Novel Paradigms for IP Trojan Attacks, Hardware Watermarking and Design Space Exploration in High Level Synthesis

M.Tech. Thesis

By

Nitish Kumar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
MAY 2025

Novel Paradigms for IP Trojan Attacks, Hardware Watermarking and Design Space Exploration in High Level Synthesis

A THESIS

Submitted in partial fulfillment of the requirements for the award of the degree

of

Master of Technology

by

Nitish Kumar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2025

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Novel Paradigms for IP Trojan Attacks, Hardware Watermarking and Design Space Exploration in High Level Synthesis** in the partial fulfillment of the requirements for the award of the degree of **MASTER OF TECHNOLOGY** and submitted in the **DEPARTMENT 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 July 2023 to May 2025 under the supervision of Prof. Anirban Sengupta, Indian Institute of Technology Indore, India.

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
(Nitish Kumar)

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

May 18, 2025

Signature of the Supervisor of
M.Tech. thesis with date
(Prof. Anirban Sengupta)

Nitish Kumar has successfully given his M.Tech. Oral Examination held on **April 30, 2025**.

May 18, 2025

Signature of Supervisor of M.Tech. thesis
Date:

Signature of Chairman, PG Oral Board
Date:

Signature of HoD
Date:

ACKNOWLEDGEMENTS

Completing this thesis has been a challenging yet immensely rewarding journey—made possible by the support, guidance, and encouragement of many individuals. I take this opportunity to express my heartfelt gratitude to all who contributed.

First and foremost, I extend my deepest gratitude to my advisor, **Prof. Anirban Sengupta**, for his exceptional mentorship, insightful feedback, and unwavering support. His guidance has been instrumental in shaping the direction and depth of my research and my personal growth.

I sincerely thank my lab seniors—**Mr. Aditya Anshul, Mr. Vishal Chaurasia, and Mr. Nabendu Bhui**—for their consistent support, technical guidance, and encouragement throughout my work.

I am grateful to my **thesis committee members** for their valuable feedback and suggestions, which greatly improved the quality and scope of this thesis. I also acknowledge the **faculty and staff of the CSE Department at IIT Indore** for providing a positive and intellectually enriching environment. Their support and infrastructure played a vital role in facilitating my research.

My heartfelt thanks to my **family and friends** for their love, patience, and constant encouragement. Special thanks to my **parents and siblings** for always standing by me and motivating me through challenges. I am thankful to my **batchmates** for their camaraderie, support, and the many shared experiences that made this journey more meaningful.

Finally, I extend my gratitude to all my **peers and colleagues** whose academic input and moral support, in big or small ways, helped me reach this milestone.

This thesis is a reflection of your collective support. Thank you all for being part of this journey.

ABSTRACT

This thesis proposes three novel and practical methodologies aimed at strengthening the **hardware security** of intellectual property (IP) core designs, especially those generated using **high-level synthesis (HLS)**. As hardware attacks such as IP piracy have become increasingly sophisticated, these contributions offer effective solutions to detect and prevent such threats, ensuring trustworthy integration of IPs into modern **consumer electronics (CE)** and **system-on-chip (SoC)** designs.

The **first contribution** introduces a **malevolent HLS (M-HLS) framework**, which demonstrates how malicious entities can insert **hardware Trojans** during the HLS design flow. These Trojans—specifically, **Performance Degradation Hardware Trojans (PDHTs)** and **Denial-of-Service Hardware Trojans (DoS-HTs)**—can silently disrupt the normal behavior of IP cores. This work focuses on inserting such Trojans at the **interconnect (mux) stage** of HLS-generated **watermarked IPs**, showing that even well-protected designs can be vulnerable. Experimental results on a benchmark IP (MESA Horner Bezier's) reveal that attackers can cause significant performance drops or even system failure with only minor area and power overhead, raising serious concerns for hardware designers.

The **second contribution** presents a **hybrid Genetic Algorithm–Particle Swarm Optimization (GA-PSO) framework** that simultaneously explores two design goals: embedding a low cost **biometric watermark** (based on **palmprint data**) and selecting an optimal **loop unrolling factor** for performance optimization. This HLS-based design space exploration ensures that the watermark is robust against tampering and difficult to forge, helping IP vendors prove ownership and protect against unauthorized reuse. The hybrid algorithm intelligently combines the exploration strengths of GA and PSO to avoid local optima, and the results show superior **tamper tolerance**

and lower **watermark collision probability**, all while keeping the **design cost negligible**.

The **third contribution** focuses on protecting **convolutional neural network (CNN) IP design**, widely used in AI-based CE systems. This method secures the IP by using HLS framework by introducing a **4-variable encoded signature** and applying **loop unrolling** for structural obfuscation. This not only improves the design's resistance to attacks but also reduces the risk of unauthorized duplication or reverse engineering. The approach maintains a **low hardware overhead** and outperforms prior techniques in terms of **tamper resistance** and **ownership traceability**.

Together, these three methodologies tackle different yet critical aspects of IP security—from malicious hardware insertion to secure watermarking and and protection of convolutional IP cores commonly used in deep learning accelerators and AI-enabled consumer electronic systems. They demonstrate that **effective hardware security** can be achieved through smart integration of security features into the **HLS design flow**, without compromising design quality or efficiency. This work contributes to the growing field of **secure hardware design automation**, offering valuable tools and frameworks for the development of **reliable and secure IPs** for the next generation of electronic systems.

LIST OF PUBLICATIONS

JOURNAL:

- A. Sengupta, A. Anshul, V. Chourasia and N. Kumar, "M-HLS: Malevolent High-Level Synthesis for Watermarked Hardware IPs," in **IEEE Embedded Systems Letters**, vol. 16, no. 4, pp. 497-500, Dec. 2024, doi: 10.1109/LES.2024.3416422.

CONFERENCES:

- A. Sengupta, V. Chourasia, A. Anshul and N. Kumar, "Robust Watermarking of Loop Unrolled Convolution Layer IP Design for CNN using 4-variable Encoded Register Allocation," **2024 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), Taichung, Taiwan, 2024**, pp. 589-590, doi: 10.1109/ICCE-Taiwan62264.2024.10674385.
- Sengupta, V. Chourasia and N. Kumar, "HLS driven Hybrid GA-PSO for Design Space Exploration of Optimal Palmprint Biometric based IP Watermark and Loop Unrolling Factor," **2024 IEEE International Symposium on Smart Electronic Systems (iSES)**, New Delhi, India, 2024, pp. 134-139, doi: 10.1109/iSES63344.2024.00036.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

1. Introduction

1.1 Evolution of IC Design and the Rise of IP-Based Methodology

Semiconductors are the foundation of modern digital technology, powering a wide range of devices across consumer electronics (CE), telecommunications, automotive systems, and more. The continued scaling and miniaturization of semiconductor devices have enabled the rise of compact, high-performance products such as smartphones, tablets, digital cameras, and smart appliances [1].

At the core of these technologies lies the design and fabrication of integrated circuits (ICs), which have become increasingly complex over time. Initially, ICs were designed manually using discrete components and transistor-level schematics, which were time-consuming, labour-intensive, and error-prone. The advent of Electronic Design Automation (EDA) tools in the early 1980s transformed this landscape. Tools such as Cadence Virtuoso automated key design tasks, enabling abstraction across multiple levels—transistor/layout, register-transfer level (RTL), and system/algorithmic levels [2], [3]. These abstraction levels formalized the IC design process and significantly boosted productivity.

By the mid-1990s, RTL-based design flows using Hardware Description Languages (HDLs) like Verilog and VHDL became industry-standard [4]. Synthesis tools such as Synopsys Design Vision, Xilinx ISE, and Altera Quartus facilitated the efficient conversion of RTL into gate-level implementations, striking a balance between design abstraction and hardware control [5]. These flows enabled faster prototyping, modular development, and improved time-to-market—crucial for CE industries [40] under constant

innovation pressure [6]. The IC design supply chain, shown in Fig. 1, involves multiple stages including RTL design, synthesis, fabrication, testing, and packaging. With contributions from third-party IP vendors and global foundries, this distributed process is susceptible to threats such as IP piracy and hardware Trojans [40].

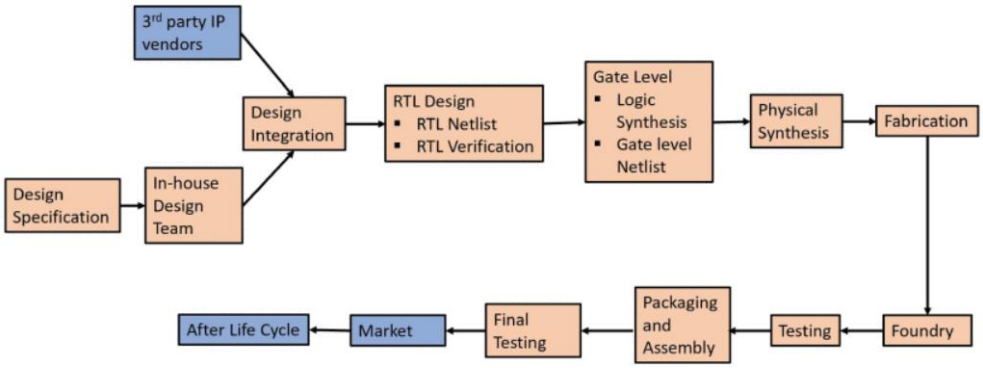

Fig. 1. Integrated circuit (IC) design supply chain [40].

1.2 Emergence of IP Cores and High-Level Synthesis (HLS)

To manage increasing design complexity, the industry adopted Intellectual Property (IP) cores—pre-verified and reusable hardware modules that could be easily integrated into larger systems. IP cores provided modularity and significantly reduced both development and verification time. Standard IPs for communication protocols, memory controllers, and DSP blocks are now readily available in vendor libraries, supporting rapid prototyping and design reuse across projects [7].

As systems-on-chip (SoCs) grew in scale and complexity, High-Level Synthesis (HLS) emerged as a powerful methodology to further enhance productivity. HLS tools accept high-level functional descriptions—usually in C or C++—and automatically generate synthesizable RTL code [8]. This shift introduced several key advantages:

1. **Simplified Design Entry:** Designers with limited HDL expertise could model hardware in high-level languages.
2. **Automated RTL Generation:** Reduced human effort and minimized manual coding errors.
3. **Accelerated Design Iterations:** Enabled faster exploration of architectural trade-offs.

In data-path-intensive applications such as DSP and multimedia processing, HLS workflows have proven especially beneficial. Furthermore, IP cores can be reused even at behavioral abstraction levels within HLS environments, enhancing design modularity and scalability [9] [40].

Despite these benefits, HLS still faces challenges in handling control-intensive applications due to limitations in modelling complex conditional behaviors. As a result, many mission-critical and control-dominant systems still rely on RTL-based design flows [10].

1.3 Security Challenges in IP-Based Design

The globalization of the semiconductor supply chain has introduced substantial security concerns, particularly in the context of outsourced design, verification, and fabrication. The increased use and reuse of third-party IPs, often in synthesizable formats, has exposed hardware systems to serious threats such as **IP piracy** and **hardware Trojan insertion [40]**.

To safeguard intellectual property in IC designs, various security mechanisms have been developed. Among these, **watermarking**, **fingerprinting**, and **IP metering** have gained significant traction.

IP Piracy and Theft

IP cores represent substantial investments in design effort and research, making them attractive targets for unauthorized use. IP piracy [40]

encompasses actions like unauthorized duplication, reverse engineering, and redistribution of proprietary designs. The risks are exacerbated by the widespread use of synthesizable RTL IPs, which are easily copied or modified [11].

To counteract such threats, the following protection techniques are commonly employed:

- **Watermarking:** Embeds a vendor-specific signature into the IP design to assert ownership.
- **Fingerprinting:** Inserts a unique buyer-specific ID, useful for tracing the source in case of leakage.

These techniques are designed to be **stealthy**, **resilient to tampering**, and **minimally intrusive** in terms of performance and area overhead. Depending on the use case, they can be embedded at different abstraction levels—behavioral, RTL, or gate-level—with growing emphasis on early integration during **HLS** for better propagation and traceability [12].

Other protection mechanisms include:

- **IP Metering:** Embeds a unique ID into each IP or chip instance, allowing vendors to monitor and control usage based on licensing agreements.
- **Computational Forensic Engineering (CFE):** Analyzes structural and statistical patterns in hardware designs to infer authorship and synthesis tools used [13].

Legal tools such as **patents** and **copyrights** further enhance IP protection by providing statutory rights to owners, although they do not address internal misuse or reverse engineering.

Hardware Trojan Attacks

A **hardware Trojan** is a malicious modification intentionally inserted into a hardware design, typically during third-party integration, logic synthesis, or fabrication stages. These Trojans are often stealthy—triggered under rare or exceptional conditions—and can cause system failure, data leakage, or altered functionality.

Key challenges in detecting hardware Trojans include:

- Their insertion during **untrusted or outsourced stages**.
- **Rare triggering conditions** that escape conventional testing.
- **Invisibility at RTL or behavioral levels**, especially in black-box IPs [14].

Countermeasures include **side-channel analysis**, **formal verification**, and **security-aware HLS flows** [15]. In some cases, **design partitioning** across trusted vendors can limit the exposure of critical subsystems to untrusted environments.

1.4 Watermarking as a Defensive Technique

Among the arsenal of IP protection techniques, **digital watermarking** stands out as an effective and robust strategy. It embeds identifiable information—such as the IP vendor's identity or a unique signature—into the design in a **non-intrusive and resilient** manner. When applied early in the design flow (e.g., during high-level or architectural synthesis), watermarks can persist through synthesis and back-end processes, allowing post-deployment ownership verification [12].

A typical watermarking methodology includes the following stages:

- **Operation Scheduling:** Schedule operations under timing and resource constraints in a manner that facilitates watermark embedding.

- **Hardware Allocation and Binding:** Assign operations to functional units while embedding signature patterns.
- **Register Allocation:** Use **coloured interval graphs** to insert watermark constraints—by introducing specific edges that influence register binding [16].
- **Signature Embedding:** Encode characters (e.g., ‘i’, ‘I’, ‘T’, ‘!’) through node pairing and multivariable encodings to insert unique identifiers.
- **Multiplexing Scheme Generation:** Modify data path and control logic to incorporate the embedded watermark [12].

This methodology ensures that the embedded watermark does not affect functional correctness while enabling reliable post-deployment validation. Carefully crafted watermarks are **stealthy**, **difficult to remove**, and **robust against reverse engineering**, making them an essential tool in modern hardware security strategies [11], [12].

1.5 Safeguarding Intellectual Property: A Survey of IP Protection Techniques

Securing intellectual property (IP) cores from threats like piracy and unauthorized use is essential in today’s semiconductor industry. Several techniques exist to guard IP across different levels of hardware design abstraction. Employing a combination of these methods can form a strong, layered defense strategy. Understanding these techniques is particularly important for design firms to protect their proprietary technologies. Common IP protection approaches include watermarking**,** IP metering**,** Computational Forensic Engineering (CFE)**,** and legal safeguards like patents and copyrights [12], [15].

Watermarking

Watermarking has emerged as a widely adopted technique to protect IP cores. This method integrates a unique signature into the IP during the **architectural**

synthesis stage, typically at the **register allocation** or **scheduling** step. A **colored interval graph** is used to map storage variables and their lifespans. By introducing extra edges into this graph, the watermarking constraints force certain variables to occupy distinct registers, embedding the watermark securely [16].

Verifying the watermark requires two steps: **reverse engineering** the suspicious product and matching the extracted watermark with the original signature. This approach offers a robust method to identify IP ownership and deter piracy [12].

IP Metering

IP metering provides a way for vendors to **monitor and regulate usage** of their IP cores, ensuring fair compensation and preventing misuse. Each IP core instance is assigned a **unique identifier** during synthesis or via configurable hardware elements. This identifier tracks how often the IP is used, supporting enforcement of licensing terms. **Hardware metering** is often employed when direct manufacturing oversight isn't feasible, especially in untrusted foundry environments [10]. **Software metering**, on the other hand, manages usage via digital licenses, especially for soft IPs.

These mechanisms uphold the vendor’s rights and support continued innovation by ensuring return on IP investment [12].

Computational Forensic Engineering (CFE)

CFE is an **investigative technique** that examines the **design features and statistics** of an IP to trace its origin. It extracts unique attributes of the IP and compares them with known patterns—such as tool-specific naming conventions or synthesis characteristics—to identify the possible creator or toolchain. This can act as both an authentication method and a deterrent against reverse engineering [11] [40].

Patents and Copyrights

Legal tools like patents and copyrights offer formal protection. A **patent** grants the creator exclusive rights to manufacture and commercialize an invention for a specified time, after verifying its novelty and utility. **Copyrights**, in contrast, protect **original creative expressions**, including RTL descriptions and software, by granting the author control over usage and distribution.

While legal safeguards can deter IP theft through litigation, they are often **reactive**, not preventive, and may not deter **internal misuse** or reverse engineering—making technical techniques like watermarking indispensable [12], [15].

1.6 Overview of High-Level Synthesis

High-Level Synthesis (HLS) [38] is the process of converting a high-level behavioral specification—typically written in languages such as C, C++, or MATLAB—into a **Register Transfer Level (RTL)** description suitable for hardware implementation. HLS enables designers to operate at a higher abstraction level, significantly reducing development time and enhancing productivity, particularly for complex digital systems. It plays a central role in modern VLSI design, supporting rapid design space exploration and automated performance optimization in terms of area**,** timing, and power [17].

Design Entry Phase

The HLS process begins with the specification of system functionality and constraints using a high-level language. From this specification, a **Data Flow Graph (DFG)** is derived to capture the computational operations and their data dependencies. A corresponding **Control Flow Graph (CFG)** models the sequencing of operations. These are merged to form a Control/Data Flow

Graph (CDFG), which serves as the foundation for subsequent hardware optimization and synthesis [10], [18].

High-Level Design Phase

In this phase, the CDFG is systematically transformed into an optimized RTL design through three core tasks:

1. **Scheduling**: Determines the execution order of operations to minimize latency and satisfy timing constraints.
2. **Resource Allocation**: Assigns operations to hardware resources such as ALUs, multipliers, registers, and buses.
3. **Binding**: Maps each scheduled operation and data transfer to specific physical hardware resources to optimize utilization and reduce critical path delay.

To enhance system performance and efficiency, the following **optimization techniques** are commonly applied:

- **Design Space Exploration (DSE)**: Evaluates alternative hardware implementations by varying architectural parameters and mapping decisions.
- **Loop Unrolling and Pipelining**: Increases parallelism and throughput by replicating loop bodies or overlapping operations across cycles.
- **Data Path Optimization**: Refines the structure of functional units and interconnects to reduce latency and area.
- **Control Path Optimization**: Streamlines the generation of control signals to improve timing and reduce complexity.
- **Power Optimization**: Minimizes dynamic and static power through techniques such as operand isolation and switching activity reduction [16], [17].

RTL Generation Phase

The final stage in HLS is the generation of a **synthesizable RTL** description. This involves:

1. **Datapath and Control Path Synthesis**: Realizes the physical data and control logic from the optimized CDFG.
2. **RTL Netlist Generation**: Produces a hardware netlist comprising standard cells and their interconnections.
3. **Verification**: Validates the RTL design against functional and timing specifications using simulation, equivalence checking, and constraint-driven synthesis.
4. **Optimization**: Applies additional refinements such as **clock gating**, **logic restructuring**, and **retiming** to improve energy efficiency and timing closure.

A well-optimized RTL output from HLS not only meets the desired functional and non-functional requirements but also ensures **smooth integration** into the downstream phases of the digital design flow, including place and route, sign-off, and tape-out [18].

1.7 Organization of the Thesis

This thesis is organized into eight chapters, detailing a series of research contributions aimed at enhancing the security and protection of hardware IP cores within high-level synthesis (HLS) design flows. Chapter 2 presents a comprehensive review of prior work in watermarking, steganography, authentication, and fault-tolerant hardware security, identifying existing gaps and motivating the need for more resilient approaches. Chapter 3 introduces the Malevolent High-Level Synthesis (M-HLS) framework, which demonstrates how attackers can exploit post-watermarking vulnerabilities in mux-based interconnect designs to insert stealthy hardware Trojans, such as performance degradation and denial-of-service Trojans. Chapter 4 proposes a

robust watermarking technique based on four-variable encoded register allocation, applied to loop-unrolled convolution layer IP designs for CNNs, with a focus on generating tamper-resistant signatures through systematic encoding rules. Chapter 5 presents a hybrid genetic algorithm and particle swarm optimization (GA-PSO) framework for design space exploration that simultaneously optimizes palmprint-based biometric watermarking and loop unrolling factors, enhancing security while maintaining low design overhead. Chapter 6 analyzes the results of all proposed approaches, demonstrating improvements in tamper tolerance, low coincidence probability, performance degradation under attack, and resistance to known detection techniques. Chapter 7 concludes the thesis by summarizing the contributions and emphasizing their role in advancing secure IP core design methodologies. Finally, Chapter 8 outlines potential future directions, including real-time threat response, multi-modal biometric integration, and broader applicability across heterogeneous SoC platforms.

Chapter 2

2. Literature Review

2.1 Review of Past Work and Problem Formulation

The field of hardware IP watermarking has become a cornerstone of IP protection, particularly for safeguarding data-intensive coprocessors commonly used in the consumer electronics industry. However, the effectiveness of earlier watermarking techniques varies depending on the specific methodology used and the resilience of the embedded watermark against various forms of attacks. As such, it is essential to critically examine existing approaches, understand their advantages and limitations, and identify potential security vulnerabilities to advance more robust IP protection mechanisms. This chapter presents a comprehensive review of prior methods and highlights the need for further innovation in hardware IP watermarking and related protection strategies.

2.2 Related Work on Trojan Insertion and detection in HLS-Based IPs

Recent advancements in high-level synthesis (HLS) tools have enabled efficient generation of hardware intellectual property (IP) cores, including the integration of watermarking to protect against IP theft. However, several studies have identified that compromised HLS tools themselves can become a threat vector for stealthy hardware Trojan insertion.

Pilato et al. [19] demonstrated that hardware Trojans could be inserted directly during HLS by modifying scheduling, allocation, and datapath synthesis stages. They introduced degradation, battery exhaustion, and downgrade attacks through a malicious HLS tool. However, their work focused only on

non-watermarked designs. A follow-up study [20] proposed a detection method using C-to-RTL equivalence checking to identify such Trojans, though it assumes access to both clean and infected designs, and it is not practical for large watermark-protected systems.

To enhance Trojan resilience, a DMR-based HLS technique was proposed in [24], which uses particle swarm optimization (PSO) to find low-cost, redundant schedules. While effective against output-modifying Trojans, it introduces hardware overhead and does not address vulnerabilities introduced by watermarking.

Watermarking approaches like those in [21], [22], and [23] embed multi-variable constraints during HLS stages such as register allocation and scheduling. These techniques protect IP ownership by encoding signatures in design behavior and structure. However, they may unintentionally introduce free input ports in the mux-based interconnect, creating an opportunity for Trojan insertion—an aspect not considered in prior work.

The proposed Malevolent HLS (M-HLS) framework builds on this gap by exploiting mux-level vulnerabilities created during watermarking to insert stealthy Trojans—namely, performance degradation (PD-HT) and denial-of-service (DoS-HT)—without affecting normal functionality. These Trojans are triggered under rare conditions and evade current detection methods like path-delay analysis [21], FSMD comparison [20], or GNN-based RTL verification [25].

Thus, while prior works offer valuable insights into Trojan detection and watermarking, none address the security risks arising from their interaction in the HLS flow. The M-HLS framework addresses this unexplored vulnerability by demonstrating how watermarking unintentionally opens the door for undetectable hardware Trojans in synthesized IP cores.

2.3 Prior Work on Watermarking for Hardware IPs

Several prior works have explored watermarking techniques for securing hardware IPs. In [26], a physical-level watermarking approach is introduced using an 8-bit signature embedded directly into the layout. In [27], digital signature embedding using secure hash algorithms and RSA encryption is proposed to ensure integrity and ownership of DSP cores. Similarly, [28] leverages facial biometric features of the IP vendor to generate a watermark, providing a biometric-based security framework. However, these techniques primarily target DSP or sequential circuits and do not offer a structured methodology for watermarking CNN hardware, particularly with loop unrolling.

Prior research has explored various techniques to secure hardware IPs against piracy, reverse engineering, and counterfeiting, primarily focusing on embedding secret signatures or biometric features for watermarking and authentication. However, these methods often face limitations in security strength, design overhead, and resistance to brute-force attacks.

Sengupta and Rathor [29] proposed a dual-layer defense combining multi-key structural obfuscation with tamper-tolerant watermarking for DSP IPs, offering strong tamper resistance but lacking automated security optimization. Sengupta et al. [30] introduced a hybrid watermarking approach at both architectural and RTL levels, though it lacked dynamic adaptability for optimal watermark cost or configuration. Sengupta and Chaurasia [28] secured CNN convolutional layers with facial biometrics, enhancing tamper resistance but failing to optimize design cost across varying inputs and loop configurations. Rathor and Sengupta [32] proposed a scheduling-driven watermarking method in HLS tools, but it lacked dynamic design space exploration. Koushanfar et al. [33] introduced dynamic watermarking with

timing and design constraints but did not incorporate biometric encoding or multi-objective optimization.

Unlike these approaches, the proposed method presents a robust framework for embedding multivariable digital signatures into loop unrolled CNN convolution layer IPs. By doing so, it provides stronger protection against IP piracy and facilitates seamless verification of ownership. Furthermore, it integrates the watermarking process during high-level synthesis, making it well-suited for scalable and high-performance CNN hardware implementations. The proposed HLS-driven hybrid GA-PSO framework combines Genetic Algorithms and Particle Swarm Optimization to optimize both loop unrolling for faster CNN computation and biometric-based watermarking for robust IP protection, balancing performance, security, and minimizing design cost for modern consumer electronics systems.

2.4 Summary

This chapter reviewed prior work in three key areas of hardware IP protection. The discussion on Trojan insertion in HLS-based watermarked IPs highlighted vulnerabilities in existing watermarking methods, particularly their susceptibility to hardware Trojans introduced during the HLS process. The proposed Malevolent HLS (M-HLS) framework addresses this gap by demonstrating how watermarking unintentionally creates opportunities for undetectable Trojans.

In the area of watermarking CNN-based IPs, previous techniques mainly targeted DSP or sequential circuits and did not consider the complexities of CNN hardware, particularly with loop unrolling. The proposed method improves upon this by embedding multivariable digital signatures into loop-unrolled CNN convolution layers, providing stronger protection and easier ownership verification.

Finally, the chapter explored the general limitations of hardware IP security techniques, including their vulnerability to brute-force attacks and high design overhead. The proposed HLS-driven hybrid GA-PSO framework offers a solution by simultaneously optimizing loop unrolling and biometric-based watermarking, striking a balance between performance, security, and design cost.

These discussions set the stage for the next chapters, which will detail the proposed methodology and evaluate its effectiveness in addressing these challenges.

Chapter 3

3. Malevolent HLS Framework for Trojan Insertion in Watermarked IP Designs

3.1 Introduction

High-Level Synthesis (HLS) has become a widely adopted approach for generating reusable hardware Intellectual Property (IP) cores used in various electronic and multimedia systems. While HLS enables faster and more efficient design flows, it also introduces new security vulnerabilities. One of the most critical threats is the potential insertion of hardware Trojans—malicious logic circuits embedded within an IP core that remain inactive until triggered under specific conditions.

Recent research has demonstrated that a compromised or untrusted HLS toolchain can serve as a vehicle for Trojan insertion [39] during key stages such as scheduling, resource allocation, binding, and interconnect design [19], [20], [34]. These threats are particularly concerning in the context of watermarked IP designs, where the presence of security features might paradoxically attract sophisticated adversaries seeking to compromise the design [22], [24].

This chapter introduces a novel Malevolent HLS (M-HLS) framework that showcases the feasibility of inserting two distinct types of hardware Trojans—Performance Degradation Hardware Trojan (PDHT) and Denial-of-Service Hardware Trojan (DoS-HT)—specifically within the multiplexer (mux)-based interconnect stage of an HLS-generated, watermarked IP core. The proposed M-HLS framework is validated on a watermarked MESA Horner Bezier's IP core [34], demonstrating that attackers can achieve significant performance degradation or complete denial of service, while incurring only minimal area and power overhead. This analysis highlights the urgent need

for security-aware HLS methodologies capable of detecting and mitigating such sophisticated threats.

3.2 Motivation and Threat Model

A compromised HLS toolchain, particularly one provided by an untrusted third-party vendor, can act as a vector for malicious insertion during design stages such as scheduling, allocation, binding, and interconnect generation [19], [24]. The core motivation behind this study is to analyse how a malicious entity can embed Trojans during the mux-based interconnect stage of the HLS process, thus compromising even protected, watermarked IP cores.

Two specific types of Trojans are considered in this context:

- **Performance Degradation Hardware Trojan (PDHT):** Degrades execution speed and energy efficiency by forcing unnecessary data paths or logic switching [34].
- **Denial of Service Hardware Trojan (DoS-HT):** Causes partial or complete loss of functionality under specific trigger conditions [34].

These threats are particularly dangerous when introduced after watermark embedding, as they may evade traditional verification techniques while still impairing the system [25].

3.3 Watermarked IP Design and Trojan Vulnerability

Watermarking in hardware IP design serves as a prominent countermeasure against IP piracy and unauthorized claims of ownership. In the context of HLS-generated IP cores, watermarking is implemented by embedding secret security constraints during the architectural synthesis stage, particularly in the register allocation phase. These constraints are often expressed through modifications in register assignments and interconnects within the Scheduled

Data Flow Graph (SDFG). The altered resource mapping ensures that only an authorized designer or entity with knowledge of the encoding rules can assert ownership or verify authenticity. Notable techniques in this domain, such as those proposed in [21], [22], and [23], leverage register colouring and allocation algorithms to embed multi-variable signatures without affecting the design's functional correctness.

However, while such watermarking techniques effectively support IP verification, they can inadvertently introduce vulnerabilities that compromise the security of the design. One such vulnerability arises from the side effects of watermark embedding on the interconnect network, particularly the multiplexer (mux)-based connections used in data routing. During the watermark embedding phase, additional constraints may necessitate the reallocation of registers and data paths. This reconfiguration can lead to structural changes in the mux network, sometimes resulting in underutilized or unused input ports within the datapath interconnect.

To illustrate this phenomenon, consider the MESA Horner Bezier's DSP benchmark SDFG shown in Figures 2 and 3. Figure 2 represents the original unwatermarked SDFG, while Figure 3 shows the SDFG after watermark embedding using the method from [22]. In the watermarked version, the register allocation has been altered to satisfy security constraints. The horizontal colored lines in the figures denote intermediate registers for specific storage variables, which are redistributed post-watermarking.

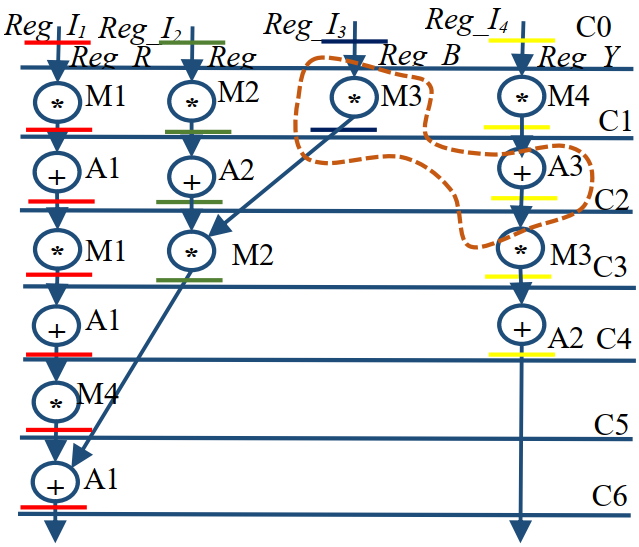

Fig. 2. SDFG of MESA Horner Bezier without watermark.

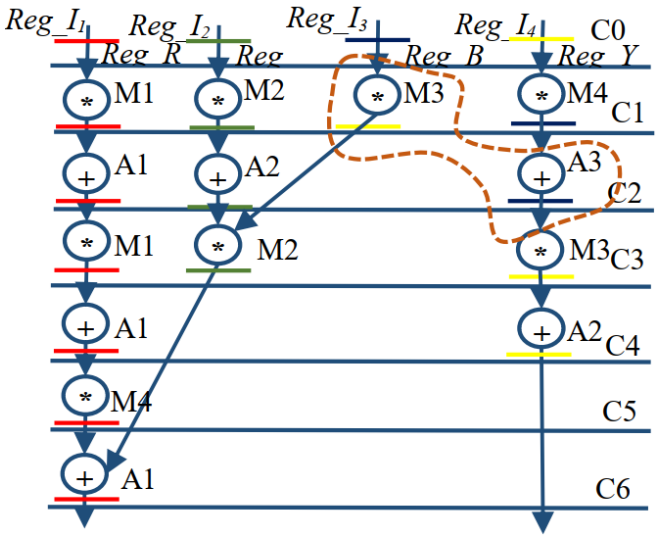

Fig. 3. SDFG of MESA Horner Bezier with embedded watermark.

Further analysis of the mux-based datapath for one of the registers—Reg_B—demonstrates the impact of watermarking. Figures 4 and 5 show the corresponding interconnect designs before and after watermark insertion. In the unwatermarked design (Figure 4), the 2×1 mux connected to Reg_B utilizes both input ports effectively. However, in the watermarked version (Figure 5), the reallocation of operations and register usage leaves one input

port of the same mux unconnected. This unused or 'free' input pin, highlighted in blue, represents a latent security flaw.

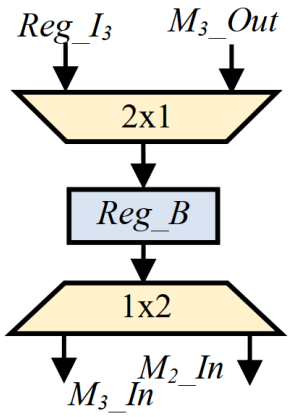

Fig. 4. Mux-based interconnect design of datapath without watermark in register (Reg_B) corresponding to Fig. 2.

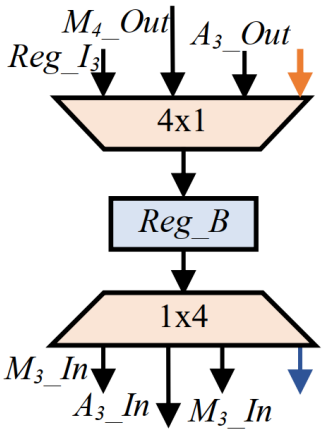

Fig. 5. Mux-based interconnect design of datapath post watermarking in Reg_B corresponding to Fig. 3.

This unoccupied mux port creates a stealthy insertion point for hardware Trojans, offering adversaries an opportunity to insert malicious logic without violating existing functional or timing constraints. Because the mux remains logically connected to a valid register but is partially unused, the attacker can exploit this idle port to introduce malicious data or trigger signals. The

presence of such structural irregularities, unnoticed during conventional verification and validation procedures, presents a subtle but significant hardware threat.

Therefore, while HLS-based watermarking enhances IP traceability and legal protection, it may inadvertently facilitate Trojan insertion when not coupled with robust structural validation. These vulnerabilities emphasize the need for a holistic security approach that goes beyond ownership verification to address the integrity and trustworthiness of the entire synthesis and interconnect design pipeline. Identifying such weaknesses early in the HLS process is crucial to developing tamper-resistant IP cores, especially in sensitive consumer electronics and critical systems.

The upcoming sections of this chapter will explore specific malevolent modifications that can be introduced at the interconnect level and how these vulnerabilities can be exploited to construct sophisticated hardware Trojans, such as performance degradation and denial-of-service (DoS) variants.

3.4 The Proposed M-HLS Framework

The proposed Malevolent High-Level Synthesis (M-HLS) framework is designed to exploit specific vulnerabilities introduced during the HLS-based watermarking process in hardware IPs. As depicted in Figure 6, this framework demonstrates how an adversary—especially one with access to or control over an untrusted HLS toolchain—can secretly insert hardware Trojans into the final synthesized design by leveraging unused interconnect resources.

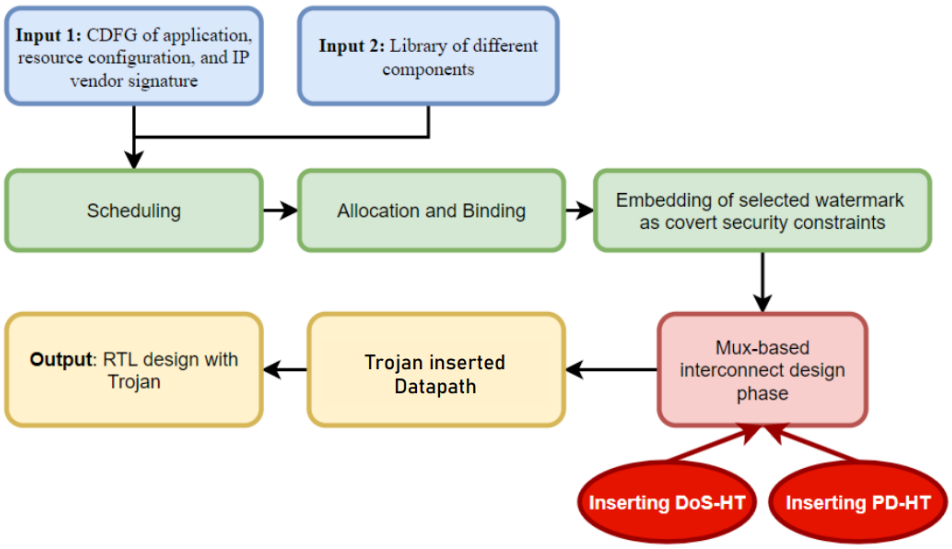

Fig. 6. Proposed M-HLS framework.

This framework introduces two distinct types of malicious logic insertions:

1. **Performance Degradation Hardware Trojan (PD-HT):** PD-HTs are crafted to subtly degrade the operational performance of an IP. These Trojans may introduce additional delays, disrupt optimal datapath flow, or affect resource usage under specific triggering conditions. The objective is not to halt the system but to deteriorate its performance in a way that can evade conventional validation and testing processes.
2. **Denial-of-Service Hardware Trojan (DoS-HT):** DoS-HTs are designed to cause system-level failure or lock-up. Upon activation, they introduce open circuit behavior, high-impedance states, or unpredictable logic that effectively disables part or all of the system's functionality. These effects are typically induced by forcing a critical path into an undefined or floating state during active operation.

Both types of Trojans remain inactive during normal operation and are triggered only when a specific, attacker-defined condition is met. This trigger is implemented using comparator-based logic, which continuously monitors

internal signals and activates the Trojan payload only when the exact triggering pattern or value is detected. Such stealthy behavior makes detection during functional simulation or routine testing extremely difficult.

Although the demonstration in this work uses the MESA Horner Bezier IP as a case study, the M-HLS framework is scalable and applicable to a wide range of HLS-generated IPs. Examples include FIR filters, JPEG codecs, DCT/IDCT processors, and other data-path-heavy designs. The generality of this approach makes it a credible and serious threat model in modern IP-based design workflows.

3.5 Insertion of Proposed Trojans in the Mux-Based Interconnect Design Stage

This section outlines how the proposed Trojans—Performance Degradation Hardware Trojan (PD-HT) and Denial-of-Service Hardware Trojan (DoS-HT)—can be embedded into HLS-generated watermarked IP designs by exploiting vulnerabilities introduced during multiplexer (mux)-based interconnect design. These vulnerabilities often surface post-watermarking, where unused input ports in muxes become available in the datapath of certain registers, such as Reg_B in the watermarked MESA IP.

The **M-HLS framework** supports the insertion of one Trojan at a time, allowing either PD-HT or DoS-HT to be embedded based on the adversary's objective. These Trojans can be incorporated into any HLS-based watermarked design, assuming an unutilized mux input is present—this is a frequent byproduct of watermark embedding during register allocation.

PD-HT Insertion

Figure 7 demonstrates PD-HT insertion in the mux structure of Reg_B. This Trojan leverages a dormant logic circuit that becomes active when a

comparator detects that a system variable i matches a preloaded attacker-defined constant c. Under this condition, the mux selects a delayed path (comprising an even number of inverters) to route the signal, thereby degrading performance by increasing propagation delay. Figure 7 illustrates the behavioral operation of the mux under normal and triggered conditions. While the functional correctness remains unaffected, the latency of computation increases, which can reduce overall system throughput or violate timing constraints.

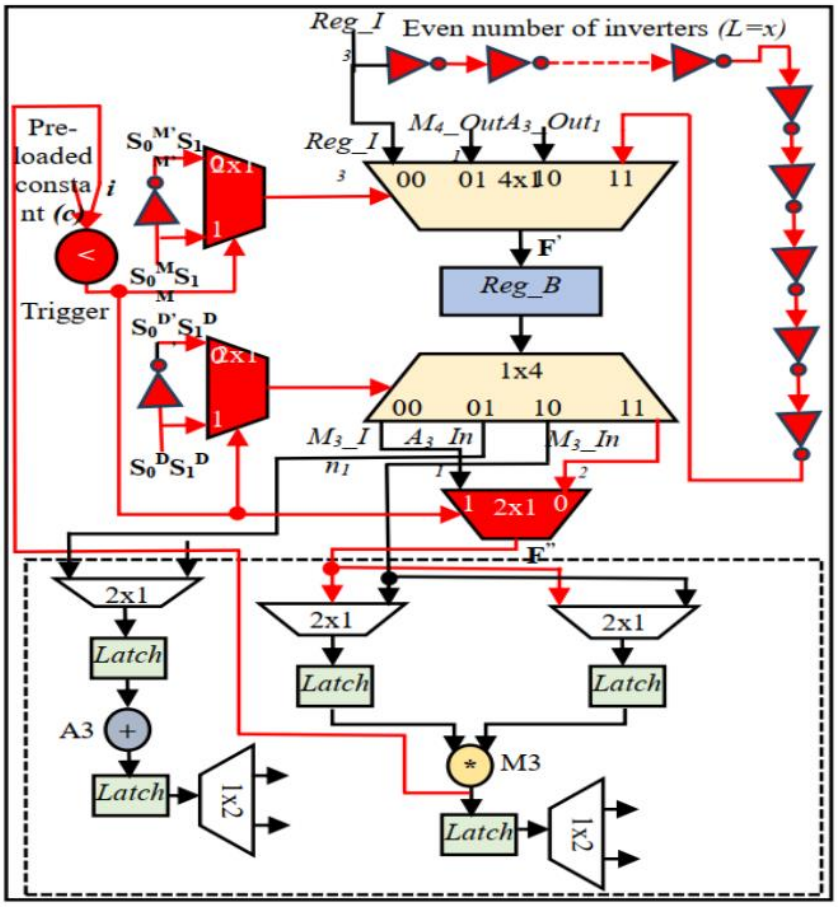

Fig. 7. Partial datapath of a watermarked MESA IP depicting insertion of PD-HT in the multiplexer of register (Reg_B).

Table 1. Behavioural Table (Output) Corresponding to Fig. 7

Trigger	Comparator o/p (Q)	S_0^M	S_1^M	F’	O/P
i=c (Trojan active)	0	1	1	$M_3\_In_1$	Delayed O/P
i≠c (Trojan inactive)	1	0	0	$M_3\_In_1$	Normal
	1	0	1	$A_3\_In_1$	
	1	1	0	$M_3\_In_2$	

DoS-HT Insertion:

Similarly, Figure 8 illustrates the implementation of the DoS-HT in the same datapath. Once triggered by the same comparator mechanism (i.e., when i = c), the mux output (F″) switches to a high-impedance or undefined state (Z). This effectively induces a denial-of-service condition at the output, rendering the downstream logic non-operational. Figure 8 presents the behavior of the mux under both normal and Trojan-activated states. The Trojan remains inactive under normal conditions, ensuring stealth and evasion from typical simulation or test-based verification processes.

The triggering value c is hardcoded into a programmable memory block during the Trojan insertion phase, making it difficult to detect without prior knowledge of the encoding strategy. Both designs were implemented and validated within a publicly available watermarking-enabled HLS toolchain, ensuring their applicability to real-world hardware IPs.

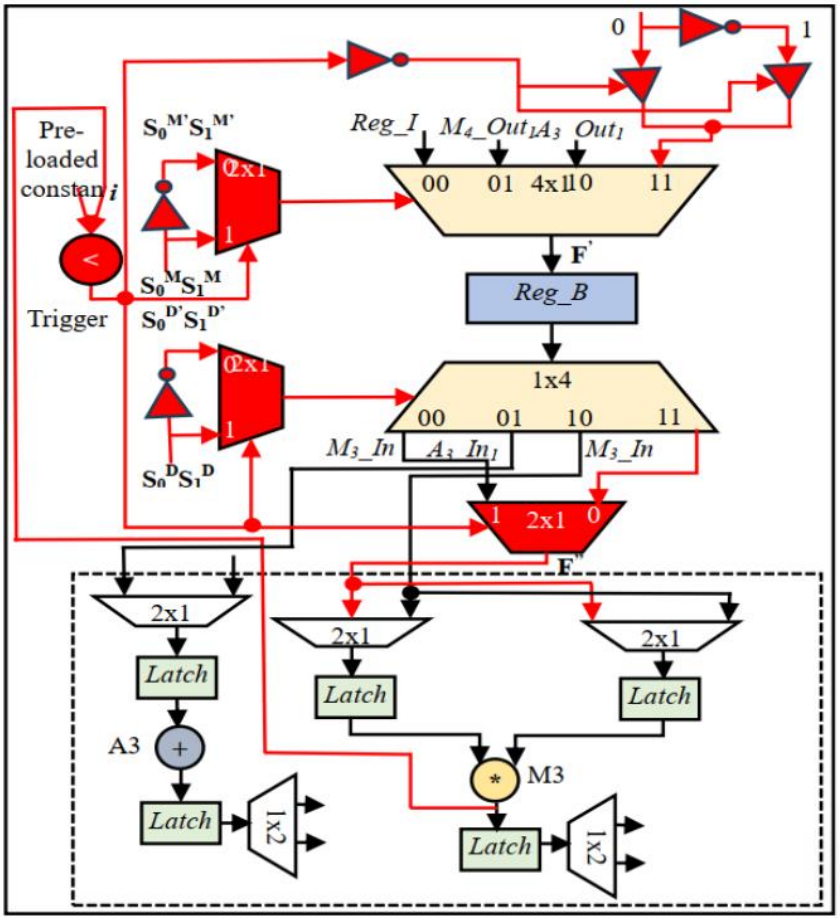

Fig. 8 Partial datapath of a watermarked MESA IP depicting insertion of DoS-HT in multiplexer register (Reg_B).

Table 2. Behavioural Table (Output) Corresponding to Fig. 8

Trigger	Comparator o/p (Q)	S_0^M	S_1^M	F'	O/P
i=c (Trojan active)	0	1	1	Z	Z (DoS)
i≠c (Trojan inactive)	1	0	0	$M_3\_In_1$	Normal
	1	0	1	$A_3\_In_1$	
	1	1	0	$M_3\_In_2$	

3.6 Summary

This chapter presented the M-HLS framework, which demonstrates how Trojan attacks can be launched in the mux-based interconnect stage of HLS-generated watermarked IP cores. By evaluating the impact of PDHT and DoS-HT on a real-world IP design, it was shown that such attacks are feasible, stealthy, and damaging. The findings emphasize the importance of securing all stages of HLS, including those traditionally overlooked, to ensure end-to-end trust in hardware IP development.

Chapter 4

4. Robust Watermarking of Loop Unrolled Convolution Layer IP Design for CNN

4.1 Introduction

The increasing adoption of Convolutional Neural Networks (CNNs) in real-time image and video processing applications has led to a surge in the design of dedicated hardware accelerators, especially for convolutional layers—the most computation-intensive part of CNNs [35] [36]. Hardware implementations of these layers are often deployed as reusable IP cores in commercial consumer electronic (CE) products such as surveillance systems, medical imaging devices, and smart cameras [36].

However, the commercial value of these IPs makes them prime targets for piracy, unauthorized reuse, and tampering [37]. To counter these threats, watermarking is an effective solution that embeds ownership signatures directly into the hardware design, allowing for post-deployment verification. This chapter presents a robust watermarking framework for CNN convolutional IPs using loop unrolling and 4-variable encoded signature embedding during the register allocation phase of high-level synthesis (HLS) [38].

4.2 Threat Model

The threat model assumes that an unprotected CNN convolutional layer IP may be illegally reused, cloned, or resold by third-party vendors. The proposed watermarking technique enables the legal IP owner (vendor) to embed a unique signature during design, which remains intact through RTL and gate-level synthesis. This embedded signature allows SoC integrators to verify ownership and protect against false claims.

4.3 The Proposed Algorithm and Encoding Rule

The watermarking framework as shown in figure 10 operates during the architectural synthesis stage of the HLS flow and leverages the parallelism introduced by loop unrolling to embed security constraints with minimal design cost. The key steps of the proposed method include:

Input Processing: Accept a CDFG (Control/Data Flow Graph) representation of the CNN convolutional layer.

Loop Unrolling: Unroll convolutional loops to expose parallelism and enhance performance.

Scheduling and Resource Allocation: Schedule the operations and assign hardware resources (adders, multipliers, registers).

Register Allocation and Initial Table Generation: Construct the initial Register Allocation Table (RAT) (Fig. 9) representing storage variable lifetimes and overlapping constraints.

C.S.																																		
0	V0	V1	V2	V3	V4	V5													V35	V36	V37	V38	V39	V40										
1	V6	V6	V10		V8		V9	V7	V11	V12	V13	V14							V41		V42		V43		V44	V45	V46	V47	V48	V49					
2	V15						V16		V17		V18		V19	V20	V21	V22	V23	V24	V50				V43		V51		V52		V53		V54	V55	V56	V57	V58	V59
3	V25						V16		V17		V18		V26		V27		V28		V60						V51		V52		V53		V61		V62		V63	
4	V29		V29						V17		V18		V26		V27		V28		V64								V52		V53		V61		V62		V63	
5	V30	V30									V18		V26		V27		V28		V65										V53		V61		V62		V63	
6	V31		V31										V26		V27		V28		V66												V61		V62		V63	
7	V32	V32													V27		V28		V67														V62		V63	
8	V33																V28		V68																V63	
9	V34	V34																	V69																	

Fig. 9. RAT before and after embedding secret security constraints

Signature Encoding and Constraint Mapping: Convert the vendor's signature into a sequence of graph constraints based on a 4-variable encoding scheme.

Watermark Embedding: Modify the register allocation graph to insert the encoded constraints.

Conflict Resolution: Reallocate registers to satisfy added constraints while preserving functional correctness.

Datapath Synthesis: Generate the secure datapath with embedded watermark.

This process ensures seamless signature embedding without affecting functional behavior.

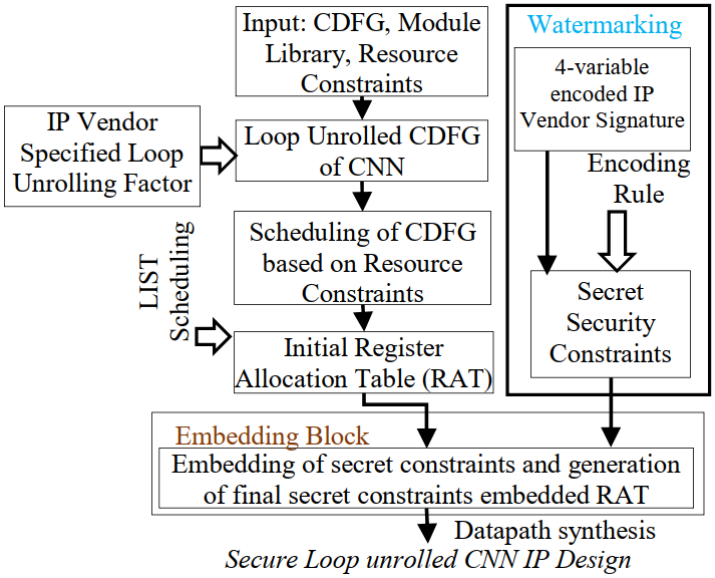

Fig. 10. The Proposed Methodology

4.4 Signature Encoding and Embedding

At the heart of the watermarking mechanism is a 4-variable encoding scheme. The purpose is to map alphanumeric characters (from a vendor-defined signature) to register allocation constraints based on node types in the interval graph. The four variables used in this encoding scheme are:

α (alpha): Represents a pair (0, prime)

β (beta): Represents a pair (0, even)

γ (gamma): Represents a pair (prime, odd)

λ (lambda): Represents a pair (odd, prime)

Each character in the digital signature is converted into one of the four symbols (α, β, γ, λ) using predefined encoding rules, which are then mapped to specific register pair constraints. These constraints are applied to the register allocation graph by inserting additional edges (conflicts) between storage variables, thereby forcing their allocation into distinct registers.

Encoding Example

Suppose the vendor's digital signature is α β β λ γ. Based on this sequence, five register-pair constraints are generated and inserted into the colored interval graph as edges. For instance:

α: Add edge between node with variable ID 0 and a node with a prime ID.

λ: Add edge between a node with odd ID and a node with prime ID.

This systematic embedding ensures that the signature is structurally encoded into the resource allocation behavior of the design.

4.5 Security Evaluation

Figure 11 illustrates the SDFG for a loop-unrolled CNN with an unrolling factor of 2, utilizing 6 multipliers and 2 adders [43]. The operations (1–34) are scheduled across nine control steps (1–9), requiring 36 registers, each represented by distinct colors and corresponding to storage variables Vn (V0, V2, …, V69).

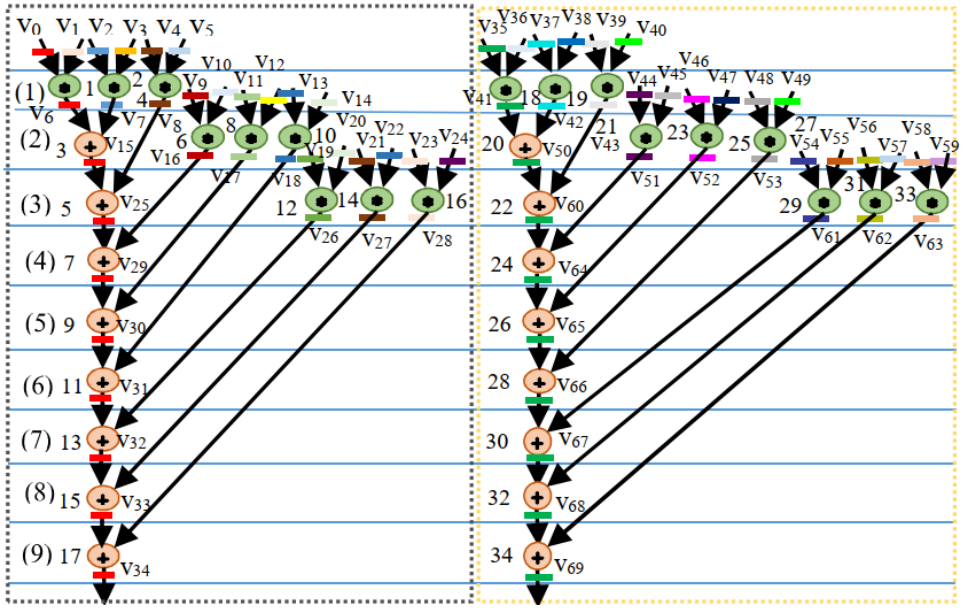

Fig. 11. Scheduled data flow graph of loop unrolled CNN using 6M and 2A

The robustness of the proposed approach is assessed using two key metrics:

Probability of Coincidence (PC): Measures the chance of an attacker accidentally replicating the same watermark in a pirated IP. PC drops exponentially with increasing signature size, e.g., PC = 4.9e-4 for a 250-digit signature.

Tamper Tolerance (TT): Reflects the resistance of the design to brute-force attacks. TT increases with signature size, e.g., TT = 3.27e+150 for a 250-digit signature.

4.6 Summary

This chapter introduced a secure and efficient watermarking methodology tailored for loop-unrolled CNN convolutional layer IPs. By embedding a 4-variable encoded signature during register allocation, the design ensures high tamper resistance and extremely low probability of coincidence—without compromising on performance or area. The approach demonstrates its effectiveness in protecting IPs against unauthorized reuse and strengthens hardware-level security in CE applications.

Chapter 5

5. Palmprint Biometric-Driven Hybrid GA-PSO Framework for Robust Watermark Embedding in HLS IP Designs

5.1 Introduction

High-Level Synthesis (HLS) is a powerful and commonly adopted technique for developing optimized and resource-efficient hardware architectures, particularly for handling computationally intensive tasks. In this design flow, Design Space Exploration (DSE) plays a crucial role in evaluating trade-offs among multiple design goals—such as area and latency—to identify hardware implementations best aligned with the target application's requirements. Heuristic algorithms are frequently utilized to assist DSE during HLS, enabling intelligent exploration of possible configurations to find optimal design solutions.

This exploration becomes more complex when multiple orthogonal objectives must be balanced—especially when incorporating hardware-level security such as watermarking, which can impact both resource usage and timing. The rise of reusable intellectual property (IP) cores in electronic and multimedia applications has simultaneously increased the risk of IP piracy and false ownership claims, making hardware security an essential concern. Watermarking is widely regarded as a highly effective post-deployment forensic defense strategy against such threats.

However, the challenge lies in embedding robust watermarks that enhance security while minimizing design cost. This is because stronger watermarks often demand additional registers or increase circuit latency, thereby inflating overall resource consumption. As a result, striking a balance between

watermark strength, area, and performance introduces a non-trivial multi-objective optimization problem during the HLS flow.

5.2 Framework Overview

The proposed framework optimizes palmprint biometric-based IP watermarking and loop unrolling factors during high-level synthesis (HLS) using a hybrid GA-PSO approach for design space exploration (DSE). It balances security and design cost for data-intensive applications. Inputs include the IP vendor's palmprint (Pv), watermark strength (Pw), feature sets (Pf), GA population (P), population size (Ps), loop iteration count, mutation (Pm) and crossover (Cp) probabilities, PSO mutation algorithm (PMa), fitness function weights (W1, W2), and diversity inclusion probability (PDi). The output is a secure, optimized IP core datapath with an unrolling factor.

The framework consists of two components:

1. **HLS-Based DSE System:** A GA-PSO hybrid explores design solutions, encoding parameters like adders (A), multipliers (M), unrolling factor (UF), and security constraints (Sc), optimizing for cost and security.
2. **Palmprint Biometric Watermarking:** This generates a watermark by capturing the palmprint, extracting features, encoding a binary signature, and embedding security constraints into the IP design's RTL, targeting FIR filters.

The process terminates after 50 iterations or if no improvement occurs over 10 iterations, yielding a globally optimal design evaluated by probability of coincidence (Pc) and tamper tolerance (TT). This ensures robust IP protection with minimal design overhead.

5.3 Proposed skeleton of secure optimal palmprint watermark

As depicted in Fig. 12, the proposed method takes several inputs: the IP vendor's palmprint (Pv), the maximum allowable strength of the palmprint biometric watermark (Pw), the number of palmprint feature sets (Pf), the GA chromosomal encoding and initial population (P), the population size (Ps), the maximum loop iteration count for the application, the mutation probability (Pm), the crossover probability (Cp), the PSO mutation algorithm (PMa), the weighting factors for the fitness function (W1 and W2), and the probability of diversity inclusion (PDi). The method outputs an optimized, secure watermarked datapath IP core design solution, including an unrolling factor. The proposed framework consists of two main components: a) an HLS-based DSE system that integrates a GA framework with PSO, and b) a palmprint biometric-based IP watermarking process. These components are elaborated in the following subsections.

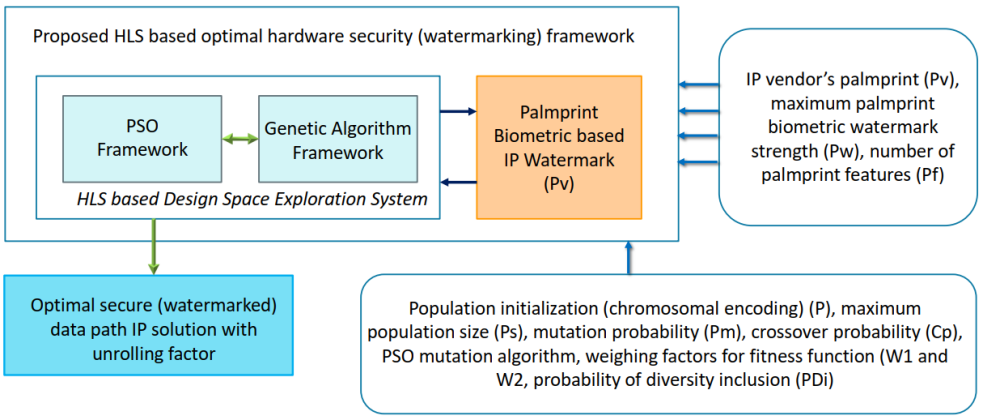

Fig. 12 Overview of proposed approach

5.4 Proposed hybrid GA-PSO based design space exploration system

Population Initialization and Chromosomal Encoding: The process begins by creating an initial population of parent chromosomes, labelled P1 to Pn, forming the first generation (G0) of the genetic algorithm (GA) within the design space exploration (DSE) framework. As illustrated in Fig. 12, the HLS-based DSE system uses an initial population encoded as:

$$Pi = (A, M, UF, Sc) \quad (1)$$

Here, each chromosome encodes the number of adders (A), multipliers (M), loop unrolling factor (UF), and palmprint-based security constraints (Sc). The genetic structure is detailed in Fig. 13. Once the population is initialized using this encoding (as per Eq. (1)), the next step involves a crossover operation to produce offspring.

Crossover and Mutation Operations

Crossover: During this phase, the genetic population combines traits from two parent chromosomes to create new offspring. In this method, crossover yields distinct offspring (O_1 to O_{12}), as shown in Fig. 14. Parents (P1, P2, P3, P4) and their unique offspring (O_1 to O_{12}) are carried forward to form the next generation (G1). Subsequently, the design cost (fitness function) is calculated for each chromosome in the population across iterations.

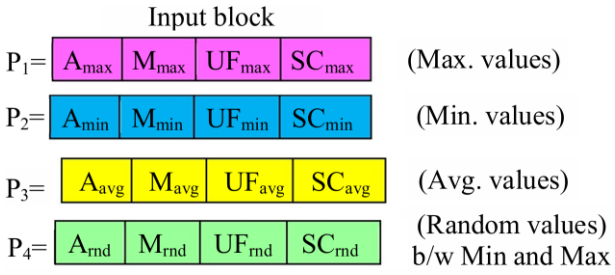

Fig. 13 Generation G0 Population Chromosomes

Populations	A	M	UF	Sc
P_1 and P_2: O_1	A_{max}	M_{max}	UF_{min}	SC_{min}
P_1 and P_3: O_2	A_{max}	M_{max}	UF_{avg}	SC_{avg}
P_1 and P_4: O_3	A_{max}	M_{max}	UF_{rnd}	SC_{rnd}
P_2 and P_1: O_4	A_{min}	M_{min}	UF_{max}	SC_{max}
P_2 and P_3: O_5	A_{min}	M_{min}	UF_{avg}	SC_{avg}
P_2 and P_4: O_6	A_{min}	M_{min}	UF_{rnd}	SC_{rnd}
P_3 and P_1: O_7	A_{avg}	M_{avg}	UF_{max}	SC_{max}
P_3 and P_2: O_8	A_{avg}	M_{avg}	UF_{min}	SC_{min}
P_3 and P_4: O_9	A_{avg}	M_{avg}	UF_{rnd}	SC_{rnd}
P_4 and P_1: O_{10}	A_{rnd}	M_{rnd}	UF_{max}	SC_{max}
P_4 and P_2: O_{11}	A_{rnd}	M_{rnd}	UF_{min}	SC_{min}
P_4 and P_3: O_{12}	A_{rnd}	M_{rnd}	UF_{avg}	SC_{avg}

Fig. 14 Generation G1 Population Chromosomes

Mutation Process: To increase diversity, the mutation technique from PSO-DSE [46] is adopted, applied to the two least-fit chromosomes every two generations (with a mutation probability Pm = 0.5). The rules are:

- For even-indexed chromosomes, left-shift the elements (A, M, UF), excluding Sc.
- For odd-indexed chromosomes, assign random values within the defined range for each constraint.

Mutated chromosomes are re-evaluated using the fitness function. If a mutated solution yields a lower design cost than the previous local best, the local best is updated. Further diversity is introduced by adding chromosomes with random gene values, governed by the probability of diversity inclusion (PDI).

5.5 Fitness Function Evaluation

The fitness function, or design cost, is determined using the area of the palmprint-watermarked design and the latency of the scheduled operations after unrolling, both derived from the chromosome's gene data. The area (A_r), latency (L), and design cost (CF) are expressed as [45] [46]:

$$A_r = A_{FU} + A_{MUX} + A_{REG} \quad (2)$$

Here, Ar represents the total area of the watermarked IP design, with A_{FU} being the area of functional units (adders and multipliers), A_{MUX} the area of multiplexers, and A_{REG} the area of registers.

$$L = (C_m * L_m) + (C_a * L_a) \quad (3)$$

In this equation, Cm denotes the watermark-embedded schedule control steps for multipliers, L_m is the multiplier latency, C_a is the control steps for adders, and L_a is the adder latency.

$$CF = (W_1 * (A_r - A_c) / A_{max}) + (W_2 * (L - L_c) / L_{max}) \quad (4)$$

Where W_1 and W_2 are set to 0.5, A_r is the area of the watermarked design, Ac is the IP vendor's area constraint, L is the schedule latency, and L_c is the latency constraint. This fitness function assesses each chromosome (parents and offspring) to identify the local best solution with the lowest design cost in the current generation. The process iterates until a global optimal solution is found.

As shown in Figure 12, the fitness function relies on data from HLS scheduling, allocation, and watermark-embedded register allocation, combined with weighting factors W_1 and W_2. The local best watermarked IP design is retained for the next generation. To enhance diversity and avoid local optima, a mutation step is introduced, integrating PSO-based DSE mutation techniques into the GA framework (detailed in the next subsection).

Termination Criteria and Global Best Solution

The framework stops under two conditions: a) reaching the maximum iteration limit, G (set to 50), or b) no improvement in the local best solution over λ iterations (set to 10). Upon termination, the final local best solution

becomes the global best design solution (Rgb), representing the output of the secure HLS-based DSE system.

5.6 Palmprint Biometric Watermark Generation

This subsection outlines the IP vendor's palmprint-based watermark creation, as depicted. The process includes:

Step 1: Input Collection: The IP vendor's palmprint is captured using a 12-megapixel digital camera (f/1 aperture, phase-detection autofocus), eliminating the need for an optical scanner.

Step 2: Nodal Point Representation: The palmprint is mapped onto a grid using nodal points, simplifying ownership verification and piracy detection without recapturing.

Step 3: Attribute Extraction and Filtering: Key palmprint attributes are extracted, excluding less relevant areas like the thumb. Attributes include measurements:

- **MLL**: Measurement of the space between the start of the life line and the end of the life line.
- **MHL**: Measurement of the space between datum points of the headline and life line.
- **PW**: Palm width.
- **PL**: Palm length.
- **MFF**: Measurement of the space between the first consecutive intersection points of the forefinger.
- **MSF**: Measurement of the space between the second consecutive intersection points of the forefinger.

- **MTF**: Measurement of the space between the third consecutive intersection points of the forefinger.
- **MFM**: Measurement of the space between the first consecutive intersection points of the middle finger.
- **MSM**: Measurement of the space between the second consecutive intersection points of the middle finger.
- **MTM**: Measurement of the space between the third consecutive intersection points of the middle finger.
- **MFR**: Measurement of the space between the first consecutive intersection points of the ring finger.
- **MSR**: Measurement of the space between the second consecutive intersection points of the ring finger.
- **MTR**: Measurement of the space between the third consecutive intersection points of the ring finger.
- **MFL**: Measurement of the space between the first consecutive intersection points of the little finger.
- **MSL**: Measurement of the space between the second consecutive intersection points of the little finger.
- **MTL**: Measurement of the space between the third consecutive intersection points of the little finger.

These measurements are normalized and converted into a binary string, forming a **biometric signature**, typically of 182 bits.

Step 4: Attribute Size Measurement: Distances between key points in the refined attributes are computed to determine their sizes.

Step 5: Binary Signature Creation: Attribute sizes are converted into binary form, then combined to generate a unique palmprint signature for the IP vendor.

Step 6: Encoding Security Constraints: The signature is encoded into security constraints using the IP vendor's specific encoding rule.

Step 7: Watermark Embedding: The security constraints are embedded into the HLS register allocation phase, integrating the watermark into the IP design's RTL structure.

5.7 Multivariable Signature Encoding Rules

The key to embedding the biometric signature into the IP design lies in translating the binary signature into register allocation constraints using multivariable encoding rules**.**

Register Allocation Background

In HLS, register allocation is often represented using a **Colored Interval Graph**, where:

- **Nodes** represent storage variables
- **Edges** represent overlapping lifetimes of variables (i.e., variables active at the same time and thus needing different registers)

To embed a watermark, additional artificial edges are introduced to the graph. These edges force the allocator to use distinct registers for specific variable pairs, thus encoding the watermark into the final hardware design.

Encoding Rules

Each binary or symbolic component of the palmprint signature is encoded as follows:

Symbol	Encoding Rule	Node Type Pair
'0'	Insert edge between even-indexed and odd-indexed nodes	(even, odd)
'1'	Insert edge between node 0 and any other integer node	(0, integer)
'.'	Insert edge between odd and prime-indexed nodes	(odd, prime)

These rules ensure that specific variable lifetimes cannot overlap, leading to **forced register separation**, which implicitly carries the encoded watermark.

For instance, if the bit sequence to encode is '01.0', then:

- Bit '0' → insert edge between node 2 and node 3 (even, odd)
- Bit '1' → insert edge between node 0 and node 5
- Symbol '.' → insert edge between node 5 (odd) and node 7 (prime)
- Bit '0' → again insert a new (even, odd) pair

This strategy does not increase the number of registers**, but it** constrains their allocation**,** embedding the watermark invisibly into the data path's storage structure.

Demonstration on FIR Filter

The proposed framework is demonstrated using an FIR filter with specific design specifications: 8 multipliers, 4 adders, a loop unrolling factor (UF) of 8, and a watermark strength (Sc) of 182 (Fig. 15). The watermark signature is generated by concatenating extracted palmprint features, such as palm width and lifeline measurements, forming a 182-digit binary string that is embedded into the design to ensure security.

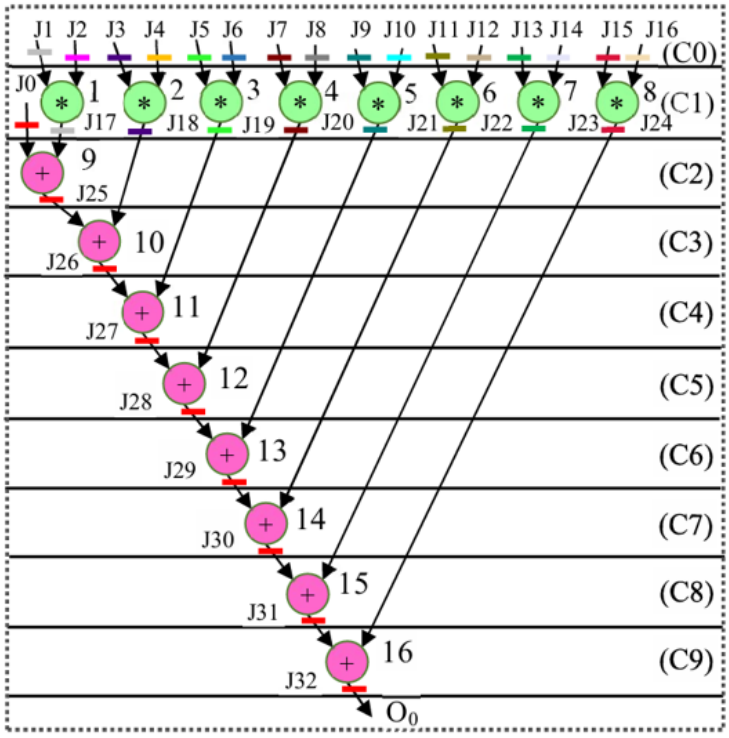

Fig. 15 Scheduled DFG using 8M,4A, UF8 and 1M,1A UF1

Table 3. RAT Before and After Embedding Proposed Palmprint Biometric Watermark Constraints

Registers	Red(R)	Silver(S	Purple(P	Indigo(I	Orange(O	Lime(LI	Blue(B)	Maroon(M	Gray(GY	Teal(T)	Aqua(A	Olive(OL	Khaki(K	Green(G	Lavender	Crimson(	Wheat(W
CO		J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13	J14	J15	J16
C1	J0	J17		J18		J19		J20		J21		J22		J23		J24	
C2	~~J25~~	J25		J18		J19		J20		J21		J22		J23		J24	
C3	~~J26~~	J26				J19		J20		J21		J22		J23		J24	
C4	~~J27~~	J27						J20		J21		J22		J23		J24	
C5	~~J28~~	J28								J21		J22		J23		J24	
C6	~~J29~~	J29										J22		J23		J24	
C7	~~J30~~	J30												J23		J24	
C8	~~J31~~	J31														J24	
C9	~~J32~~	J32															

5.8 Watermark Embedding into HLS Flow

Once the binary signature is encoded into constraints, the embedding takes place during the **register allocation phase** (Table 3) of the HLS process. The steps are:

1. **Scheduling**: Loop-based IP logic is scheduled as per data and resource constraints.
2. **Hardware Allocation**: Operations are mapped to available hardware units (e.g., adders, multipliers).
3. **Register Allocation (with Encoding)**: The previously defined encoding constraints are applied to the interval graph, altering how registers are assigned to variables. This process ensures the biometric watermark is embedded structurally.

Notably, the watermark is embedded **without requiring extra hardware resources**, making it highly efficient.

5.9 Design Space Exploration Using Hybrid GA-PSO

While embedding a watermark, performance metrics like **latency** and **area** must also be optimized. The framework as in figure 16 uses **hybrid GA-PSO** to determine the optimal:

- Number of functional units (adders, multipliers)
- Loop unrolling factor (UF)
- Encoded security constraints (Sc)

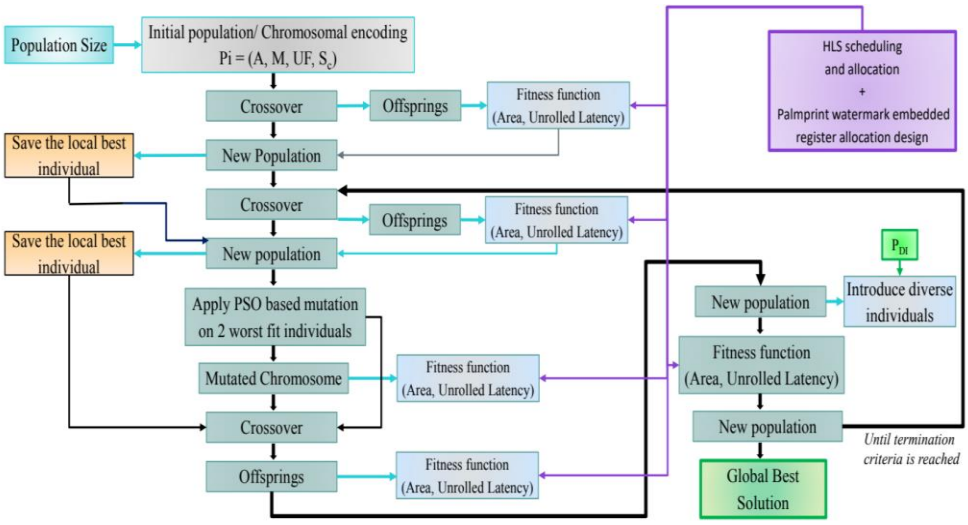

Fig. 16. Flow diagram of DSE using Hybrid GA-PSO Framework

Each candidate solution (chromosome) encodes (A, M, UF, Sc) and is evaluated based on a **cost function** combining latency and area. The optimization process includes:

- **Selection** based on fitness
- **Crossover and mutation** for diversity
- **Swarm behavior** (inspired by PSO) to guide exploration
- **Diversity injection** to avoid local minima

This hybrid mechanism ensures the **watermark embedding is optimal** from both a performance and security standpoint.

5.10 Summary

This chapter presented a hybrid GA-PSA framework for concurrent design space exploration (DSE) [41] of optimal palmprint biometric based intellectual property (IP) watermark and optimal loop unrolling factor that utilizes palmprint-derived features and a rule-based encoding scheme to embed unique, robust signatures directly into the register allocation stage of HLS. By mapping signature bits into graph-based constraints, the watermark

is embedded invisibly and securely. The encoding rules ensure complexity and non-replicability, while the hybrid GA-PSO optimization guarantees minimal design overhead. The proposed method not only secures the IP against piracy but also integrates seamlessly into standard HLS flows, making it both effective and practical for industry deployment.

Chapter 6

6. Results and Analysis

6.1 Results and Analysis: M-HLS Framework for Watermarked IPs

This section presents a detailed evaluation of the proposed Malevolent High-Level Synthesis (M-HLS) framework, which facilitates stealthy insertion of two types of hardware Trojans—Performance Degradation Hardware Trojan (PD-HT) and Denial-of-Service Hardware Trojan (DoS-HT)—into HLS-generated watermarked IPs. The framework was tested on the MESA IP core using the **NanGate 15nm technology library** [47].

Area Overhead Analysis

Table 4 reports the gate count of the baseline watermarked IP and its Trojan-inserted variants. The insertion of PD-HT and DoS-HT results in a minimal area overhead of approximately 2.65% compared to the original design. This demonstrates that the M-HLS framework can stealthily embed malicious logic with negligible design cost overhead.

Table 4. Area Overhead due to Trojan insertions

Parameters	Baseline IP design	IP design with PD-HT	IP design with DoS-HT
Area(gate count)	7424	7622	7620
Area overhead w.r.t. baseline (gate count)	--	2.66%	2.64%

Performance Degradation Characteristics

The performance degradation impact of PD-HT is controlled by the number of inverters inserted by the attacker. Table 5 summarizes the degradation observed for different values of inverters (x). At x = 24, the design shows up to 27.37% degradation in performance compared to the baseline.

Table 5. Performance Degradation with variations in number of invertors used

Parameters	**X =10**	**X =16**	**X =24**
Area overhead	2.74%	2.83%	2.94%
Performance degradation	11.41%	18.25%	27.37%

Power Overhead

As seen in Table 6, the insertion of either Trojan type incurs an average power overhead of approximately 3.14%, indicating that the malicious logic remains lightweight and does not significantly impact power consumption.

Table 6. Power Overhead due to Trojan insertions

Parameters	**Baseline IP design**	**IP design with PD-HT**	**IP design with DoS-HT**
Power(µw)	64.81	66.85	66.85
Power overhead w.r.t. baseline	--	3.14%	3.14%

Comparative Evaluation

In comparison to prior work [7], the M-HLS framework shows competitive or improved efficiency. For the FIR IP core, the area overhead using M-HLS is up to 1.89% versus 4% in [7]. Moreover, the M-HLS Trojan achieves more than 27% performance degradation on the MESA IP core, which was not supported in [7].

Trojan Evasion from Detection Techniques

Table 7 summarizes the limitations of existing Trojan detection techniques in identifying the M-HLS Trojans. Due to stealthy insertion at the HLS level and the absence of significant functional or side-channel deviations, these Trojans evade detection by known techniques like TL-HLS [24], high-level transformation-based detection [42], and GNN-based detection [25].

Table 7. Demonstration of Trojan Evasion for known Detection Techniques.

Detection approach	**Features in proposed IP (baseline) - Design A**	**Features in proposed IP (with Trojan) Design B**	**Justification of evasion**	**Trojan Status**
TL-HLS [24]	# of distinct 3PIP vendor: 1 (U^{OG}: 14 opn. allocations)	# of distinct 3PIP vendor: 1 (U^{DP}: 14 opn. allocations)	Difference in functional o/p between designs A & B using comparator: Nil	Not detected
HLS based detection [42]	Opn. count (MESA): 14 HLT: 0	Opn. count (MESA): 14 HLT: 0	Difference in opn. count between designs A & B: Nil	Not detected
GNN based [25]	>1000 lines of VHDL code for MESA processor (datapath & controller)IP	> 1000 lines of VHDL for MESA processor (datapath & controller) IP	Incapable of handling VHDL using Pyverilog, weaker learning	Not detected

6.2 Results and Analysis: Robust Watermarking of Loop Unrolled Convolution Layer IP Design for CNN

This section evaluates the proposed watermarking methodology applied to CNN convolutional layer IP designs, emphasizing its security robustness and design efficiency. The assessment is conducted in terms of design cost, overhead, and security strength using metrics such as tamper tolerance (TT) and probability of coincidence (PC).

Design Cost and Overhead Analysis

The proposed approach is tested using a CNN convolution layer IP where loop unrolling is performed to improve throughput. The watermark is embedded during the register allocation phase using a 4-variable encoding strategy, without altering the functional correctness of the IP.

Design cost is computed as:

$$\textbf{Design Cost} = 0.5\times(\text{Area}_{IP}/\text{Area}_{Max}) + 0.5\times(\text{Latency}_{IP}/\text{Latency}_{Max}) \quad (1)$$

The proposed watermarking technique incurs **zero design cost overhead**. This highlights the strength of the encoding scheme, which successfully embeds security without increasing area or latency.

Register Allocation Table (RAT) Transformation

The Register Allocation Table (RAT) before and after the insertion of watermarking constraints shows that additional register-to-variable bindings (based on 4-variable encodings) do not alter the execution semantics. Instead, the forced distinct register allocation enhances the watermark's uniqueness and stealth.

Tamper Tolerance and Probability of Coincidence

To measure the security strength of the watermark, two key metrics are used:

- **Probability of Coincidence (PC):** The likelihood of randomly generating the same watermark signature in a pirated design.
- **Tamper Tolerance (TT):** The robustness of the design against brute-force tampering attacks.
  - $PC = (1-1/r)^S$ (2)
  - $TT = P^S$ (3)

Where:

  - r = number of available registers
  - S = length of the watermark signature

Table 8 presents the variation in PC and TT with increasing signature size. As the signature length increases, PC drops exponentially, while TT increases significantly.

Comparison with Existing Approaches

The proposed methodology is benchmarked against prior works [26], [27], and [28] based on signature size, TT, and PC (Table 8). The results clearly show superior performance:

Table 8. Comparison of security in terms of TT and PC with previous works

Approach	Signature Size	Tamper Tolerance (TT)	Probability of Coincidence (PC)
Proposed	250 digits	3.27 E+150	4.9 E-4
[27]	128 digits	3.40 E+38	2.0 E-3
[28]	83 bits	9.67 E+24	7.9 E-2
[26]	8 digits	6561	7.8 E-1

These results confirm that the proposed watermarking framework delivers enhanced security with minimal design cost, making it suitable for high-performance and sensitive CNN IPs in SoC environments.

6.3 Results and Analysis: Hybrid GA-PSO Framework

This section presents the evaluation of the proposed Hybrid GA-PSO-based Design Space Exploration (DSE) framework aimed at discovering the globally optimal combination of palmprint biometric watermarking and loop unrolling factor. The effectiveness of the approach is assessed with respect to robustness, design cost, and convergence performance.

Evaluation Setup and Objectives

The DSE framework is tested across different FIR filter configurations (8-tap, 20-tap, 60-tap, 100-tap) and population sizes (P = 4, 6, 8), with key tuning parameters set as follows:

- Mutation probability: Pm = 0.5
- Crossover probability: Cp = 1
- Probability of Diversity Inclusion: $P_{DI} = 0.33$
- Weight factors for fitness: W1 = W2 = 0.5
- Max signature strength Sc(max) = 182 digits

- Max generations: G = 50

The evaluation focuses on:

- Sensitivity Analysis
- Pareto Optimal Set Generation
- Security Metrics: Probability of Coincidence (Pc) and Tamper Tolerance (TT)
- Convergence Performance

Sensitivity and Design Cost Evaluation

Sensitivity analysis (Table 9) explores how different population sizes affect the global best solution and associated design cost (Cf). The proposed hybrid framework consistently finds high-quality global solutions across all FIR filter benchmarks with minimal design cost. This validates the robustness and adaptability of the approach across diverse parameter spaces.

Table 9. Sensitivity Analysis for FIR Filters for Proposed Approach

Benchmarks	**P=4**		**P=6**		**P=8**	
	Rgb	**C_F**	**Rgb**	**C_F**	**Rgb**	**C_F**
8 Tap	[2,2,4,140]	-0.21	[2,2,2,140]	-0.21	[1,2,2,154]	0.23
20 Tap	[3,4,15,150]	-0.29	[1,4,10,161]	-0.28	[1,5,10,161]	-0.28
60 Tap	[1,5,1,173]	-0.36	[3,5,3,172]	-0.36	[3,6,40,152]	-0.33
100 Tap	[1,7,62,144]	-0.38	[1,7,20,156]	-0.37	[1,7,40,140]	-0.38

Pareto Optimal Set and Encoding Metrics

The approach successfully generates multiple Pareto optimal design points for each FIR filter configuration. These represent the most efficient solutions (Table 10) that jointly satisfy both area and latency constraints, forming the optimal design boundary in the explored solution space.

Table 10. Pareto Optimal Set Generation for Proposed Framework

Benchmarks	P=4	P=6	P=8
8 Tap	233	264	178
20 Tap	406	343	391
60 Tap	503	719	471
100 Tap	565	1088	1036

Watermark Security Evaluation

Two widely adopted hardware watermarking security metrics—Probability of Coincidence (Pc) and Tamper Tolerance (TT)—are used to assess the resilience of the proposed watermarking approach (Table 11).

- **Probability of Coincidence (Pc):** It is the likelihood of randomly encountering the same watermark signature in a pirated/unsecured IP design. Lower Pc implies higher robustness against watermark collision or false positive detection.

$$\mathbf{Pc = (1-1/c)^r} \quad (4)$$

Where, 'c' represents the count of colors (registers) utilized before incorporating the watermark security constraints, while 'r' signifies the number of encoded digits used in watermark security constraints.

- **Tamper Tolerance (TT):** TT indicates the resilience of the watermark against brute force tampering. Higher TT values signify enhanced robustness. Reported results demonstrate significantly low Pc and extremely high TT values, showing strong resistance against tampering and signature forging.

$$\mathbf{TT = V^r} \quad (5)$$

In this context, 'V' denotes the overall number of unique variables employed within the watermark encoding rules, while 'r' indicates number of security

watermark constraints generated (In case of proposed approach, the number of optimal security constraints explored in the global best design solution).

Table 11. Analysis of PC and TT

Benchmarks	Register (c)	PC	TT
8 Tap	17	8.81E-5	2.99E+73
20 Tap	41	1.87E-2	6.55E+76
60 Tap	121	2.37E-1	3.48E+82
100 Tap	201	4.59E-1	2.69E+74

Comparative Security Evaluation

The proposed watermarking method significantly outperforms other leading techniques like Physical Level Watermarking, Multilevel Watermarking, and Facial Biometric Watermarking. It offers lower Pc (higher collision resistance) and higher TT (better tamper resilience) [28] [31] [33], which are attributed to the dynamic encoding of palmprint biometric features and the broader search space enabled by GA-PSO.

Convergence Performance

The convergence analysis (Table 12) evaluates how quickly the proposed DSE framework stabilizes to an optimal solution. Results show convergence occurs within practical generation counts and time spans, validating the hybrid model's efficiency in reaching globally optimal solutions with minimal computational overhead.

Table 12. Convergence Analysis of Proposed Work

Benchmarks	P=4		P=6		P=8	
	Ic	Tc(ms)	Ic	Tc(ms)	Ic	Tc(ms)
8 Tap	G5	170.64	G7	413.79	G1	69.62
20 Tap	G9	705.73	G5	519.28	G3	398.65
60 Tap	G15	3226.50	G11	2355.0	G5	1013.27
100 Tap	G11	2807.86	G19	8519.4	G25	13692.03

- **Convergence Condition:** Process stops when iteration reaches ($\lambda = 10$) with no change.
- **Results:** Tc and the iteration at convergence (Ic) are recorded for each configuration, showing the efficiency of the approach in reaching optimal solutions.

Chapter 7
7. Conclusion and Future Scope

7.1 Conclusion

This thesis has explored three critical advancements in the domain of hardware security and IP core protection, focusing on vulnerabilities introduced during high-level synthesis (HLS), robust watermarking of loop unrolled IPs, and the optimization of biometric-based watermarking using evolutionary computation.

Firstly, a **Malevolent HLS (M-HLS) framework** was proposed that demonstrates how an attacker can exploit hardware vulnerabilities introduced during the watermarking process of HLS-generated IPs. Two novel stealthy hardware Trojans—**Performance Degradation Hardware Trojan (PD-HT)** and **Denial-of-Service Hardware Trojan (DoS-HT)**—were embedded into the multiplexer-based interconnect stage. Experimental evaluation confirmed minimal area and power overhead while achieving significant performance degradation, along with the ability to evade several state-of-the-art detection mechanisms.

Secondly, a **Robust Watermarking methodology** was introduced for loop unrolled convolutional IP designs used in CNN accelerators. This method utilizes a **4-variable encoded register allocation** scheme to embed the designer's signature during architectural synthesis. The encoding mechanism enhances tamper tolerance while preserving performance, and the results show significant improvement in security metrics such as **Probability of Coincidence (PC)** and **Tamper Tolerance (TT)** compared to previous works.

Finally, an HLS-driven **Hybrid GA-PSO optimization framework** was developed for embedding palmprint biometric watermarks and jointly

optimizing the loop unrolling factor. This method achieves a well-balanced trade-off among area, latency, and security strength, guiding the designer to select Pareto-optimal solutions during architectural synthesis. The use of biometric signatures introduces uniqueness, while the expandable encoding dictionary and evolutionary optimization further strengthen watermark robustness.

Across all three contributions, the thesis has emphasized secure IP design at the early stages of synthesis, offering novel methodologies for proactive security integration. The proposed frameworks are validated on real-world benchmarks and show promising improvements in robustness, performance, and stealth compared to state-of-the-art techniques.

7.2 Future Scope

The scope of this research can be extended in several promising directions:

1. **Automated Trojan Detection in HLS Flow:** Future work could focus on developing lightweight and scalable detection techniques specifically tailored to pre-RTL Trojan detection during HLS. Integration with machine learning models (e.g., LLMs, GNNs) trained on hardware synthesis flows could be investigated.
2. **Extension to Analog/Mixed-Signal IPs:** The current methodologies primarily target digital IPs. Applying similar watermarking and security techniques to analog/mixed-signal or RF circuits could open new research avenues.
3. **Multi-biometric and Behavioral Watermarking:** Combining palmprint with other biometric modalities such as iris, ECG, or behavioral patterns could further enhance uniqueness and tamper resistance in watermarking.
4. **Adaptive Watermarking Frameworks:** Future designs could involve adaptive watermarking that adjusts based on IP usage patterns, threat

levels, or environmental conditions, possibly leveraging reconfigurable hardware platforms like FPGAs.

5. **Hardware–Software Co-Security Techniques:** Investigating unified frameworks that secure both hardware IPs and the embedded software running on them, using joint watermarking or secure boot mechanisms, would be a logical next step in real-world applications.
6. **Integration into Commercial EDA Tools:** Collaborating with EDA tool vendors to integrate watermark embedding and verification modules directly into commercial HLS/RTL tools can help translate this research into industry-grade solutions.

REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, Apr. 1965.

[2] A. Sangiovanni-Vincentelli, "Electronic design automation: Past, present, and future," *Proc. IEEE*, vol. 95, no. 3, pp. 467–506, Mar. 2007.

[3] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*, 2nd ed., Upper Saddle River, NJ, USA: Prentice-Hall, 2003.

[4] S. Palnitkar, *Verilog HDL: A Guide to Digital Design and Synthesis*, 2nd ed., Prentice-Hall, 2003.

[5] M. Keating and P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs*, 3rd ed., Springer, 2002.

[6] W. Wolf, *Modern VLSI Design: IP-Based Design*, 4th ed., Prentice-Hall, 2008.

[7] A. Sengupta and S. Mohanakrishnan, "Efficient IP core integration methodology for consumer SoC design," *IEEE Trans. Consumer Electron.*, vol. 58, no. 2, pp. 428–436, May 2012.

[8] D. D. Gajski, N. Dutt, A. Wu, and S. Lin, *High-Level Synthesis: Introduction to Chip and System Design*, Springer, 1992.

[9] A. Sengupta and D. Bhattacharya, "Quality driven behavioral IP protection in high-level synthesis for SoC design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 10, pp. 1516–1529, Oct. 2014.

[10] S. Gupta, R. Gupta, N. Dutt, and A. Nicolau, "SPARK: A high-level synthesis framework for applying parallelizing compiler transformations," in *Proc. Int. Conf. VLSI Design*, 2003, pp. 461–466.

[11] A. Sengupta, A. Ranjan, and S. Mohanakrishnan, "A survey of IP watermarking techniques for hardware design protection," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–39, Jan. 2018.

[12] A. Sengupta and P. K. Mishra, "High-level synthesis-driven IP watermarking methodology for SoC designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 11, pp. 1794–1805, Nov. 2017.

[13] S. Narasimhan et al., “Hardware IP trust: Side-channel-based watermarking for IP protection,” in *Proc. ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2010, pp. 1–4.

[14] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, “Trustworthy hardware: Identifying and classifying hardware Trojans,” *Computer*, vol. 43, no. 10, pp. 39–46, Oct. 2010.

[15] D. Bhattacharya and A. Sengupta, “Hardware Trojan detection using layout-centric and logic-centric approaches in high-level synthesis,” in *Proc. Int. Symp. Hardware Oriented Security and Trust (HOST)*, 2014, pp. 40–47.

[16] A. Sengupta and R. S. Chakraborty, “Register allocation-level invisible watermarking for IP protection of high-level synthesis designs,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1675–1684, Sep. 2019.

[17] A. Canis et al., “LegUp: An open-source high-level synthesis tool for FPGA-based processor/accelerator systems,” *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 2, pp. 24:1–24:27, Sep. 2013.

[18] A. Sengupta and A. Dey, *High-Level Synthesis: Fundamentals and Recent Advancements in IP Protection*, SpringerBriefs in VLSI Design and Embedded Systems, 2022.

[19] C. Pilato, K. Basu, F. Regazzoni, and R. Karri, “Black-hat high-level synthesis: Myth or reality?” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 4, pp. 913–926, Apr. 2019.

[20] M. Abderehman, R. Gupta, R. R. Theegala, and C. Karfa, “BLAST: Belling the black-hat high-level synthesis tool,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 3661–3672, Nov. 2022.

[21] Y. Jin and Y. Makris, “Hardware Trojan detection using path delay fingerprint,” in *Proc. IEEE Int. Workshop HOST*, 2008, pp. 51–57.

[22] A. Sengupta and D. Roy, “Antipiracy-aware IP chipset design for CE devices: A robust watermarking approach,” *IEEE Consum. Electron. Mag.*, vol. 6, no. 2, pp. 118–124, Apr. 2017.

[23] F. Koushanfar, I. Hong, and M. Potkonjak, “Behavioral synthesis techniques for intellectual property protection,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, pp. 523–545, Jul. 2005.

[24] 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,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 4, pp. 655–668, Apr. 2017.

[25] R. Yasaei, L. Chen, S.-Y. Yu, and M. A. A. Faruque, “Hardware Trojan detection using graph neural networks,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, May 26, 2022, doi: 10.1109/TCAD.2022.3178355.

[26] A. Sengupta and M. Rathor, “Enhanced security of DSP circuits using multi-key based structural obfuscation and physical-level watermarking for consumer electronics systems,” *IEEE Trans. Consumer Electron.*, vol. 66, no. 2, pp. 163–172, 2020.

[27] A. Sengupta, E. R. Kumar, and N. P. Chandra, “Embedding digital signature using encrypted-hashing for protection of DSP cores in CE,” *IEEE Trans. Consumer Electron.*, vol. 65, no. 3, pp. 398–407, Aug. 2019.

[28] A. Sengupta and R. Chaurasia, “Secured convolutional layer IP core in convolutional neural network using facial biometric,” *IEEE Trans. Consumer Electron.*, vol. 68, no. 3, pp. 291–306, Aug. 2022.

[29] A. Sengupta and M. Rathor, “Enhanced security of DSP circuits using multi-key based structural obfuscation and physical-level watermarking for consumer electronics systems,” *IEEE Trans. Consumer Electron.*, vol. 66, no. 2, pp. 163–172, 2020.

[30] D. Roy and A. Sengupta, “Multilevel watermark for protecting DSP kernel in CE systems,” *IEEE Consum. Electron. Mag.*, vol. 8, no. 2, pp. 100–102, 2019.

[31] A. Sengupta, D. Roy, “Automated low cost scheduling driven watermarking methodology for modern CAD high-level synthesis tools”, *Elsevier J. Advances in Engineering Software*, vol 110, 2017, pp 26-33.

[32] A. Sengupta and D. Roy, “Automated low-cost scheduling driven watermarking methodology for modern CAD high-level synthesis tools,” *Adv. Eng. Softw.*, vol. 110, pp. 26–33, 2017.

[33] F. Koushanfar, I. Hong, and M. Potkonjak, “Behavioral synthesis techniques for intellectual property protection,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, no. 3, pp. 523–545, 2005.

[34] A. Sengupta, A. Anshul, V. Chourasia, and N. Kumar, “M-HLS: Malevolent high-level synthesis for watermarked hardware IPs,” *IEEE Embed. Syst. Lett.*, vol. 16, no. 4, pp. 497–500, Dec. 2024.

[35] “An FPGA-Based Accelerator Enabling Efficient Support for CNNs with Arbitrary Kernel Sizes,” *arXiv*, 2024.

[36] “Convolutional neural network IP core based on FPGA,” *CN109784489B*, Google Patents, 2019.

[37] “Rethinking Watermark: Providing Proof of IP Ownership in Modern SoC Designs,” *IACR Cryptology ePrint Archive*, 2022.

[38] A. Anshul and A. Sengupta, “A survey of high-level synthesis based hardware security approaches for reusable IP cores [Feature],” *IEEE Circuits Syst. Mag.*, vol. 23, no. 4, pp. 44–62, Q4 2023.

[39] A. Sengupta, A. Anshul, V. Chourasia, and N. Bhui, “Security vulnerability (backdoor Trojan) during machine learning accelerator design phases,” *IT Prof.*, vol. 27, no. 1, pp. 65–72, Jan.–Feb. 2025.

[40] S. Akter, K. Khalil, and M. Bayoumi, “A survey on hardware security: Current trends and challenges,” *IEEE Access*, vol. 11, pp. 77543–77565, 2023.

[41] V. K. Mishra and A. Sengupta, “MO-PSE: Adaptive multi-objective particle swarm optimization based design space exploration in architectural synthesis for application specific processor design,” *Adv. Eng. Softw.*, vol. 67, pp. 111–124, Jan. 2014.

[42] M. Rathor and A. Sengupta, “Revisiting black-hat HLS: A lightweight countermeasure to HLS-aided Trojan attack,” *IEEE Embed. Syst. Lett.*, vol. 16, no. 2, pp. 170–173, Jun. 2024.

[43] A. Sengupta, V. Chourasia, A. Anshul and N. Kumar, "Robust Watermarking of Loop Unrolled Convolution Layer IP Design for CNN using 4-variable Encoded Register Allocation," 2024 *International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, Taichung, Taiwan, 2024.

[44] Sengupta, V. Chourasia and N. Kumar, "HLS driven Hybrid GA-PSO for Design Space Exploration of Optimal Palmprint Biometric based IP Watermark and Loop Unrolling Factor," 2024 *IEEE International Symposium on Smart Electronic Systems (iSES)*, New Delhi, India, 2024.

[45] Anirban Sengupta, Reza Sedaghat, Zhipeng Zeng, "Rapid Design Space Exploration for multi parametric optimization of VLSI designs", Proc of 2010 *IEEE International Symposium on Circuits and Systems (ISCAS)*, Paris, France, pp: 3164-3167, 2010.

[46] Vipul Kumar Mishra, Anirban Sengupta, "MO-PSE: Adaptive Multi Objective Particle Swarm Optimization Based Design Space Exploration in Architectural Synthesis for Application Specific Processor Design", *El sevier Journal on Advances in Engineering Software*, Volume 67, January 2014, pp. 111-124.

[47] (Silicon Integr. Initiat. Inc., Vadodara, Gujarat). *Open Cell NanGate Library, 15 nm Open Cell Library*. Accessed: Feb. 2024. [Online]. Available: https://si2.org/open-cell-library/