#:~:text=A%20measure%20of%20the%20amount,is%20usually%20stated%20in%20bits, accessed on Feb 2022.

[135] J. Zhang, Y. Lin, Y. Lyu and G. Qu, "A PUF-FSM Binding Scheme for FPGA IP Protection and Pay-Per-Device Licensing," in *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1137-1150, June 2015.

[136] D. B. Roy, S. Bhasin, I. Nikolić, and D. Mukhopadhyay. 2019. Combining PUF with RLUTs: A Two-party Pay-per-device IP Licensing Scheme on FPGAs. *ACM Trans. Embed. Comput*. Syst. 18, 2, Article 12 (March 2019), 22 pages.