ENCODED ENCRYPTION AND BIOMETRICS FOR HIGH-LEVEL SYNTHESIS BASED HARDWARE SECURITY AGAINST IP PIRACY AND FRAUD IP OWNERSHIP

MS (Research) THESIS

By

Bharath Kollanur

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2023

ENCODED ENCRYPTION AND BIOMETRICS FOR HIGH-LEVEL SYNTHESIS BASED HARDWARE SECURITY AGAINST IP PIRACY AND FRAUD IP OWNERSHIP

A THESIS

Submitted in fulfilment of the
requirements for the award of the degree
of

Master of Science (Research)

by

Bharath Kollanur

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
MAY 2023

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **ENCODED ENCRYPTION AND BIOMETRICS FOR HIGH-LEVEL SYNTHESIS BASED HARDWARE SECURITY AGAINST IP PIRACY AND FRAUD IP OWNERSHIP** in the fulfilment of the requirements for the award of the degree of **MASTER OF SCIENCE (RESEARCH)** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from August 2021 to May 2023 under the supervision of Dr Anirban Sengupta, Associate Professor, IIT Indore.

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

03/07/2023
Signature of the student with date

BHARATH KOLLANUR

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

July 4, 2023
Signature of the Supervisor of MS (Research) thesis (with date)

DR ANIRBAN SENGUPTA

K BHARATH has successfully given his MS (Research) Oral Examination held on **JULY 3 2023**.

04/07/2023
Signature of Chairperson (OEB) with date

July 4, 2023
Signature(s) of Thesis Supervisor(s) with date

04/07/23
Signature of Convener, DPGC with date

4/7/2023
Signature(s) of head of Discipline with date

ACKNOWLEDGEMENTS

Completing a thesis is a challenging and demanding journey that requires perseverance, hard work, and support from many individuals. As I look back at my journey, I am grateful to the many people who have played a significant role in making this accomplishment possible.

Firstly, I would like to express my deepest gratitude to my thesis advisor, Prof Anirban Sengupta, for their guidance, patience, and support throughout my research journey. Their insightful feedback, constructive criticism, and expertise have been invaluable in shaping the direction and scope of my work. I am grateful for the opportunities and challenges they provided me and their unwavering encouragement and support during good and difficult times.

I would like to thank my lab seniors, Mr. Rahul Chaurasia and Mr. Aditya Anshul, for their invaluable guidance, mentoring, and support during my research journey. Their expertise and dedication to their work have been a constant source of inspiration for me, and I have learned so much from their vast knowledge and experience. Their continuous encouragement and motivation have been instrumental in shaping my research work and career goals. I would also like to extend my gratitude to my lab senior, Dr. Mahendra Rathor, who has passed out from our lab. Their invaluable contributions, advice, and support have played a signification role in shaping my research journey. Their experiences, insights, and guidance have been a valuable asset, and I am grateful for their continuous support and encouragement.

I would also like to thank the members of my thesis committee, Prof Abhishek Srivastava, Prof Somaditya Sen, Prof Somnath Dey, Prof Neminath Hubballi, and Prof Aniruddha Singh Kushwaha, for their

valuable feedback, critical evaluation, and insightful suggestions. Their expertise and guidance helped me navigate through the complexities of the research process and provided me with the necessary guidance to produce a quality thesis. I would like to express my heartfelt thanks to the faculty and staff of the Computer Science and Engineering department at IIT Indore for providing me with a stimulating and supportive environment to pursue my research work. I am grateful for the resources, infrastructure, and opportunities provided to me by the department, which have been instrumental in shaping my research journey.

I am also deeply grateful to my family and friends for their love, encouragement, and unwavering support throughout my academic journey. Their patience, understanding, and motivation have been the driving force behind my success. I particularly acknowledge my parents, Raja Kumar and Manjula, and my brother, Dr. Charan, for their endless support, encouragement, and belief in me. I am also grateful to Vaishnavi and Shibani Das for their love, encouragement, and unwavering support.

Lastly, I would like to express my gratitude to all the individuals who have helped me in various ways during my research journey, including my peers, colleagues, and other professionals who have provided me with their valuable insights, support, and assistance.

Completing this thesis has been a challenging but rewarding experience, and I am grateful to everyone who has contributed to my journey.

ABSTRACT

This thesis presents two novel methodologies for securing intellectual property (IP) core designs against hardware security threats. The first methodology is a quadruple-phase watermarking technique for securing hardware IP cores during high-level synthesis (HLS). In contrast, the second methodology explores unified biometrics with an encoded dictionary for the hardware security of fault-secured digital signal processing (DSP) intellectual property (IP) core designs.

The first methodology addresses the issue of IP piracy and ownership infringement that poses a significant threat to the security of authentic IP vendors. The proposed quadruple-phase watermarking technique employs graph portioning, encoding tree, and eightfold mapping to generate a robust watermarking signature. The signature is embedded at four stages of HLS, including scheduling, register binding, resource binding, and interconnect binding, to ensure high-quality hardware security constraints. The results demonstrate a considerable decrease in the probability of coincidence and a higher level of tamper tolerance compared to the state-of-the-art techniques without incurring a significant design cost overhead.

The second methodology focuses on the hardware security of fault-secured digital signal processing (DSP) intellectual property (IP) core designs against IP piracy. The methodology exploits scheduled and allocated DSP design using a behavioural synthesis process to generate a fault-secured DSP IP core. The proposed technique embeds encoded unified biometric-based hardware security constraints into the design to provide a detective control mechanism against IP piracy. This results in the generation of protected fault-secured DSP designs against IP piracy, ensuring the safety of end consumers against pirated and unreliable designs

by isolating them before integration into the system-on-chips of consumer electronics (CE) systems.

Overall, both methodologies address the critical issue of IP piracy and ownership infringement that sabotage the revenue and reputation of genuine IP vendors. The proposed techniques provide a higher level of security with a low probability of coincidence and high tamper tolerance without incurring significant design cost overhead. These methodologies pave the way for more robust and secure IP designs, thereby ensuring the safety and security of end consumers.

LIST OF PUBLICATIONS

- Mahendra Rathor, Aditya Anshul, **K Bharath**, Rahul Chaurasia, Anirban Sengupta, "Quadruple Phase Watermarking during High-Level Synthesis for Securing Reusable Hardware IP Cores", ***Elsevier Journal on Computers and Electrical Engineering***, Volume 105, January 2023, 108476

- Aditya Anshul, **K Bharath**, Anirban Sengupta**, "**Designing a Low Cost Secured DSP Core Using PSO-DSE Driven Steganography for CE Systems", ***Proceedings of 8th IEEE International Symposium on Smart Electronic Systems (IEEE – iSES)***, India, Dec 2022, pp. 95-100, doi: 10.1109/iSES54909.2022.00030

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

P_i	—	Partition of the Scheduled Data Flow Graph (SDFG)
S_i	—	Signature generated from the partition P_i
HD_i	—	Hash digest.
V_i	—	Vendor
σ_i	—	Control step in the SDFG
Ti	—	Storage variables
Ri	—	Registers
A_i^j	—	Adder of *j*th vendor
M_i^j	—	multiplier of *j*th vendor
T_c	—	Transient fault strength
R_c	—	Resource constraints
XR_a	—	'X' represents the number of hardware units and 'a' represents the type of hardware resource.
N_{OG}	—	Original unit
N_{DP}	—	Duplicate unit
Ci	—	Control step in SDFG
Vi	—	Storage variables
CN	—	Crossing number
P_c	—	Probability of coincidence
T^p	—	Tamper tolerance
C_t	—	Design cost
A_H	—	Design area
L_h	—	Design latency
Pb_c	—	Probability of coincidence of the fault secured DSP
TT	—	Tamper tolerance of the fault-secured DSP design
$D_c(s_n^t)$	—	Design cost of the fault-secured DSP design

ACRONYMS

DSPs — Digital Signal Processors

DSP — Digital Signal Processing

VHDL — Very High-Speed Integrated Circuit Hardware Description Language

FPGAs — Field Programmable Gate Arrays

ASICs — Application-Specific-Integrated-Circuits

CPU — Central Processing Unit

IP — Intellectual Property

HLS — High-Level Synthesis

IoT — Internet of Things

SoC — System-on-Chip

IP-XACT — Intellectual Property Core Provider's Group

SIP — Silicon Intellectual Property

CE — Consumer Electronics

CFE — Computational Forensic Engineering

HDL — Hardware Description Language

RTL — Register-Transfer Level

DFG — Data Flow Graph

CFG — Control Flow Graph

CDFG — Control/Data Flow Graph

DSE — Design Space Exploration

PUFs — Physically Unclonable Functions

CED — Concurrent Error Detection

DMR — Dual Modular Redundancy

EM — Electromagnetic

FU — Functional Unit

IC — Integrated Circuit

SDFG — Scheduled Data Flow Graph

FIR — Finite Impulse Response

ET	—	Encoding Tree
Opn	—	Operation
CIG	—	Coloured Interval Graph
SEU	—	Single-Event Upsets
IDCT	—	Inverse Discrete Cosine Transform
FFT	—	Fast Fourier Transform
CN	—	Crossing Number
DCT	—	Discrete Cosine Transform
JPEG	—	Joint Photographic Experts Group
IIR	—	Infinite Impulse Response
MPEG	—	Moving Picture Experts Group

Chapter 1

Introduction

1.1 Overview

Digital Signal Processors (DSPs) are specialized processors designed to perform mathematical operations on real-time signals quickly and efficiently, typically used in various real-time applications. DSPs enhance sound and speech quality in audio and speech processing, while in telecommunications, they process signals for communication systems; in RADAR, LIDAR, and sensors, DSPs are used to process signals for various applications such as navigation, mapping, and object detection. In image and visual processing, they are used to enhance the image and video quality, while in neural network processing, they are used to perform real-time inferencing of deep learning models. DSPs can be found in a variety of devices, ranging from consumer electronics such as mobile phones to satellites and military communications. The widespread use of DSP technology highlights its importance in enabling advanced capabilities in consumer electronics. The first DSP was created by Texas Instruments and was famously used in the child's toy "The Speak & Spell" in the late 1970s. The DSP in the toy was used for speech processing, which allowed the toy to recognise and produce speech sounds. This marked the beginning of the widespread use of DSPs in consumer electronics, and since then, DSPs have become an essential component in many different fields and applications.

Digital Signal Processing (DSP) algorithms are crucial for a variety of applications such as image and audio processing, compression, and de-noising, as mentioned earlier. These algorithms require high performance and low power consumption, which can be achieved through hardware

acceleration. One way to achieve hardware acceleration is by using reusable intellectual property (IP) cores. An IP core, or Intellectual property core is a pre-designed and pre-verified block of digital logic that can be easily integrated into a larger system to perform specific functions. IP cores are designed to be reused in multiple applications, allowing designers to save time and resources while achieving high performance and low power consumption. Reusable IP cores are typically designed in a standard digital design language, such as Very High-Speed Integrated Circuit Hardware Description Language (VHDL) or Verilog, and can be implemented on Field Programmable Gate Arrays (FPGAs) or Application-Specific-Integrated-Circuits (ASICs).

The CPU (Central Processing Unit) and a reusable DSP IP Core are both components used in digital signal processing (DSP) applications, but they have some key differences. A CPU is a computers main processor that executes instructions and performs data processing. It is designed to perform a wide range of tasks and can be programmed to perform DSP algorithms, but it is not optimized for DSP processing. The CPU is a general-purpose processor that can handle a wide range of tasks, but its processing speed and power consumption for DSP algorithms can be relatively slow compared to dedicated DSP hardware. On the other hand, a reusable DSP IP Core is a predesigned and pre-verified block of digital logic that is optimized specifically for DSP processing. It can be integrated into larger systems and provides faster processing speeds and lower power consumption compared to CPU-based implementation of the same DSP algorithm. A reusable DSP IP Core is designed to perform a specific DSP function and can be optimized for high performance and low power consumption, making it a better choice for DSP applications than a general-purpose CPU.

Reusable IP cores offer several advantages over traditional software-based implementations of DSP algorithms. In hardware, they can be implemented in parallel, which eliminates the overhead of software-based processing. IP cores are optimized for low power consumption, which is important for battery-powered devices or applications that need to minimize power consumption. Another advantage of reusable IP Cores is their ease of integration into larger systems. IP Cores can be used as building blocks for larger systems, reducing design time and increasing design reliability. This allows designers to focus on the overall system design rather than on implementing individual DSP algorithms. With the growing demand for high-performance and low-power computing systems, reusable IP cores will likely become increasingly widespread in the future.

1.2 Evolution of IP Core Design

The evolution of IP (Intellectual Property) core design has been driven by the increasing complexity of integrated circuits and the need for more efficient and cost-effective design methods. IP Core design is a methodology for creating reusable blocks of digital logic that can be used in the design of integrated circuits. Over the past several decades, IP Core design has evolved to meet the changing needs of chip designers and the wider electronics industry. The early days of IP Core design were characterized by the use of hardware description languages such as VHDL and Verilog. Designers would use these languages to create digital logic circuits from scratch, using a combination of manual design and simulation tools. While this approach was effective for relatively simple circuits, it became increasingly time-consuming and complex as integrated circuits became more complex. In response to these challenges, IP Core design began to evolve towards a more automated and efficient approach. The introduction of High-Level Synthesis (HLS) tools allowed designers to describe the functionality of a digital logic circuit using a high-level

programming language, such as C or C++, rather than hardware description languages. This reduced the time and effort required to create digital logic and improved the quality and reliability of the final product, as HLS tools used advanced algorithms to generate optimized digital logic.

The increasing importance of low-power and energy-efficient design has also played a key role in the evolution of IP Core design. Many IP Cores are now designed specifically to minimize power consumption, and their use in a design can help reduce the system's overall power consumption. This is especially important for battery-powered mobile devices, where power consumption is a critical consideration. The rise of the Internet of Things (IoT) and the increasing demand for connected devices have led to a growth in the use of IP Cores for embedded systems. IP cores are being used to address the need for high performance, low area, minimum cost, and timely operation in many embedded systems, especially in mobile phones, where low power consumption and high performance are critical requirements. Another key development in IP Core design was the introduction of platform-based design. Platform-based design enables designers to reuse common components and systems across multiple applications, reducing development time and cost and improving the quality and reliability of the final product. This approach also allows for easier integration of IP Cores from different sources, making it easier for designers to access a wide range of high-quality, reusable IP Cores.

Further, the growing demand for IP Cores led to the development of IP Core libraries that could be easily integrated into a larger System-on-Chip (SoC) design. These libraries allowed designers to quickly and easily access a range of IP Cores, speeding up the design process and reducing the time to market for a product. The development of IP Core libraries was a significant milestone in the evolution of IP Core design, as it allowed designers to take advantage of pre-existing IP Cores and focus on the

overall system design. The evolution of IP Core design has also led to the development of IP Core standards, such as the Intellectual Property Core Provider's Group (IP-XACT) standards. IP-XACT provides a standardized format for describing IP Cores, making it easier for designers to integrate IP Cores from different vendors into a larger SoC design. The standardization of IP cores has also enabled the creation of IP Core exchange platforms, such as the Silicon Intellectual Property (SIP) Core Exchange, where designers can easily access and compare IP Core from different vendors.

1.3 The Advantages of Third-Party IP Core Suppliers

IP Cores are supplied by third-party IP vendors for several reasons:

1. Specialist expertise: IP vendors specialise in designing and developing specific IP Cores, allowing them to focus on the latest technological advancements and provide their customers with high-quality, reliable IP Cores.
2. Cost-effectiveness: Designing IP cores from scratch is a time-consuming and resource-intensive process. By outsourcing this work to third-party IP vendors, companies can take advantage of the lower costs associated with specialised design teams and reduce the time and effort required to bring a product to market.
3. Time-to-market: IP vendors have a pre-existing library of IP Cores, which can be used to speed up the development process and reduce the time to market for a product.
4. Risk reduction: IP Cores are thoroughly tested and validated before they are made available to customers, reducing the risk of design bugs and improving the quality and reliability of the final product.
5. Scalability: IP vendors have the resources and expertise to scale up the production of IP cores as demand increases, enabling customers to take advantage of economies of scale and reduce costs.

In summary, IP Cores are supplied by third-party IP vendors because they offer a cost-effective solution that reduces design time and risk, and improves quality while allowing companies to focus on the overall system design. By outsourcing IP Core development to specialist vendors, companies can take advantage of the latest technology advancements and bring their products to market more quickly.

1.4 Security Concerns and Piracy Issues of Reusable IP Cores in the Global SoC Supply Chain

The integration of Intellectual Property (IP) Cores in System-on-Chip (SoC) [1] design has become a standard practice in the consumer electronics (CE) industry. The use of IP Cores supplied by third-party IP vendors [2] maximizes design productivity and minimizes design time. These hardware IP cores are designed to perform specific functions and are often reused in various electronic designs. The increasing reuse of IP Cores has brought to the forefront of the security concerns and IP piracy issues that arise from the global SoC supply chain. The reuse of IP Cores is driven by several benefits, including reduced development time and cost, improved design quality, and reduced risk. However, this increased reuse of IP cores leads to risks like copying and piracy. IP Cores can represent many man-years of design, research, and verification testing; therefore, it is essential to protect this investment. If IP Cores are not properly secured, they can easily be copied and used without authorization, resulting in significant financial losses for the original IP Core developer.

One of the primary security concerns in the global SoC supply chain is the potential for IP Core tampering. Tampering with IP Cores can take many forms, including unauthorized modifications, insertion of malicious code, or unauthorized copying. This can lead to significant

security breaches, particularly if the IP Cores are used in critical systems such as those found in the aerospace, defence, or medical industries. Another security concern in the global SoC supply chain is the potential for IP Core counterfeiting. Counterfeit IP Cores can be difficult to detect and can have serious consequences for the end user. Counterfeit IP Cores may not perform as intended, contain malicious code, or may not comply with industry standards. This can lead to significant safety and security risks and financial losses for the end user [2, 3, 4, 5].

Another major concern is the infringement of the licensing agreement through the utilization of the IP Core in multiple products with only a single license obtained. This is a common occurrence, as IP core providers often sub-license other IP Cores for inclusion in their designs, and once a design has completed testing and verification, it is tempting to reuse it in additional products. This poses a significant threat to the original IP owner, as their investment in the design and development of the core is not protected. In addition to the threat of license violations, there is also the risk of direct piracy, where fraudulent means or reverse engineering may allow the direct theft or copying of the IP for reuse without permission. In such cases, the adversary may even claim the IP to be their own, making it difficult for the original IP owner to prove ownership and protect their investment. Unauthorized duplication and distribution of IP Cores can lead to significant financial losses for the owners and undermine their competitive advantage in the market.

Intellectual Property (IP) Cores are critical components in the design of silicon chips and play a crucial role in the electronics industry. These cores contain valuable technology, trade secrets, and propriety information that are the result of extensive research and development efforts. Protecting IP Cores from securing concerns and piracy issues is

vital to ensure companies' financial stability and competitiveness in the chip industry.

1.5 Safeguarding Intellectual Property: An Examination of Available IP Protection Methods

The use of IP protection mechanisms is critical in ensuring the protection of IP Cores from security concerns and piracy issues. The different protection mechanisms provide different levels of protection based on the design abstraction levels. A comprehensive IP protection strategy can be achieved by combining these mechanisms. Understanding the different protection mechanisms and their benefits is crucial for companies in the chip design industry, as it can help them to better protect their valuable IP assets. Some well-known IP Protection mechanisms widely used in various consumer electronics (CE) products include watermarking, IP metering, Computational Forensic Engineering (CFE), and patents and copyrights.

Watermarking

The insertion of additional watermarking constraints is a widely used method for protecting Intellectual Property (IP) Cores in recent years. This method is implemented during the architectural synthesis stage of IP design, specifically in the register allocation or scheduling step. In this process, a coloured interval graph is used to represent the storage variables and their overlapping lifetimes. By adding additional edges between the nodes of the graph as watermarking constraints, the storage variable is forced to be stored in distinct registers, thus increasing the security of the signature. The watermarking scheme requires a signature detection process which is done in two steps: reverse engineering and signature verification. Reverse engineering involves obtaining a sample of the product suspected of using the IP illegally and sending it to a specialist laboratory for analysis and reverse engineering. Signature verification involves comparing the

detected signature with the original signature to confirm the presence of the IP in the product. This method of IP protection offers a secure and reliable way to protect the IP from security concerns and piracy issues, ensuring that the owner's rights are protected.

IP Metering

IP metering is a technique used by IP vendors to control and monitor the usage of their intellectual property (IP) Cores. IP metering aims to ensure that IP vendors receive fair compensation for their work and to prevent the unauthorized or illegal use of their IP. IP metering is performed by assigning a unique identifier to each unit of the IP Core. This identifier can be created through a variety of methods, including different configurations during architectural synthesis or programmable hardware elements. The unique identifier acts as a meter that records the usage of the IP and enables the IP vendor to enforce royalties for each unit sold. Hardware and software metering are techniques used to protect IP cores from piracy and illegal use. Hardware metering is employed in situations where the design company does not have control over the number of copies being made by the silicon foundry. In the case of software IP Core vendors, the number of uses of the soft core can be metered to ensure that the user is not making unauthorized copies. This is achieved through the use of hardware/software locks and license agreements. Hardware metering is performed by making a small portion of the design programmable during configuration time. This small portion is configured in a unique way for each manufactured chip, allowing the manufacturer to determine the number of units (or batches of units) produced. On the other hand, software metering involves tracking the number of uses of the software IP Core by the user through the use of license agreements.

IP metering is an important tool for IP protection, as it helps to ensure that IP vendors are properly compensated for their work. This not

only protects their investment in the development of the IP Core but also incentivizes further innovation and investment in the field. In addition to enforcing royalties, IP metering also provides IP vendors with a level of control and monitoring over the usage of their IP. This helps to prevent piracy and unauthorized use, ensuring that IP vendors' rights are protected and safeguarding their investment in the development of the IP Core.

Computational Forensic Engineering (CFE)

Computational Forensic Engineering (CFE) involves the collection of features and statistics of a given IP design, which can be analyzed to determine the likelihood of a specific entity having created it. In the next phase, the collected features and statistics are extracted to determine the unique characteristics of the design. The extracted features are then clustered and compared to a pool of algorithms used to solve the same optimization problem to identify the algorithm that has been used to create the IP design. Finally, the results are validated to ensure the accuracy and reliability of the findings. The use of CFE helps in identifying the entity responsible for creating a particular IP design, thereby providing a way to enforce IP rights and prevent piracy issues.

Patent and Copyright

A patent serves as a form of intellectual property protection that gives the inventor the exclusive right to prevent others from making, using, selling, and importing the patented invention for a specified number of years. This helps the inventor to protect their innovations and prevent others from profiting from their work without their permission. Obtaining a patent requires a thorough examination process, including a search for prior art, to determine the novelty and non-obviousness of the invention. The process of obtaining a patent can be time-consuming and expensive, but it can also provide valuable protection and exclusivity for the inventor's ideas.

Copyright is a form of legal protection provided to creators of original works for authorship, such as litter, dramatic, musical, artistic, and certain other intellectual works. It gives the creator the exclusive right to control the use and distribution of the work for a limited period. Copyright protection applies to works that are fixed in a tangible form of expression, such as a book, a painting, or a software program. The owner of the copyright has the exclusive right to reproduce the work, distribute copies, and create derivative works based on the original. Infringement of copyright can lead to legal action under civil law.

1.6 Background on High-Level Synthesis

High-level synthesis (HLS) [2], [5] is the process of automatically translating a high-level hardware description language (HDL) specification into a register-transfer level (RTL) description, which can be used to implement the design on a specific hardware platform. The goal of HLS is to reduce the time and effort required to design complex hardware systems by allowing designers to describe the system at a higher level of abstraction and then automatically generate the low-level hardware implementation. HLS has become increasingly important in the field of digital design as the complexity of modern hardware systems has grown dramatically. The use of HLS can significantly reduce the time-to-market and development costs of such systems by enabling designers to quickly explore and evaluate a large number of design alternatives and optimize their designs for different performance metrics such as power consumption, area, and latency.

Design Entry Phase:

The design entry phase is the initial stage of the high-level synthesis process. In this phase, the designer defines the high-level system specification, which consists of the functional behaviour of the system and

the constraints on the system's resources. The input to this phase is a high-level language description of the system's behaviour, such as C or MATLAB, and the constraints on the system's resources, such as the area, power, and execution time. The design entry phase begins with the conversion of the high-level language description into a data flow graph (DFG). The DFG represents the data dependencies among the system's operations. Each node in the DFG represents an operation, and the edges represent the data dependencies between the operations. The DFG provides a high-level representation of the system's behaviour.

In addition to the DFG, the designer also creates a control flow graph (CFG) in the design entry phase. The CFG represents the control flow of the system, i.e., the sequence of operations executed by the system. The CFG provides a high-level representation of the system's control behaviour. The designer combines the DFG and CFG into a control/data flow graph (CDGF) in the design entry phase. The CDFG is a unified representation of the system's control and data flow behaviours. The CDFG consists of nodes that represent operations and edges that represent both data and control dependencies.

High-level design phase

The high-level design phase is a critical stage in the high-level synthesis, where a behavioural description of the system is transformed into an optimized register transfer level (RTL) design. The objective of the high-level design phase is to determine the most efficient way to implement the functionality of the system while satisfying the constraints specified by the user. During this phase, the system is modelled as a set of data flow graphs (DFGs) and control flow graphs (GFGs) that capture the computational and control aspects of the system. These graphs are used to analyze the system and identify the optimal way to implement the

functionality of the system. The high-level design phase can be divided into three main steps: scheduling, resource allocation, and binding.

- **Scheduling:**

The scheduling step determines the order in which the operations in the DFG will be executed. The goal of scheduling is to minimise the number of clock cycles required to execute the operations while satisfying any timing constraints specified by the user.

- **Resource allocation:**

Resource allocation is the process of determining which hardware resources (such as functional units, registers, and buses) will be used to implement the operations in the DFG. The goal of resource allocation is to minimise the overall cost of the system while satisfying any resource constraints specified by the user.

- **Binding:**

Binding is the process of assigning each operation in the DFG to a specific hardware resource. The goal of binding is to minimise the critical path delay of the system while satisfying any timing constraints specified by the user.

In addition to these three steps, the high-level design phase may also include optimization techniques. Optimization techniques play a crucial role in High-level synthesis (HLS) to improve the quality of the synthesized hardware design. Optimization techniques help to achieve design objectives such as minimum area, maximum speed, and low power consumption while satirizing the constraints and goals of the design. Some of the commonly used optimization techniques in HLS are:

- **Design Space Exploration (DSE):** DSE is the process of exploring the design space of a system to find an optimal implementation that satisfies the design objectives such as minimum area, maximum speed, and low power consumption. DSE allows the designer to explore different trade-offs between design objectives and select the best possible solution.

- **Loop unrolling and Pipelining:** Loop unrolling is a technique used to improve the performance of a loop by executing multiple iterations of the loop in parallel. Pipelining is another technique used to improve the performance of a design by breaking it down into smaller stages and executing them in parallel. Both techniques help to improve the throughput of a design.
- **Data path optimisation:** Data path optimisation is the process of optimising the data path of a design to improve its performance. It involves optimising the number and type of functional units used, the number and type of registers used, and the interconnect between the functional units and registers.
- **Control path optimisation:** Control path optimisation is the process of optimising the control path of a design to improve its performance. It involves optimising the control logic used to generate the control signals that drive the functional units and registers.
- **Power optimisation:** Power optimisation is the process of optimising the power consumption of a design. It involves minimising the dynamic power consumption by reducing the switching activity in the design and minimising the static power consumption by reducing leakage currents.

Overall, the high-level design phase is a critical step in the high-level synthesis that determines the optimal way to implement the functionality of the system while satisfying the constraint specified by the user. By using a combination of scheduling, resource allocation, binding, and optimization techniques, high-level synthesis tools can produce optimized RTL designs that meet the performance, power, and area requirements of the system.

RTL generation phase:

RTL (Register Transfer Level) generation is the final phase of the High-level synthesis process, where the synthesized hardware design is

transformed into an RTL implementation. The RTL implementation is a low-level hardware description that can be used to generate a physical implementation of the design. In this phase, the control and data path structures of the design are synthesized and integrated to produce a complete RTL description. The RTL generation process involves the conversion of the synthesized CDFG (Control/Data flow graph) to RTL-level structural description. The CDFG contains all the information about the design, including the operation, data dependencies, control flow, and resource allocation information. This information is used to generate an RTL description that is compatible with the target technology and the design constraints. The RTL implementation is then verified using simulation and synthesis tools.

The RTL generation process typically involves the following steps:

1. **Datapath and Control Path Synthesis:** In this step, the hardware resources such as registers, memories, and arithmetic units required for the design are identified and allocated. The datapath and control path structures are then synthesised by mapping the operations of the CDFG to the hardware resources.
2. **RTL Netlist Generation:** Once the datapath and control path structures are synthesised, an RTL netlist is generated that describes the hardware implementation of the design. The RTL netlist is a structural description of the design that includes information about the hardware components, their connectivity, and the timing constraints.
3. **Verification:** The RTL implementation is then verified using simulation and synthesis tools. The simulation is done to verify the correctness of the design functionality and the timing constraints. Synthesis tools are used to check the design against the target technology libraries and constraints.
4. **Optimisation:** Finally, the RTL implementation is optimized to improve its performance, power consumption, and area utilisation.

Various optimisation techniques such as logic restructuring, clock gating, and retiming are applied to the RTL implementation to improve its efficiency.

The RTL generation phase is a critical step in the High-level synthesis process as it provides a complete hardware implementation of the design that can be used for further verification, testing, and physical implementation. The accuracy and quality of the RTL implementation have a significant impact on the final performance, power consumption, and area utilization of the design. Therefore, it is important to ensure that the RTL implementation is optimized, verified, and meets all the design requirements before proceeding to the physical implementation phase.

1.7 Organisation of the Thesis

This thesis is organised into eight chapters. Chapter 2 describes the related works regarding the proposed approaches, chapter 3 and 4 discusses the proposed quadruple-phase watermarking methodology, chapter 5 and 6 discuss the proposed unified biometrics with an encoded dictionary for hardware security of fault-secured IP core designs, chapter 7 presents the results of the proposed methodologies, demonstrating a significant decrease in the probability of coincidence and a higher level of tamper tolerance compared to previous techniques, without incurring significant design cost overhead. Finally, chapter 8 concludes the thesis, summarising the proposed methodologies' contributions and their impact on the field of IP security in chip designs.

Chapter 2

Review of Past Work and Problem Formulation

Hardware IP watermarking techniques have been a popular form of IP protection techniques for securing data-intensive hardware co-processors used in consumer electronics-based industry, but their (earlier approaches) effectiveness depends on various factors such as the type of watermarking technique used, and the potential attacks that the watermarking technique can withstand. Therefore, the development of effective hardware IP watermarking techniques requires a clear understanding of the strengths and limitations of existing techniques and the identification of potential vulnerabilities or attacks that could compromise their effectiveness. In this section, we highlight the need for further research into the development of hardware IP watermarking techniques to enable the effective protection of valuable IP assets in the chip design industry against hardware security threats.

Prior works:

Various hardware security techniques have been proposed for protecting combinational/sequential circuits and complex DSP circuits using IP watermarking. One approach proposed by Cui *et al.* [6] employed a constraint-based watermarking scheme where closed cones are modulated to embed security constraints at the logic level. Another approach by Cui and Chang [7] employed template substitution-based watermarking, where specific cells are replaced with equivalent templates in the library. To protect combinational circuits, watermarking is usually employed during the combination logic synthesis phase of the design process. For securing sequential circuits, a watermarking scheme has been proposed where the output of transitions of the state transition graph is

used to embed signature bits [8]. Cellular automata-based FSM watermarking schemes have also been proposed by Karmakar and Chattopadhyay [9, 10] to secure IP cores. However, these watermarking schemes [6, 7, 8, 9, 10] have been proposed at the combinational/ sequential logic synthesis level and do not target the security of complex DSP circuits.

Other watermarking schemes [11-15, 16] have targeted the security of DSP circuits, including an approach proposed by Sengupta and Rathor [16], where a watermark is employed in a DSP circuit during the early floor planning stage at the physical level. Some of the watermarking approaches such as [41], [13] are utilised at the lower levels of abstraction, such as at the gate level. When operating at the gate level, a vendor signature may be incorporated into the design using either (i) the netlist and bit stream of an IP design as proposed in D. Ziener *et al*. [41], or (ii) during the in-synthesis process of design like approaches by Le Gal and Bossuet [13] implanted during the in-synthesis phase of the HLS process of DSP designs. However, since both of these approaches embed the signature at a lower level, they are not appropriate for complex DSP cores, and the insertion of the signature results in significant overhead on the system. As a result, alternative techniques have been developed that target insertion of the signature at a higher abstraction level such as the architecture level. For instance, a watermarking technique involves implanting a secret mark at algorithmic synthesis, which can be accomplished using various methods, including (i) multi-variable signature encoding rules for IP core protection [15], [42], (ii) multi-variable signature watermarking at three different phases of architectural synthesis (the scheduling phase, the hardware allocation phase, and the register allocation phase). Koushanfar *et al*. [13] and Hong and Potkonjak [14] embedded signatures during the register binding phase of the HLS process, and Sengupta and Bhadauria [15] secured DSP circuits using a four-

variable signature to embed watermark during the register biding phase of the HLS process. A seven-variable signature embedded during the three phases (scheduling, register binding and FU binding) of HLS was used by Sengupta *et al*. [12], and (iii) encoding the author's signature by adding a set of design and timing constraints to the design [11], [14]. Castillo *et al*. [40] introduced a technique for IP watermarking at the hardware description language (HDL) design level, aimed at safeguarding IP cores. The authors in [40] employed a secure signature extraction methodology integrated with minimal system modification in their approach. In their work, they also utilised a tool to discover diverse input patterns that yield the same output, and this is where the signature block is located. However, these watermarking schemes use a signature that is converted into security constraints using the designer's encoding rules, the goal of watermarking is thwarted when an attacker possesses knowledge of the selected signature and encoding rules.

In addition to the watermarking techniques, an IP core steganography scheme [17] has been proposed that embeds vendors' steno-constraints into the DSP design to secure them against IP piracy. However, these constraints are also replicable by the attacker. The proposed quadruple-phase watermarking approach overcomes this limitation by generating a robust author's signature through a novel mechanism of graph partitioning, eight-variable encoding using an encoding tree, and hashing. The signature is embedded during four different phases of the HLS process to achieve high-quality watermarking, with a low probability of coincidence, in contrast to the related approaches of securing DSP circuits. A qualitative comparison of the proposed approach with different existing techniques is presented in Chapter 7.

Prior methods used for securing hardware IP cores include IP watermarking [18, 10, 11], [7, 13, 19, 20, 21, 22, 8] stenography [17], [23]

hardware authentication using physically unclonable functions (PUFs) [24], [25] unimodal palmprint biometrics [26], unimodal facial biometrics [27], and unimodal fingerprint biometrics [28]. Rai *et al*. [18] used a hardware watermarking technique based on polymorphic inverter designs using reconfigurable technologies. Koushanfar *et al*. [11] presented a hardware watermarking technique that embeds the generated watermark signature into the design. Gal and Bossuet [13] presented an IP watermarking included in high-level synthesis based on mathematical relationships between numeric values. Shayan *et al*. [19] used a watermarking technique inspired by a stealthy hardware trojan. Kuai *et al*. [20] developed a combined locking and watermark gin technique based on finite-state machines. Kean *et al*. [21] presented the approach of the embedding watermark by creating specific electromagnetic (EM) information. Becker *et al*. [22] presented a side-channel-based watermark gin that relies on side-channel information to embed the watermark.

To provide multi-cycle transient fault resiliency at the behavioural level, some authors in [43, 44, 45] have adopted a concurrent error detection (CED) approach. Specifically, they use dual modular redundancy (DMR) logic to duplicate the control data flow graph (CDFG) operations and impose specific hardware allocation rules to provide detection ability. However, the approach presented in [44] differs from that in [43, 45] in terms of advanced resiliency rules. In [43], at least two distinct hardware units are required for assignment to sister operations of the original and duplicate unit in DMR, whereas this is not necessary for [44]. In [44], even a single hardware module of a particular type can provide transient fault resiliency, making the approach more robust and cost-effective. Multiple transient faults have received very little attention because they were rear in past technologies. The focus was only on memory, not hardware modules. However, approaches, such as [46], have focused on multiple transient faults using a simulation-based technique. Specifically, [46] used

simulation to estimate the size of multiple transients resulting from a single radiation strike and their impact on the gate output for different gate input combinations. Furthermore, [47] focused on modelling transient fault propagation once a fault occurs at the gate output inside a logic circuit. The proposed fault-secured design in the unified biometric hardware security approach simultaneously tackles multi-cycle transient and multi-transient fault resiliency at a higher behavioural/architectural level.

Additionally, Sengupta and Rathor [17] presented a steganography approach that generates the steno-mark based on secret design data, encoding rule, and chosen threshold value to be embedded into the design. Rathor and Sengupta [23] presented hardware steganography using switch-based key-driven hash chaining. However, all of these methods are vulnerable to an adversary such as, in the case of watermarking, if an adversary manages to access the decoding combination of encoding digits, signature size, and encoding rule, they can easily replicate and reuse it to evade the IP piracy detection process. Similarly, in the case of steganography, if an adversary manages to decode the entropy threshold, stage keys, and encoding rule, they can also evade IP piracy detection by replicating the stego-mark. The proposed unified biometric-driven hardware security methodology, on the other hand, uses a unified biometric-driven encoded signature to incapacitate an adversary, unlike prior works which have only used a secret signature scheme. Moreover, none of the previous methods exploited the expandable encoded dictionary technique on top of unified biometric-driven hardware security methodology to enhance the security of IP cores, unlike the proposed work. Additionally, methods based on PUFs have been suggested by Zalivaka *et al*. [24] and Lao *et al*. [25] for the authentication of IP. These methods provide a security primitive for FPGA/system-on-chip bitstream and device authentication. Although these works have demonstrated their efficiency against such devices, they do not focus on the security of DSP

cores against IP piracy and false claims of ownership, unlike the proposed methodology.

Sengupta *et al*. [26, 27, 28] introduced biometric-based methods that use unique biometric features to create a digital signature. For instance, a contact-based high-resolution palmprint image acquisition system is presented in [48], a palmprint feature generation and expatriation using DSP algorithms and principal component analysis is presented in [49], and a multimodal palm biometric system was implemented on FPGA [50]. Furthermore, a high-resolution palmprint authentication system based on the pore feature was presented in [51]. Although these palmprint biometric approaches [48, 49, 50, 51] are used for the identification/ recognition of persons during authentication, however, palmprint biometrics has never been employed for the security of DSP cores so far.

Additionally, some approaches like [52] involve cryptography to encrypt palmprint, face and signature images using advanced hill cypher techniques or analyse features present in palmprint and palm vein images using contourlet transform [53]. While cryptographic digital signature-based techniques (such as those proposed in [54]) are effective, there exist some differences between the proposed unified biometric approaches when compared with cryptographic digital signatures, such as the generation process of cryptographic digital signatures [54] is complex and involves several steps, making it cumbersome.

In contrast, the proposed unified biometric approach is simple yet highly secure as they rely on natural biometric features to provide uniqueness and also the encoded expandable dictionary, without the need for complex security-enhancing steps in between and also the process of generating cryptographic digital signatures [54] relies on a casing algorithm that involves multiple intermediate steps to produce a hash or

digest. This algorithm requires knowledge and storage of multiple hash buffers and additive constants, as well as complex word computation functions, and round computation functions (including condition, rotation, summation, and majority functions), all of which contribute to the complexity of the process. In contrast, the proposed unified biometrics approach provides uniquely secure constraints with minimal complexity. In the proposed unified biometric approach, in the case of the palmprint approach, the palm image is divided into a specific grid size, and nodal points are created based on the palm features. The final signature is generated by concatenating the palm features. Similarly, the facial biometric approach generates facial nodal points and concatenates them to form the facial signature. The fingerprint biometric approach preprocesses the captured fingerprint impression to extract minutiae points, and then combines the coordinates of minutiae points, crossing number value, and angle magnitude to generate the fingerprint signature.

However, these approaches do not provide protection against IP piracy for fault-secured DSP design, unlike the proposed unified biometric approach. Moreover, the proposed unified biometric approach combines palmprint, facial and fingerprint biometrics to create a unified biometrics signature for embedding into the design. Our proposed methodology utilises the expandable encoded dictionary technique to achieve enhanced security. We can tailor the proposed unified biometrics signature to select the biometric signature strength and combination. This offers several times higher security with a lower probability of coincidence and higher tamper tolerance than recent state-of-art approaches. Therefore, our proposed approach provides robust security for fault-secured designs with minimal design cost overhead.

Chapter 3

Quadruple phase watermarking during high-level synthesis for securing reusable hardware intellectual property cores

The watermarking approach is a robust hardware security technique to protect IP cores from hardware threats like IP counterfeiting, cloning, and ownership infringement. Watermarking refers to the process of embedding a unique signature, also known as a watermark, into the design of the IP core. The signature serves as a way to identify the IP core's origin and authenticity and can detect unauthorised copies or modifications. In these watermarking approaches [11, 12, 13, 14, 15] for securing IP cores, the designer or vendor usually determines the signature and its encoding rules. The signature is then transformed into security constraints based on the encoding rules provided. Nevertheless, if the signature and encoding rules are compromised by an adversary, the watermark becomes vulnerable to attacks and can no longer provide the intended level of security against hardware security threats. Given this situation, the adversary can fraudulently claim IP ownership or may try to evade the IP counterfeit detection process. This limitation of watermarking approaches, where the signature can be compromised by an adversary, can be overcome by generating the signature rather than using a signature directly provided by the IP vendor. By using a robust process to generate the signature, the watermark can be made more secure and resistant to attacks, providing a higher level of protection for the IP core and this would hinder the attacker’s malicious effort of decoding the signature and claiming it for wrong purposes such as IP piracy and claiming IP ownership.

We proposed a novel watermarking technique for DSP-based IP cores where the signature is generated through a robust process and covertly embedded into the design during the four phases of the high-level synthesis (HLS) process viz. Scheduling, Register Binding, Functional Unit (FU) binding, and Interconnect binding. By embedding the signature into the design during the four phases of the HLS process, the watermark becomes more resistant to attacks and also ensures that the signature is not only present in the design but also deeply ingrained in the internal workings of the IP Core, making it more difficult for an adversary to exactly reproduce the signature.

Threat Model:

The increasing use of reusable hardware IP cores in IC design flow has made them susceptible to the threats such as IP piracy and fraudulent claim of IP ownership. In the case of IP piracy, an adversary may illegally pirate or imitate the hardware IP core without the knowledge and consent of the original IP vendor or designer [29, 17]. This type of piracy can occur in various scenarios, but one common situation is when a third-party design house is contracted to develop a design on behalf of a client, the client may provide the design house with proprietary information, such as the functional description of the IP core, and expect that the design house will keep the information confidential and use it only for the intended purpose. However, an adversary within the third-party design house may attempt to use the proprietary information for their benefit, such as by copying the design of the IP core and selling it to others without the knowledge or consent of the original IP vendor or designer. This can result in financial losses for the IP vendor or designer, as well as damage to their reputation. In the case of a fraudulent claim of IP ownership, an adversary may unlawfully claim ownership of the intellectual property (IP), despite not having any legal rights to the IP core [11]. For example, an adversary working for a third-party design house could claim ownership of an IP core

that they did not create or license, and then use that IP core in the development of a consumer product that competes with the original owner's product. The adversary may be motivated by a desire to profit from the product without having to pay royalties to the true owner of the IP core. Another example of a fraudulent IP ownership claim in the case of an IP core could be a situation where a competitor falsely claims that they own the IP core and sues the true owner for infringement of their IP rights. This type of scenario could result in the true owner of the IP core losing valuable time and resources defending against a frivolous lawsuit, potentially leading to financial losses and damage to their reputation.

A quadruple-phase IP watermarking scheme has been proposed to counteract the potential threats of IP piracy and fraudulent claims of IP ownership within the Integrated Circuit (IC) design flow process. By implanting the signature into the IP design during the High-Level Synthesis (HLS) process, the proposed scheme enhances the robustness of the watermark and provides a higher strength of ownership proof and also enhances the tamper tolerance of the watermark by deeply embedding the signature constraints into the IP design during the four phases of the HLS process: scheduling, register binding, functional unit binding, and interconnect binding.

3.1 Overview of the proposed approach

The proposed quadruple-phase watermarking approach is outlined in Fig. 3.1. Fig. 3.1 depicts the steps involved in generating and embedding a unique signature into the target DSP application. The proposed approach requires the following inputs such as (i) algorithmic representation of the target DSP application to be secured, (ii) designer-selected encoding tree, (iii) module library, (iv) resource constraints, and (v) mapping rules. Initially as shown in Fig. 3.1, the DSP application's algorithmic

representation is converted into an equivalent form of a data flow graph (DFG). This DFG is then scheduled and resource allocated using resource constraints and a module library provided by the designer. Then, the scheduled and resource-allocated data flow graph (SDFG) is divided into a specified number of partitions, denoted as 'N'. Further, in the proposed watermarking approach, the first partition (P_1) of the SDFG is encoded to create the signature (S_1), which is then embedded into the second partition (P_2) of the SDFG and later the encoding of the partition (P_2) with the embedded signature is used to generate the next signature (S_2) which is then embedded into the next partition (P_3). This process is repeated for subsequent partitions of the SDFG.

By using this chain-like process, the author's signature is generated and embedded into the design of the provided DSP application. The signature generation and embedding process details are discussed in this chapter and the next chapter respectively. To produce the *i*th signature from the *i*th partition of the SDFG, the partition P_i is converted into alphanumeric characters using the proposed encoding tree (ET). Later, these alphanumeric characters are given as input to the SHA-512 algorithm to generate the corresponding hash digest (HD_i). The resulting 512-bit hash is then truncated to the designer-specified size of the bitstream, which is used to create the signature S_i. The truncated bitstream is represented as 3-bit triads, with each triad representing a single digit in the author's signature. Using the combination of triads in the signature, each triad is mapped to its corresponding security (watermarking) constraints using the proposed eightfold mapping. This mapping allows the signature to be embedded into the design in such a way that it will be difficult to remove or modify the watermarking constraints without impacting the functionality of the design. The proposed mapping rules map signature triads (or signature digits) into four types of design constraints viz. scheduling, register biding, FU binding and interconnect binding. The constraints that

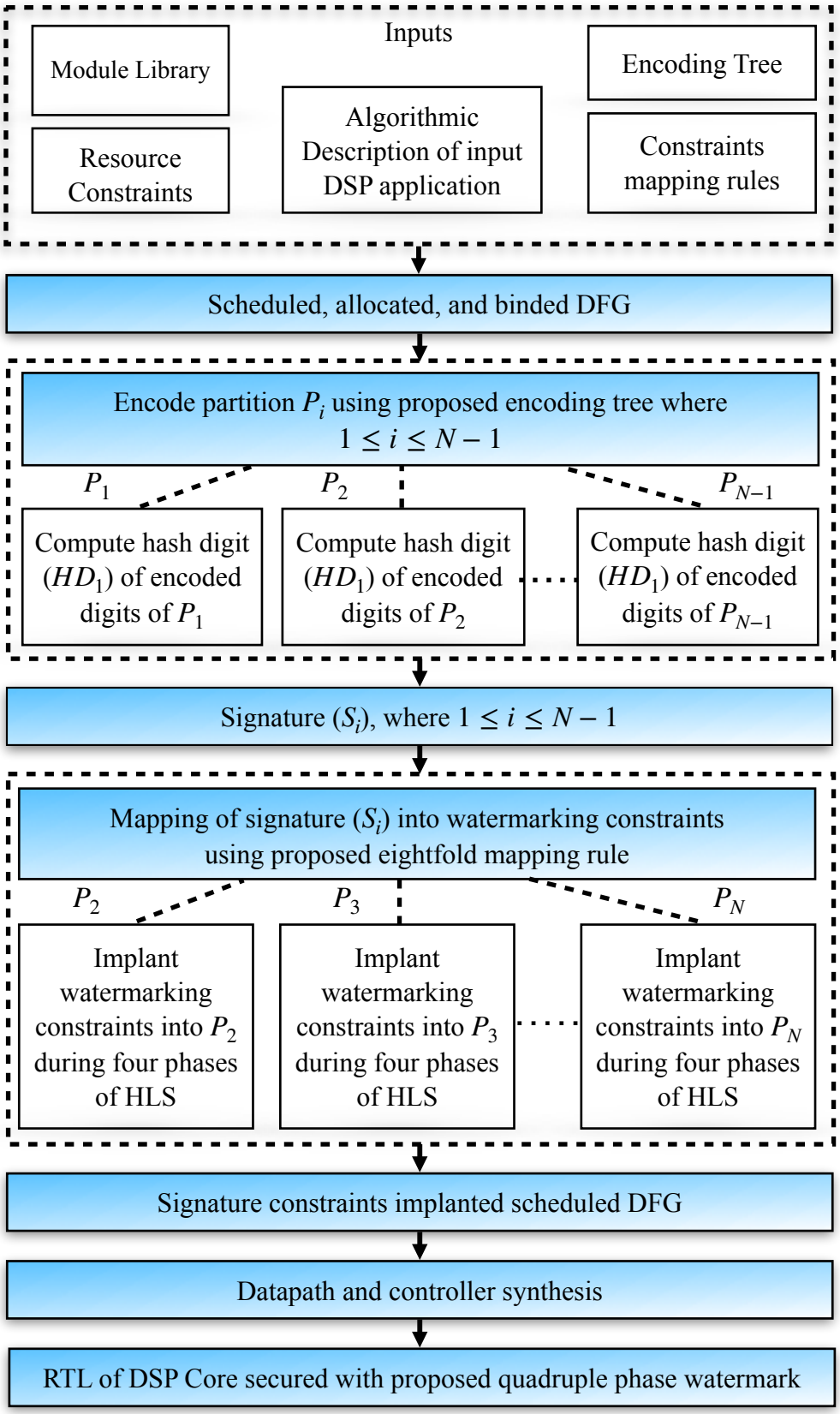


Fig. 3.1 Flow diagram of proposed quadruple phase watermarking approach

correspond to the *i*th signature are embedded into the (*i*+*1*)th partition of the SDFG during four phases of the High-Level Synthesis (HLS) design process. This process is repeated for all signature segments up to S_{N-1}, which is embedded into the Nth partition of the SDFG. After embedding the entire signature into the design during the HLS process, the datapath synthesis phase is executed to generate the Register Transfer Level (RTL) datapath with the embedded watermark.

The details of the proposed watermarking scheme are divided into two parts as shown in Fig. 3.2, the signature generation phase and the signature embedding phase. The signature generation phase will be discussed in this chapter and the signature embedding phase will be discussed in the next chapter.

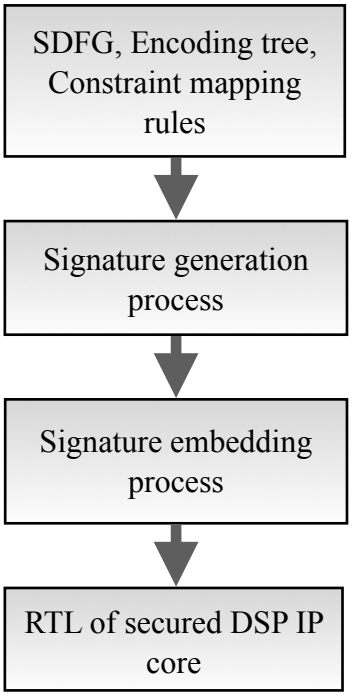


Fig. 3.2 Abstract view of the proposed quadruple-phase watermarking approach

The proposed quadruple-phase watermarking approach is explained thoroughly and demonstrated using an 8-point Finite Impulse Response

(FIR) core in separate subsections. The demonstration of the watermarking scheme on the FIR core serves to illustrate how the watermarking process can be applied to a specific design, providing a clear and tangible example of each step involved in the process. This approach helps to offer a more comprehensive and practical understanding of the proposed watermarking scheme.

3.2 Partitioning of Scheduled Data Flow Graph (SDGF)

In the proposed watermarking approach, an author's signature is generated by encoding a specific partition of the SDFG, referred to as the "*i*th" partition. The signature is then inserted into the next partition, which is the "(*i*+*1*)th" partition of the SDFG. The process is repeated for all partitions, with "*i*" varying from *1* to "*N-1*", where "*N"* represents the total number of partitions in the SDFG. For the partitioning of SDFG to be effective in watermarking, certain requirements must be satisfied, such as (i) the smallest possible partition should contain at least two connected nodes of the graph to facilitate more meaningful encoding and embedding of constraints, (ii) there should be a minimum of two partitions of the graph for the proposed approach to be applicable, (iii) the first partition P_1 should be the smallest, as the constraints are not embedded in this partition, but it is used to derive the signature for the subsequent partition, and finally (iv) the number of partitions should vary based on the size of the target application (in terms of the number of operations) to enable effective watermarking. It is important to note that the partitioning of the Scheduled Data Flow Graph (SDFG) is not in any way linked to the circuit partitioning. The motivation behind the partitioning of SDFG is to improve the robustness of watermarking.

The proposed method of partitioning the scheduled data flow graph plays a significant role in augmenting the strength of the signature. As a

result of the partitioning, the complexity of determining the exact signature is increased by a significant factor for an attacker, this is because an attacker would require the knowledge of the partition location, the total number of partitions of the SDFG, and the partition encoding to deduce the signature. In addition, the partition of the graph containing the embedded signature also participates in generating the next signature. This process makes the generated watermarking constraints highly robust, and it becomes challenging for an attacker to decode it.

The details of the SDFG partitioning mechanism on an FIR core are as follows, the scheduled data flow graph (SDFG) of FIR is scheduled from the DFG using the resource constraints of 3 adders and 2 multipliers as shown in Fig. 3.3. The integration of micro 3PIPs from various vendors is a common practice in the case of hardware IP core designs. In our demonstration, we utilise two vendors (V_1, V_2) to allocate Functional Units (FU) within the hardware IP core designs. Using the two vendors allocation scheme, the operations in the SDFG are allocated to the respective functional units as shown in Fig. 3.3. Adders are represented with the letter 'A', and the multipliers are represented with 'M'. The subscript for a functional unit represents the instance number, while the superscript represents the vendor number denoted as follows A_i^j or M_i^j, where 'i' represents the instance number, 'j' represents the vendor number of the functional unit. Registers R1-R8 are being utilised to execute storage variables T0-T30 within the design. The scheduled data flow graph SDFG that was generated has been separated into three partitions (P1, P2, and P3) based on the designer's choice, as illustrated in Fig. 3.3.

3.3 The proposed signature generation process

The proposed approach generates a final signature, which is a distinct representation made by combining several segments denoted as

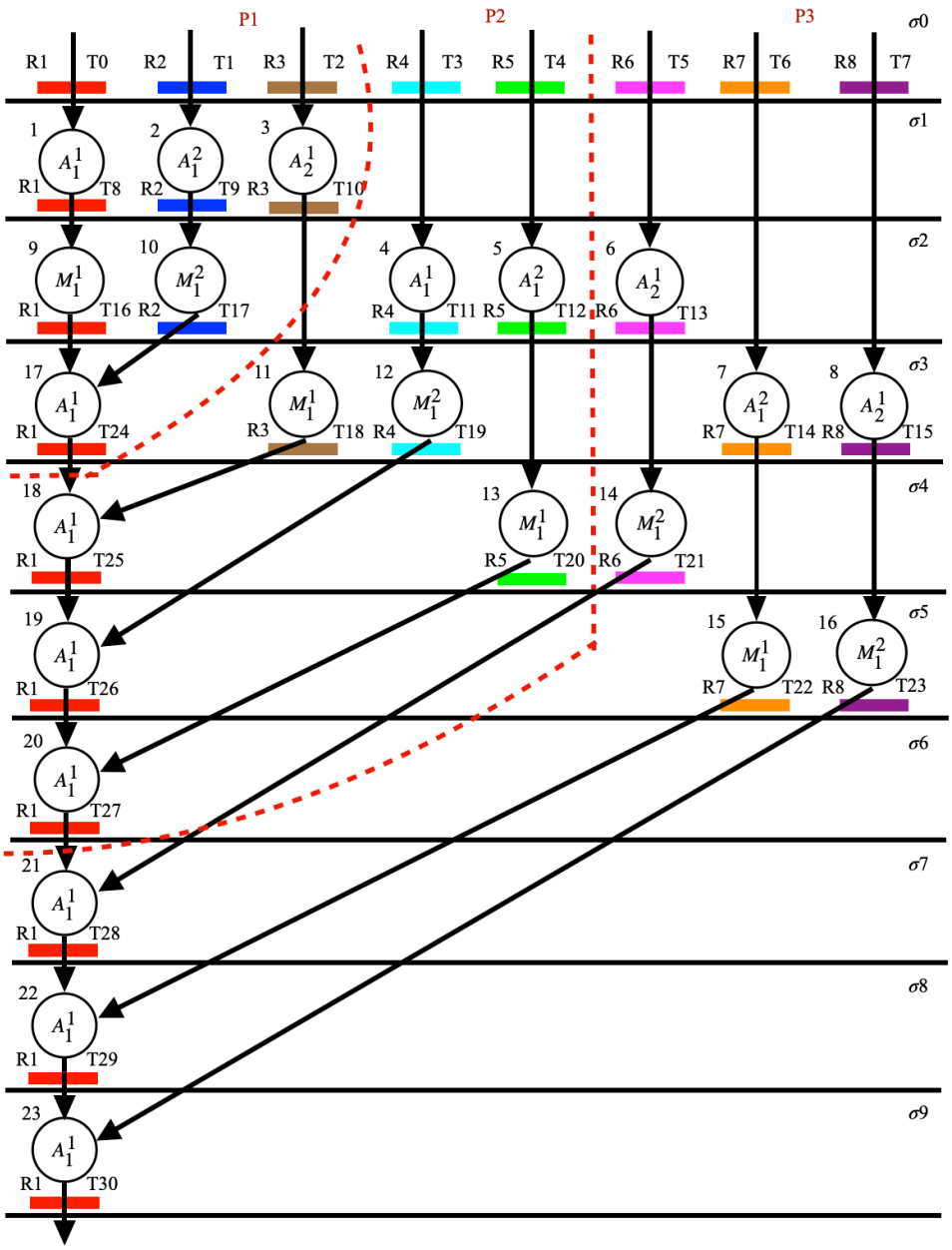


Fig. 3.3 Scheduled Data Flow Graph (SDFG) of FIR core with partitions P1, P2, and P3. (note: dashed lines indicate the partitions and the different colour bars indicate registers for primary and intermediate storage variables)

'S_1', 'S_2', 'S_3', and so on in a linked manner. The total number of partitions in the SDFG is denoted by N. Each segment contributes to the overall signature, with subsequent segments being generated in a chained fashion.

The process of generating the signature involves three generic steps. These steps include:

(1) Encoding of partitions using the proposed encoding tree (ET):

Each partition of a scheduled data flow graph (SDFG) of a DSP IP core is encoded into alphanumeric digits using the proposed encoding tree ET which is presented in Fig. 3.4. The proposed encoding tree has three levels. The nodes of the encoding tree in each level indicate various information associated with the operations in the design. At level 0 the root node is associated with the operation (opn) number. At level 1 there are two nodes, each indicating the control step and output register numbers, respectively. Level 2 consists of four nodes with two nodes associated with the input register number, and the other two with operation type, and vendor number respectively.

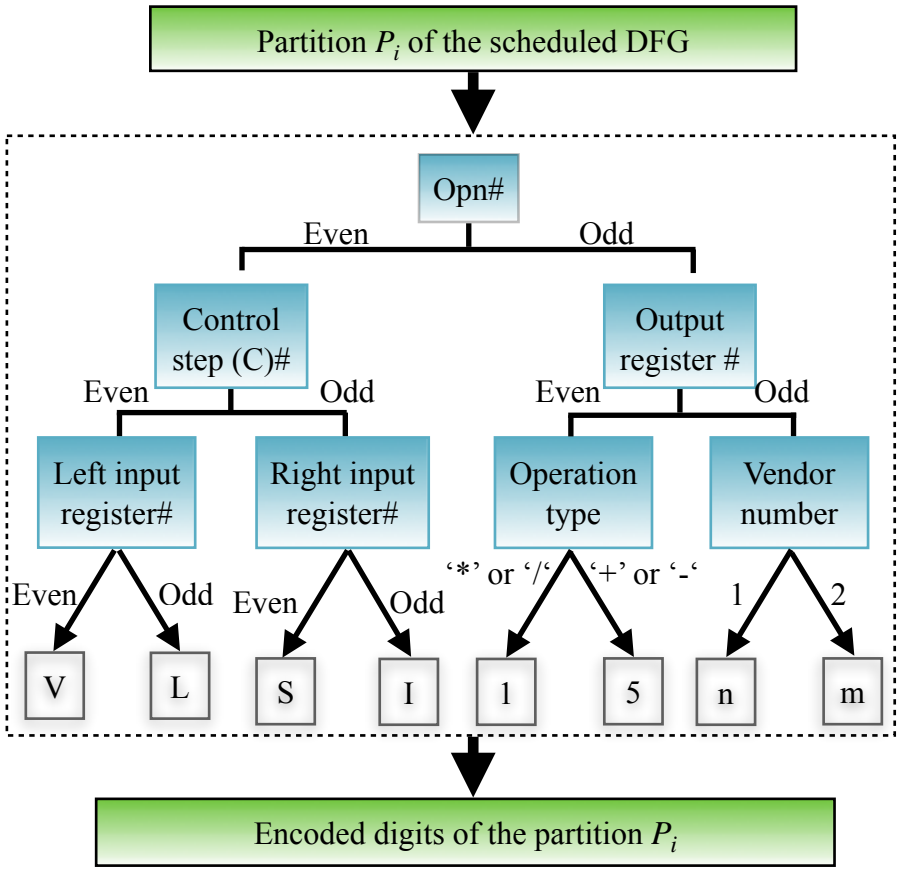


Fig. 3.4 Proposed encoding tree used for encoding partitions of SDFG

Finally, the last level of the encoding tree consists of leaves, which are alphanumeric digits chosen by the designer. To generate alphanumeric digits (encoded digits) for a partition of SDFG, each opn# in the given partition is traversed through the encoding tree. As each opn# is traversed through the encoding tree, the design information associated with the operation is used to determine which alphanumeric character it should be encoded into. The possible characters include { 'V', 'L', 'S', 'I', '1', '5', 'n', 'm,). The length of the encoding is determined by the number of opn# in the partition. Each opn# is encoded individually, resulting in a series of alphanumeric characters that together form the encoded number for that particular partition.

Below, we describe the encoding of the partition P_1 of SDFG (shown in Fig. 3.3) using the proposed encoding tree. The partition P_1 has six operations, so the length of the encoding will be six. By considering the first operation (opn #1), which has odd parity and odd output

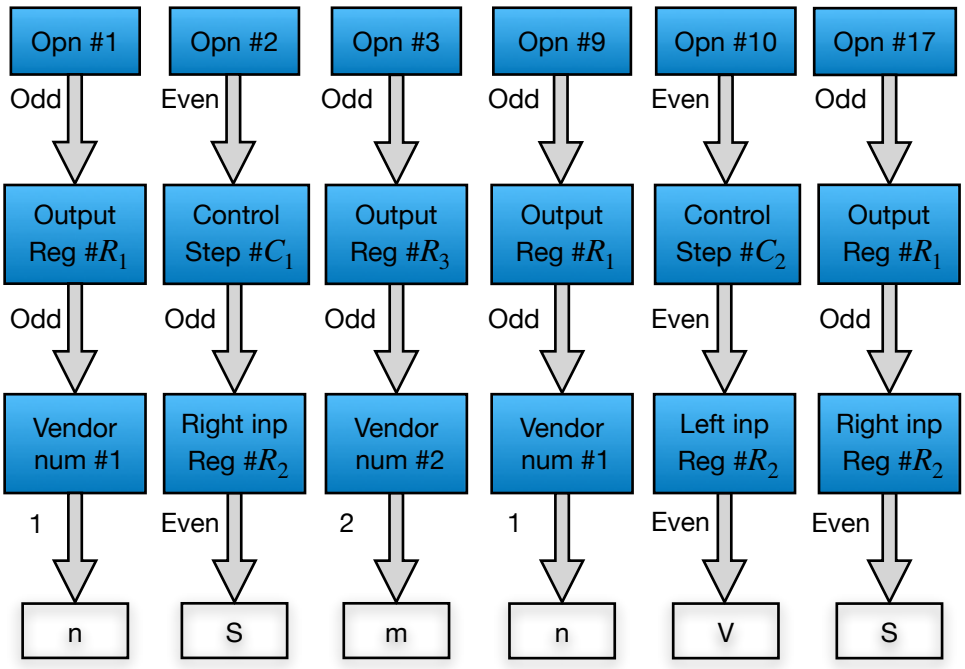


Fig. 3.5. Traversal details of operations in the partition P_1 of SDFG along the proposed encoding tree

register# (R1) and is assigned to vendor number 1. Hence it is encoded into 'n' through traversal of the encoding tree which is shown in Fig. 3.4, similarly, the second operation (opn #2) has even parity, and it is in control step #1 which is odd, and its right input register# (R2) is even. Hence it is encoded into "S" through the traversal of the encoding tree. The traversal details of all the operations of the partition P_1 are shown in Fig. 3.5. The final encoding of the partition P_1 of SDFG using the proposed encoding tree is "nSmnVS" as shown in Fig. 3.5.

(2) Calculating hash digest of encoded digits:

To calculate the hash digest (HD_i) of encoded digits, generated from the partition P_i, the SHA-512 hash function is used. First, the encoded digits of each partition P_i of SDFG are provided as input to the SHA-512 hash function, then the hash function will transform the encoded digits into the 512-bit hash digest HD_i. The final hash of the overall encoded digits is highly intricate for an attacker.

Now for calculating the hash digest of encoded digits, generated from the partitions of SDFG as shown in Fig. 3.3. The final encoding of the partition P_1 is "nSmnVS", now these alphanumeric characters are provided as input to the SHA-512 hash function which will transform the encoded digits into a hash digest (HD_1) "98adb4b082e02d3d3bb5bd3 ae8048e02378086da72b6dcebf8dc11f35f2b262b71b0f92ca3e40ef462c614 f0b7947cdbbb238bb0fe8de1859db04a4e89d187df" which are represented in the hexadecimal format for convenience.

(3) Forming author's signature:

The overall signature is the concatenation of all the segments (S_i) where each hash digest (HD_i) is truncated into a segment S_i, and the length of the segment is equal to three times of designer's chosen size. Further, the truncated bitstream is represented in the form of triads, where each triad

represents a signature constraint. The final signature embedded in the design is the concatenation of its different segments $(S_1, S_2, \ldots, S_{N-1})$ generated from the encodings of partitions $(P_1, P_2, \ldots, P_{N-1})$ of SDFG respectively. Hence, the author's signature W is represented as follows:

$$W = \&_{i=1}^{N-1} S_i$$

Where N indicates the total partitions and S_i indicates the signature S_i generated using the encoding of the ith partition of SDFG followed by hashing using SHA-512.

For calculating the segment S_1 from the hash digests HD_1 generated from the encoding of the partition P_1 of SDFG shown in Fig. 3.5. Assuming that the IP designer has truncated the obtained hash-bit stream of HD_1 into 48 bits, therefore there are 16 triads for the segment S_1 (for the sake of brevity). Further, an IP designer can select a signature of varying lengths (scalable) and depending on which no of triads can be increased. For example, in the case of 72 hash-bit stream of HD_1 there will be 24 triads, therefore the more the size of the hash-bit stream, the more the no of triads, which subsequently enables the generation of more security constraints for robust hardware security against IP piracy. Further, this segment S_1 of signature is represented in the form of triads, therefore there will be 16 triads in each signature. The segment S_1 (size =16 triads) from the encoding of the partition P_1 of SDFG (Fig. 3.5) is as follows:
"100-110-001-010-110-110-110-100-101-100-001-000-001-011-100-000".

In this chapter, we discussed the signature generation process of the proposed quadruple-phase watermarking approach as shown in Fig. 3.2. We also demonstrated the signature generation process using an example DSP IP core of FIR digital filter. In the next chapter, we will further discuss the signature embedding process of the proposed watermarking scheme as shown in Fig. 3.2, and also the signature detection mechanism using the proposed approach for authentic IP verification.

Chapter 4

Signature embedding and detection process in the quadruple phase watermarking approach during high-level synthesis

In the previous chapter, we discussed the signature generation process of the proposed quadruple-phase watermarking approach as shown in Fig. 3.2, in this chapter we discuss the signature embedding process and then the signature detection mechanism using the proposed approach for authentic IP detection.

4.1 The proposed signature embedding process

In the signature embedding process each segment of the signature from one partition of the SDFG is embedded into the next partition of the SDFG during the four phases of the HLS process based on the mapping rule provided in Table. 4.1. The signature generated from the proposed signature generation process is in the form of triads. As shown in Table. 4.1, each triad is mapped into a watermarking constraint using a mapping rule, since there are only eight possible combinations of the triads which are “000”, “001”, “010”, “011”, “100”, “101”, “110”, and “111”, we have eight sets of rules and each rule is associated with a single triad. The constraints corresponding with the triad “011” are embedded into the scheduling phase (phase-1) of the HLS process, and the constraints corresponding with the triads “000”, “001”, and “010” are embedded into the register binding phase (phase-2) of the HLS process, the constraints corresponding with the triads “100”, and “101” are embedded into the Functional Unit (FU) binding phase of the HLS process (phase-3), and finally, the remaining constraints corresponding to the triads “110”, and

"111" are embedded into the interconnect binding phase (phase-4) of the HLS process.

From the last chapter, the signature S_1 generated from the partition P_1 of the SDFG of an FIR filter (shown in Fig. 3.2) is: "100-110-001-010-110-110-110-100-101-100-001-000-001-011-100-000". There are a total of 16 triads in the generated signature S_1, where each triad is mapped into a hardware security constraint using the mapping rule shown in Table. 4.1, then embedded into any one of the four phases of the HLS. For example, the first triad "100" of the signature is mapped into a security constraint and embedded into the FU binding phase of HLS, similarly, other triads are mapped into a security constraint, and embedded into any one of the four phases of the HLS, the details of the embedding process of security constraints in different phases of HLS are explained in the subsequent sections.

Table. 4.1 Mapping triads in the signature into the hardware security constraints

Triads	Mapping into hardware security (watermarking hardware security constraints)
"000"	Embed an edge between (even, even) node pair in Coloured Interval Graph (CIG).
"001"	Embed an edge between (odd, odd) node pair in CIG.
"010"	Embed an edge between (odd, prime) node pair in CIG.
"011"	Move an operation of non-critical path with highest mobility into immediate next control step (C).
"100"	Bind vendor-1 to even opn and vendor-2 to odd opn.
"101"	Bind vendor-1 to odd opn and vendor-2 to even opn.
"110"	Assign odd register to the 'right' input of FU and even register to the 'left' input of FU.
"111"	Assign odd register to the 'left' input of FU and even register to the 'right' input of FU.

4.2 Embedding constraints in the scheduling phase (phase-1)

During this phase, the constraints corresponding to the triad "011" are embedded into the scheduling phase of the HLS process. To implant the constraints (triad "011"), rescheduling of operations is performed during the scheduling phase. The rescheduling of operations during the scheduling phase is performed as follows, the operation with the highest mobility of a non-critical path is moved into the next immediate control step of the SDFG. While rescheduling, the operations with the highest operation number are scheduled first, and the operations with the lowest operation number are scheduled last, the constraints are applied in the order of decreasing order of operation numbers.

Below, we describe how the signature S_1 generated from the partition P_1 of SDFG from the previous chapter is embedded into the scheduling phase of HLS in the partition P_2. The no of triads of type "011" in the signature S_1 (generated in Chapter 3) is only one, so there is only one security constraint that will get embedded into the partition P_2 of the SDFG shown in Fig. 3.3, in the scheduling phase of the HLS. The partitioning P_2 post embedding of the security constraints generated from the partition P_1 is shown in Fig. 4.1.

There are three operations which are in the non-critical path each having a mobility of one control step in the partition P_2 of the SDFG, since each operation has the same mobility we choose the operation with the highest operation number i.e, opn #13, as evident from Fig. 4.1, the opn #13 is moved from the control step #4 to control step #5. The post-embedding of security constraints generated from the partition P_1 into the partition P_2 is shown in Fig. 4.1.

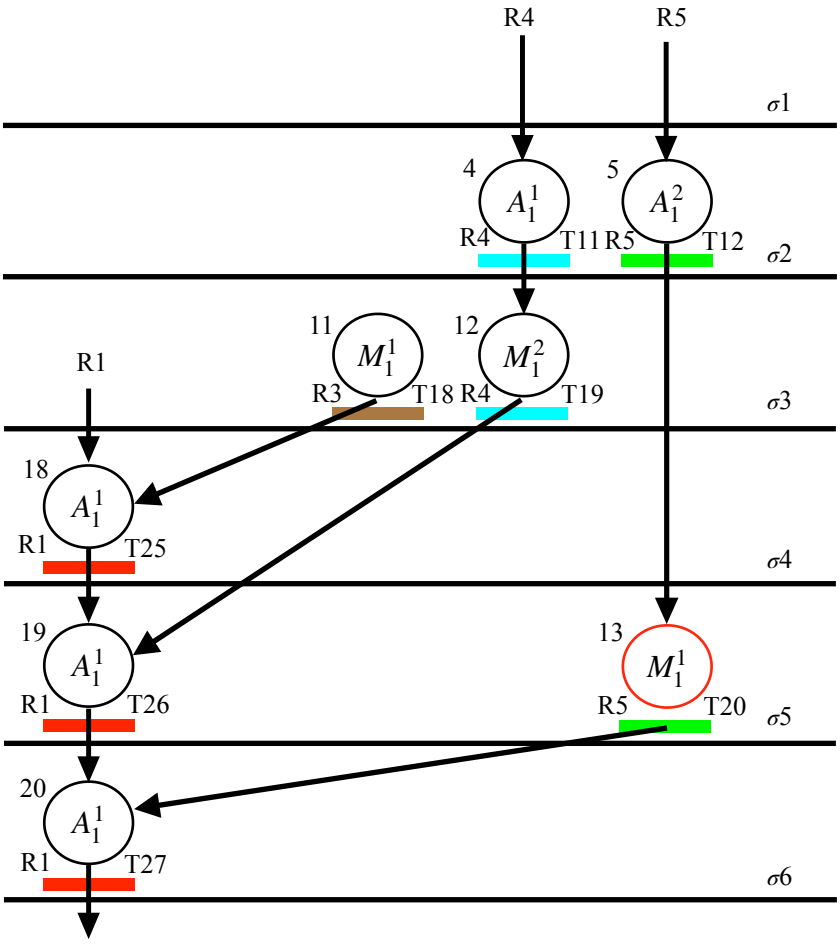


Fig. 4.1 Post-embedding scheduling constraints in the partition P_2 of the SDFG (shown in Fig. 3.3)

✦ **Embedding constraints in the register binding phase (phase-2)**

Post embedding constraints in the scheduling phase, the constraints corresponding to the triads “000”, “001”, and “010” are embedded into the register binding phase of the HLS using a coloured interval graph (CIG) framework. To achieve this, a CIG is created for the respected partition of the SDFG in which the scheduling constraints are embedded. A CIG [30] is a graphical representation of how the storage variables (Ti) are bound to the registers (Ri) in the design. The CIG consists of nodes and edges that indicate the lifetime of the storage variables and where they overlap in the design. The security constraints for the triads “000”, “001”, and “010” are represented by the constraint edges based on the mapping rule shown in

Table. 4.1. By embedding these constraint edges into the CIG, local alterations are made to the register binding of the storage variables, as a result, the storage variables are bound to registers based on the imposed constraints.

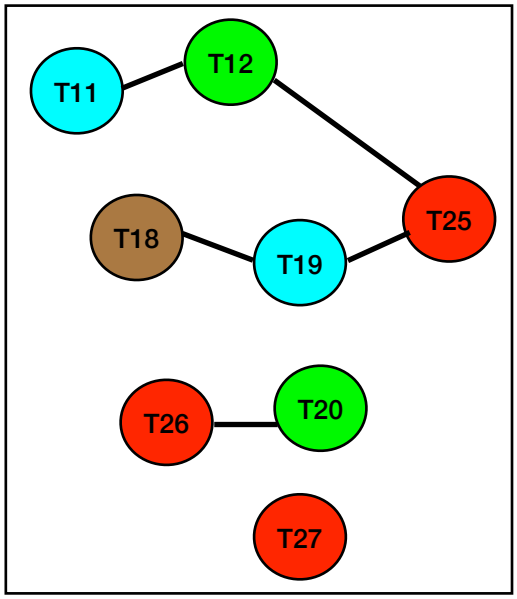


Fig. 4.2 CIG of partition P_2 of SDFG post-embedding scheduling constraints

As shown in Fig. 4.2, a CIG is created from the SDFG of partition P_2. From the signature S_1, the number of triads of type “000” is two, so there are two security constraints associated with this triad, the mapping rule associated with this triad is to embed an edge between (even, even) node pairs. The number of triads of type “001” is three, therefore there are two security constraints generated from this type of triad from the signature S_1. As shown in the mapping rule table, the mapping rule associated with this triad is to embed an edge between (odd, odd) node pairs in the CIG. Similarly, the number of triads of type “010” is one, and one security constraint is generated from this type of triad from the signature S_1. The mapping rule associated with this triad is to embed an edge between (odd, prime) node pairs in the CIG. These constraint edges are inserted one by one into the CIG. It is important to recognise that when two nodes have the same colour, they cannot be connected by an edge.

This is because two storage variables cannot occupy the same register simultaneously. Among all the constraint edges generated from the triads using the mapping rule table the two possible constraint edges between the (even, even) node pairs are (T12, T26) and (T12, T20). First, we insert an edge between the node pair (T12, T26), both of the nodes are of different colours, so there is no need to alter the colours of the nodes because of no conflict between them. Later, we insert an edge between the node pair (T12, T20), and there is a conflict between the node pairs because of the same colour, so we need to alter the colours of the node. To resolve the conflict, we can change the colour of node T20 from green (R5) to cyan (R6). Next, the three possible constraint edges between the (odd, odd) node pairs are (T11, T27), (T11, T25) and (T19, T27). All of the node pairs are in a different colour, therefore there is no conflict after inserting an edge between them, so there is no need to alter the colours of the nodes. Finally, one of the possible constraint edges between the node pairs is (T11, T19).

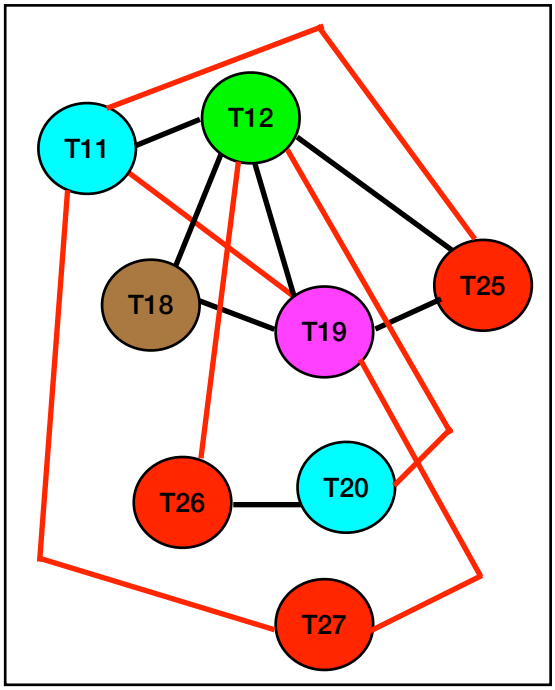


Fig. 4.3 CIG of partition P_2 of SDFG post-embedding register binding constraints
Note: Red-coloured edges denote constraint edges

There is a conflict while inserting an edge between the node pair (T11, T19), both the colours of the nodes are of the same colour, so to resolve this conflict we need to alter the colours of any one of the nodes. For example, the colour of node T11 is changed from cyan (R4) to magenta (R6) in the CIG post embedding the register binding constraints as shown in Fig. 4.3.

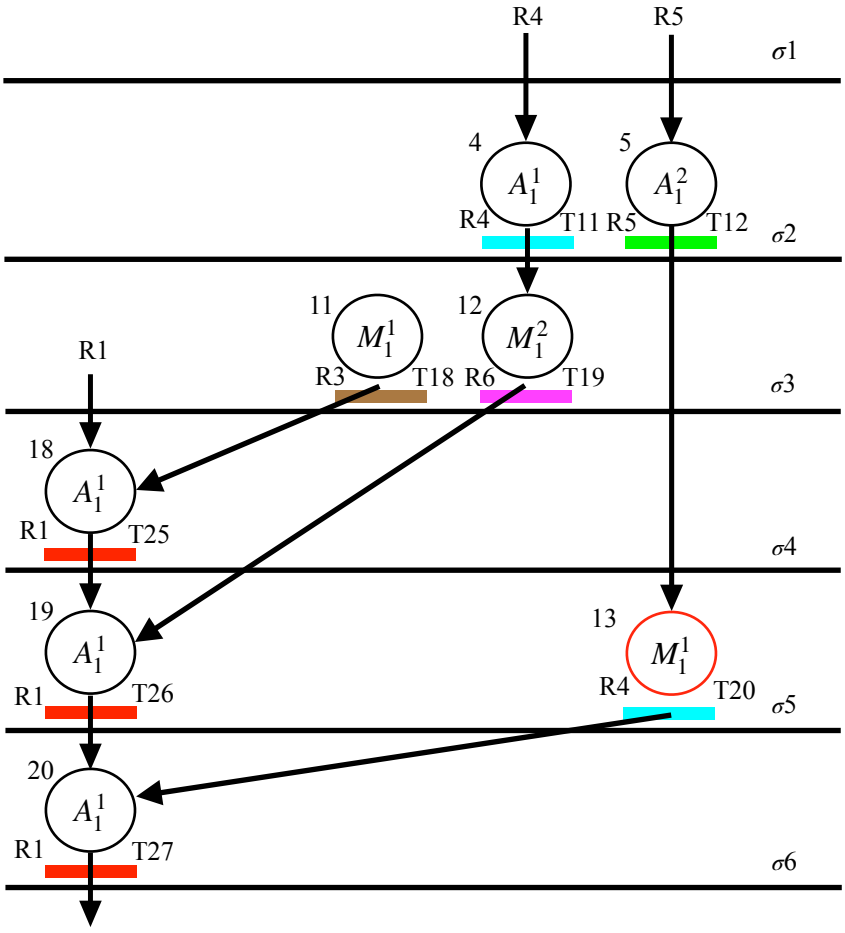


Fig. 4.4 Post embedding register binding constraints in partition P_2

✦ **Embedding constraints in the FU binding phase (phase-3)**

Next, the security constraints linked to the triads “100” and “101” are embedded into the design in the functional unit binding phase of the HLS process by associating an operation with the specific vendor’s functional unit (FU) determined from the mapping rule as shown in Table.

4.1. A constraint is embedded by binding an opn to the corresponding FU unit of a specific vendor. In the signature S_1 generated from the partition P_1 of SDFG (as shown in Fig 3.3), there are four security constraints which are of triad type "100" and one security constraint of triad type "101". From the mapping rule from Table 4.1, security constraints of type "100" are embedded into the functional unit (FU) of the HLS process by binding vendor-1 to even operation and vendor-2 to odd operation. Similarly, the security constraints of type "101" are embedded into the functional unit (FU) of the HLS process by binding vendor-1 to odd operation and vendor-2 to even operation. Based on the FU binding constraints from the signature S_1, the opns are assigned to the respective FU of a specific vendor number as shown in Fig. 4.6 (highlighted using the red colour of FU).

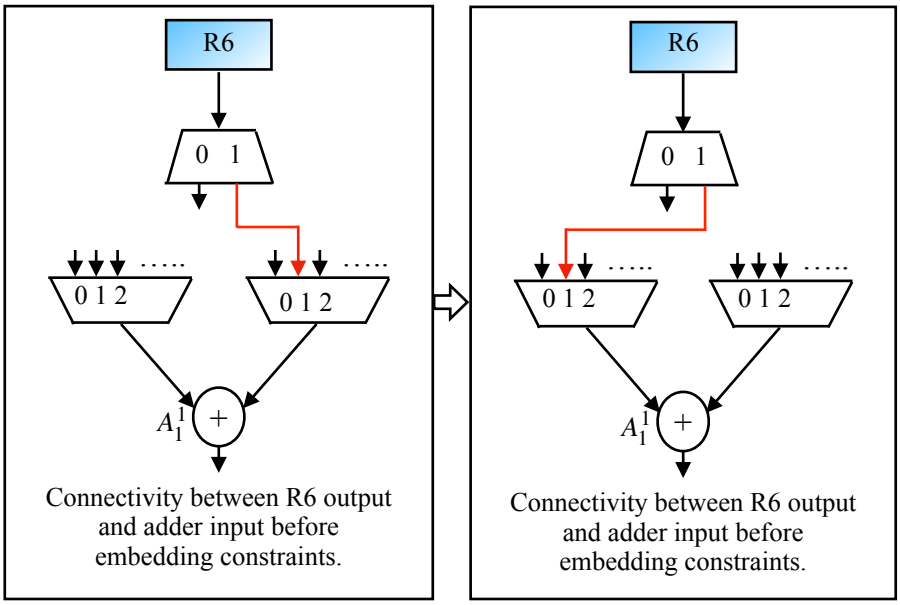


Fig. 4.5 Embedding of constraints in interconnect binding phase, on RTL

✦ **Embedding constraints in the interconnect binding phase (phase-4)**

The security constraints corresponding to the triads "110″ and "111″ are embedded in interconnect binding phase of the HLS design process. To embed a constraint, a specific register with even or odd parity is assigned

to either the left or right input of the functional unit based on the mapping rule shown in the Table., the output registers are then chosen in increasing order of their associated operation number to embed the constraints one by one. An example of how this affects the RTL circuit is depicted in Fig. 4.5, where the interconnect binding constraint for the triad "101" resulted in register R6 (with even parity) being assigned to the left input of the adder unit after the constraints were embedded.

Demonstration of embedding signature S_2 into partition P_3:

Using the proposed encoding tree (shown in Fig. 3.4), the SDFG of the partition P_2 (shown in Fig. 4.6) is encoded into the alphanumeric characters "VmmSnLnL". Then, these alphanumeric characters are first

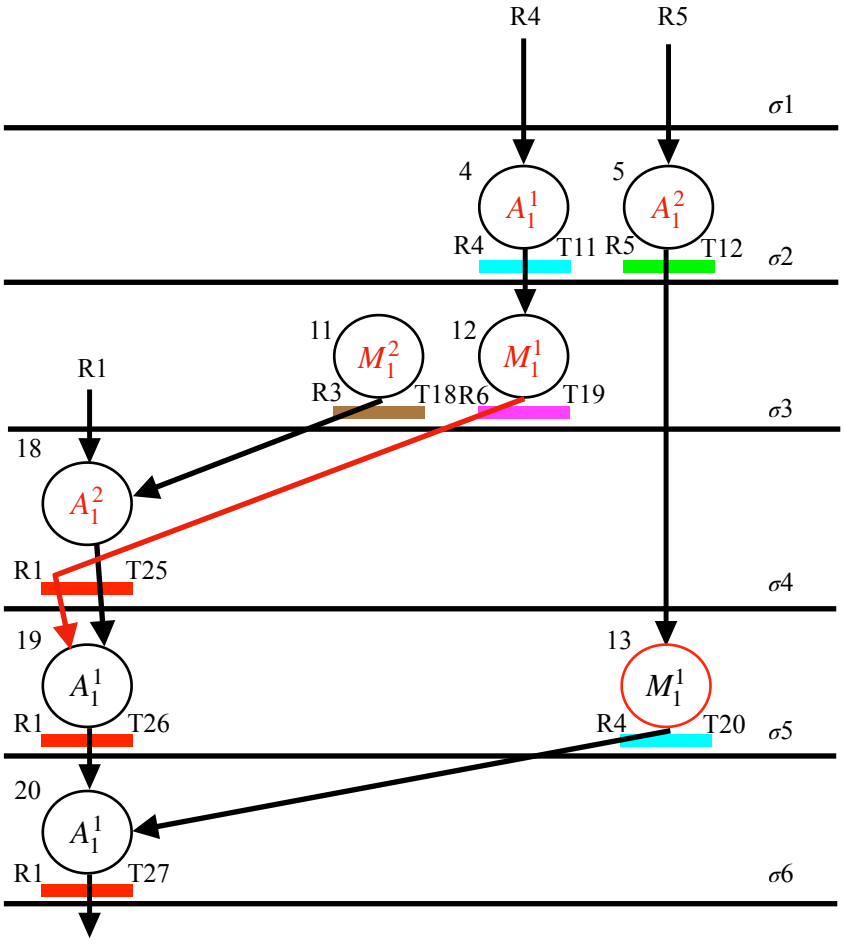


Fig. 4.6 SDFG of partition P_2 post-embedding signature S_1 generated from partition P_1.

transformed into a 512-bit hash digest. Further, the obtained hash bit stream is truncated to 33 (= 11 * 3) bits based on the designer-chosen size 11 of the signature S_2. The signature S_2 (size = 11 triads) is "100-101-010-001-001-000-110-110-011-010-010". Once the signature S_2 is generated from the partition P_2 (shown in Fig. 4.6), the triads of the signature are mapped into the hardware security constraints using the mapping table (shown in Table. 4.1). These hardware security constraints are embedded during the four phases of the HLS process. The details of the constraints to be embedded are shown in Table. 4.2. From Table 4.2, the security constraints corresponding to the triad "011" are embedded during the scheduling phase of the HLS process, by shifting operation #16 from control step #5 to #6 in the SDFG of the partition P_3 (shown in Fig. 4.8).

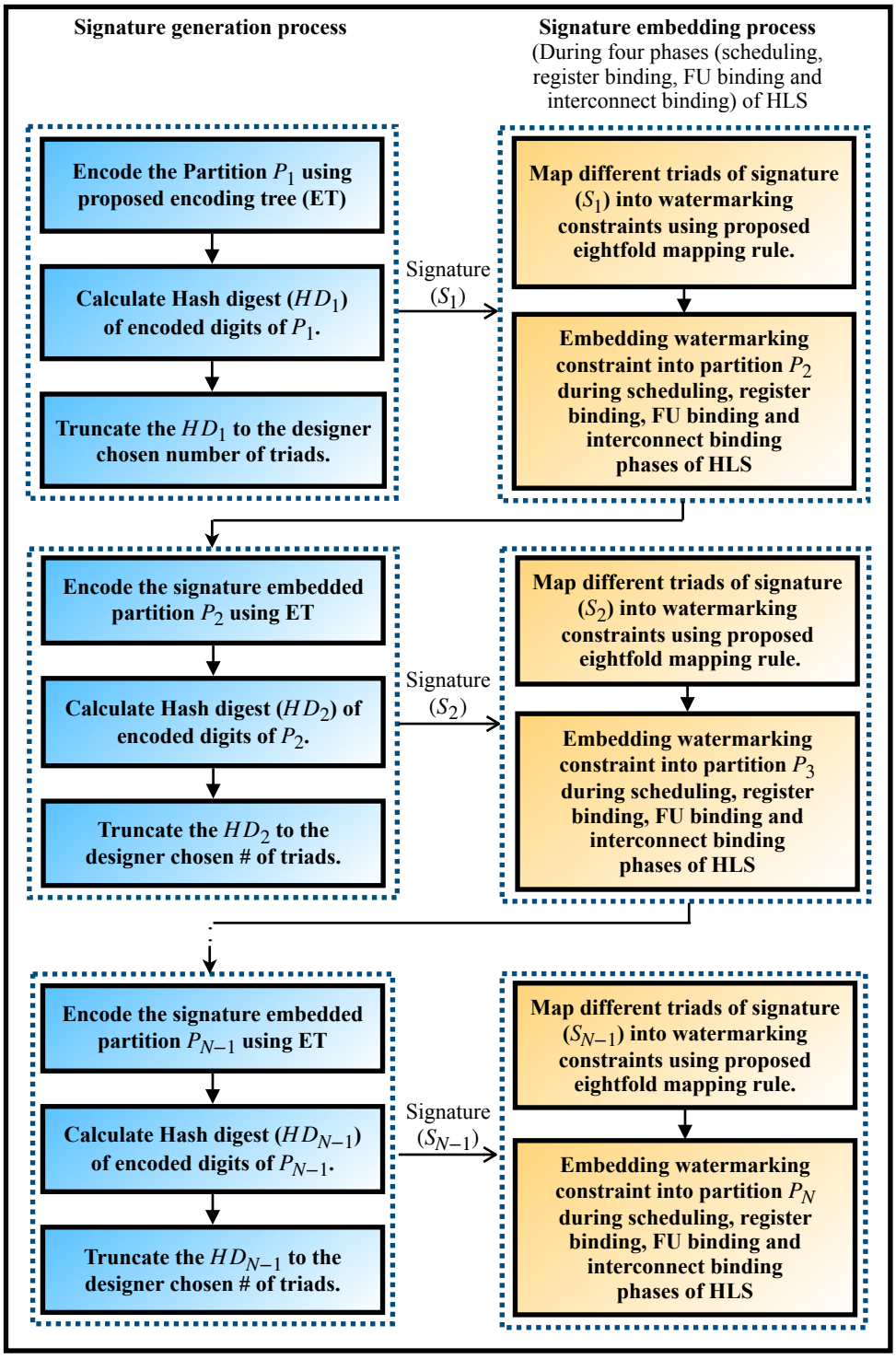
Post embedding security constraints in the scheduling phase, now the security constraints corresponding to the triads "000", "001" and "010" are embedded during the register binding phase. Initially, a CIG is created from the partition P_3, then based on the security constraints of the register binding phase listed in Table 4.2, constraint edges are inserted one by one into the CIG. No two nodes (storage variables) with an edge connecting them can have the same colour (register) in a CIG because two storage variables cannot share the same register, so after implanting the edges derived from the security constraints of the register binding phase, alteration of node colours (registers) takes place if at all required to resolve the conflict raised between any two nodes.

The signature constraints represented by the triads "100" and "101" (as listed in Table 4.2) are embedded into the partition P_3 during the FU binding phase. Based on the FU binding constraints, the operations are assigned to the respective FU of a specific vendor number, which is highlighted using the red colour (shown in Fig. 4.8). The signature constraints represented by triads "110" and "111" (listed in Table. 4.2) are

embedded into partition P_3 during the interconnect binding phase. Based on the interconnect binding constraints, the registers are assigned to specific inputs of FUs, which are highlighted using red arrows in Fig. 4.8. Thus, all the signature constraints generated from the partition P_2 are successfully embedded into the partition P_3 of SDFG (shown in Fig. 4.8) during the four phases of the HLS process.

```
Begin
    Mapping triads in the signature into security constraints;
    if triads in ( “011” ) do
        # Embedding constraints in the scheduling phase;
        The opn with highness mobility is scheduled in the immediate
        next control step;
    endif
    if triads in  ( “000”, “001”, “010” ) do
        # Embedding constraints in the register binding phase;
        A coloured-interval graph (CIG) is created from the scheduling
        graph modified post embedding scheduling constraints;
        if triad == “000” do
            Embed an edge between (even, even) node pair in CIG;
        endif
        if triad == “001” do
            Embed an edge between (odd, odd) node pair in CIG;
        endif
        if triad == “010” do
            Embed an edge between (odd, prime) node pair in CIG;
        endif
    endif
    if triads in ( “100”, “101” ) do
        # Embedding constraints in the FU binding phase;
        if triad == “100” do
            Bind vendor-1 to even opn and vendor-2 to odd opn
        endif
        if triad == “101” do
            Bind vendor-1 to odd opn and vendor-2 to even opn
        endif
    endif
    if trains in ( “110”, “111” ) do
        # Embedding constraints in the interconnect binding phase;
        if triad == “110” do
            Assign odd register to the ‘right’ input of FU and
            even register to the ‘left’ input of FU;
        endif
        if triad == “111” do
            Assign odd register to the ‘left’ input of FU and
            even register to the ‘right’ input of FU;
        endif
    endif
End
```

(a)



(b)

Fig. 4.7 (a) Pseudo code of the embedding process, (b) Signature generation and embedding flow of proposed quadruple phase watermarking approach.

Further, the algorithmic representation of the embedding process is depicted in Fig. 4.7(a). The signature generation and embedding process of

the proposed quadruple-phase watermarking approach is demonstrated in Fig. 4.7(b).

Table. 4.2 Watermarking constraints for embedding in the partition P_3.

Embedding phase	Triads in signature	Corresponding watermarking (security) constraints based on mapping rules.
Scheduling	011	Shift opn16 from C5 to C6
Register binding	000	Edge between node pair (T14, T22) in CIG
	001	Edge between node pair (T15, T23) in CIG
	001	Edge between node pair (T15, T29) in CIG
	010	Edge between node pair (T13, T29) in CIG
	010	Edge between node pair (T23, T13) in CIG
	010	Edge between node pair (T21, T13) in CIG
FU binding	100	Bind operation 6 to the FU of vendor 1
	101	Bind operation 7 to the FU of vendor 1
Interconnect binding	110	R6 (storage variable T21) to right input of FU
	110	R7 (storage variable T22) to left input of FU

Using the first partition, the signature S_1 is generated and then it is embedded into the next partition P_2 during the four phases of the HLS process, and again it is used to generate the signature S_2. This generated signature S_2 is embedded into the partition P_3. This process continues (as shown in Fig. 4.7) up to N-1 partitions and finally, the signature S_{N-1} is embedded into the last partition P_N (N is the number of partitions). The final DFG of the FIR core (shown in Fig. 3.3) after embedding the security constraints using the proposed quadruple-phase watermarking approach is shown in Fig. 4.9.

4.3 Signature detection in the proposed watermarking approach

The designer's signature must be identified in the design to detect and prevent IP piracy and false claims of IP ownership. There are two

scenarios based on the threat model: (i) ensuring that only genuine IPs are integrated into systems and (ii) preventing IP misuse and fraudulent claims

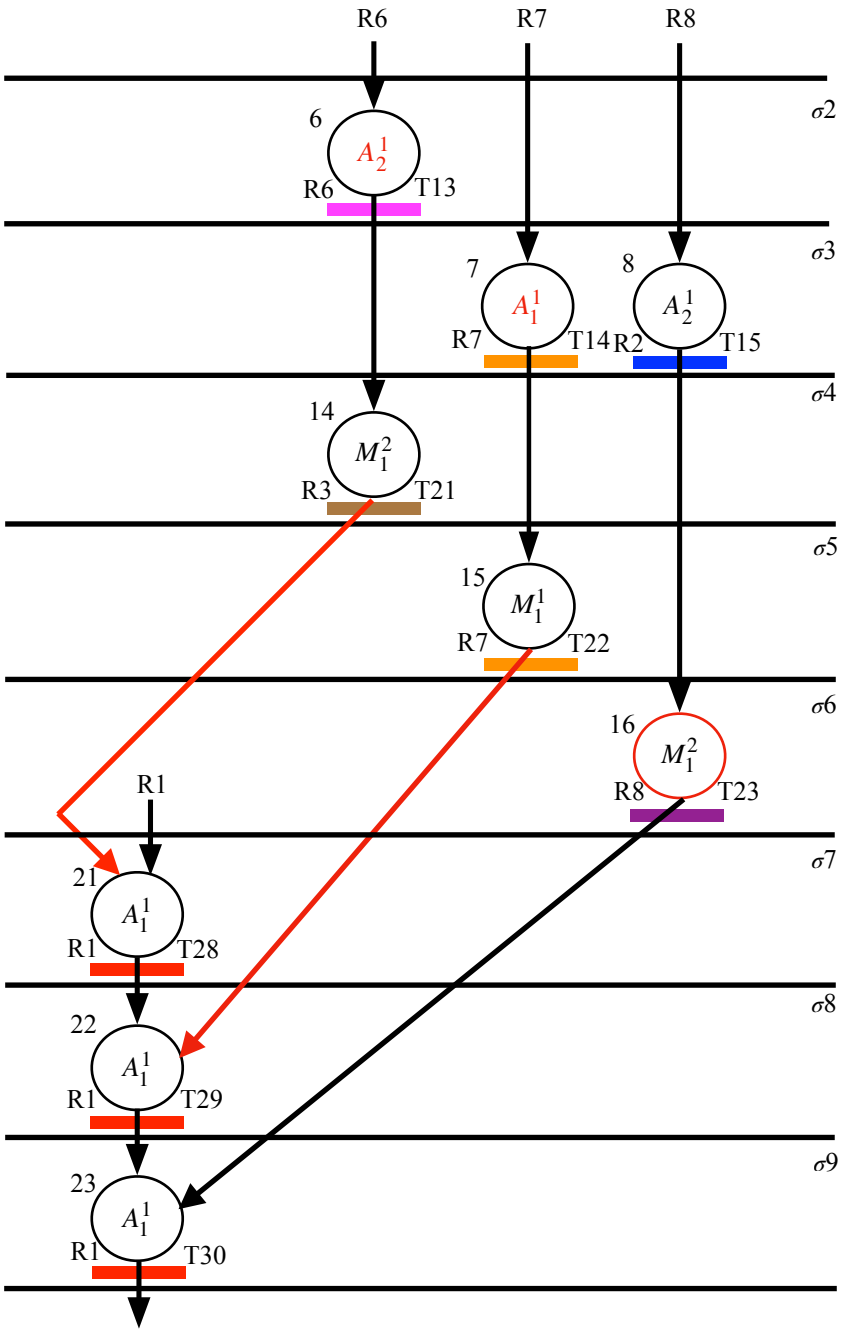


Fig. 4.8 SDFG of partition P_3 post-embedding signature S_2 generated from partition P_2.

of IP ownership. In the first scenario, the signature is detected in the SoC design stage to prevent IP counterfeiting. In the second scenario, the

author's signature is detected in the hardware IP core under test in specialised IP courts to resolve ownership conflicts. The signature

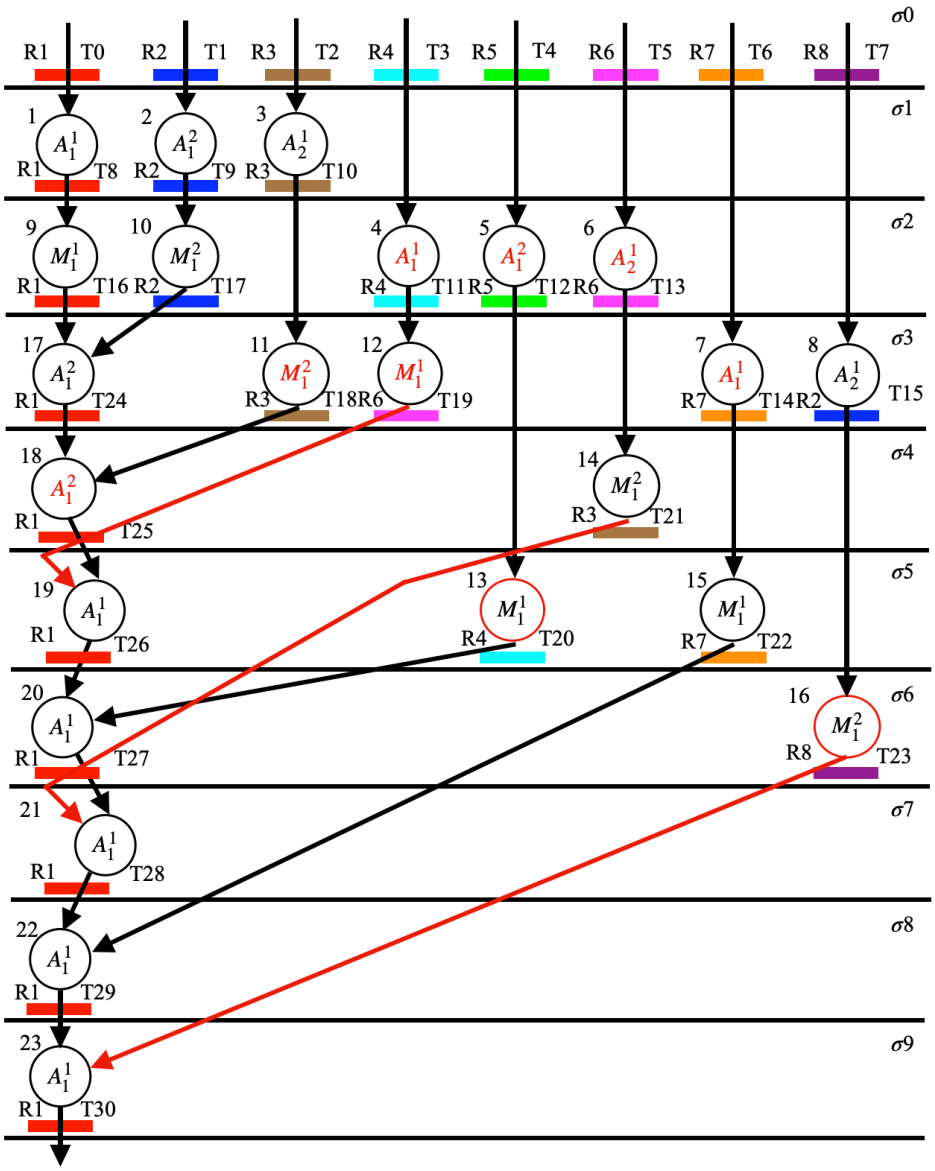


Fig. 4.9 SDFG of FIR core after embedding watermark.

detection process involves converting the signature triads into security constraints using mapping rules and embedding them into various design phases, including scheduling, register binding, FU binding, and interconnect binding. Inspection of these constraints in the controller HDL file and datapath HDL file of the design can determine if the true vendor's signature is present in the design. If the signature is detected, the IP design is considered authentic, and if not, it is likely counterfeit. Using this

approach to detect the proposed robust watermark in designs can ensure the use of secure and reliable hardware in computing systems. Fig. 4.10 illustrates this process.

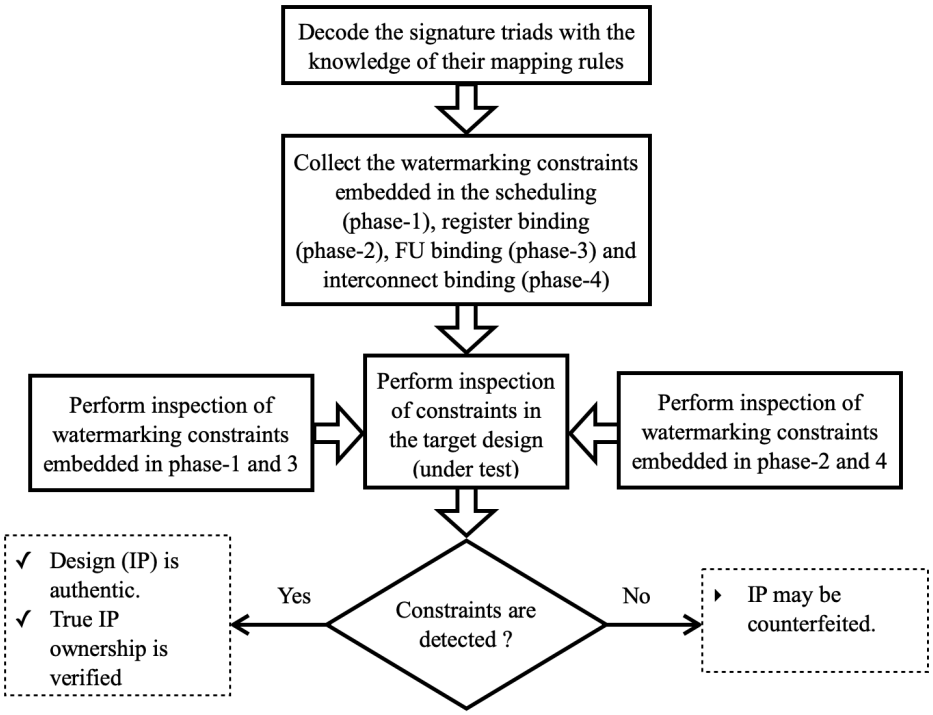


Fig. 4.10 Signature detection using the proposed approach for authentic IP verification

Chapter 5

Exploring Unified Biometrics with Encoded Dictionary for Hardware Security of Fault-Secured IP Core Designs

Digital signal processing (DSP) intellectual property (IP) cores are an integral part of many consumer electronic (CE) devices, including smartphones, cameras, and IoT-enabled devices. These hardware IP cores perform critical tasks such as audio processing/filtering and image/video processing etc. Therefore, in such critical situations, ensuring the proper operation/functionality of DSP hardware IP cores against the occurrence of faults is very important. However, these DSP hardware IP cores are vulnerable to faults caused by single-event upsets (SEU). These faults can be triggered by alpha particles (due to the uranium and thorium impurities in the system-on-chip while packaging [31]), electromagnetic interference, or noise. With transistors' increasing complexity and speed, multi-cycle transient faults manifested from SEU have become a major concern for DSP hardware IP cores. To mitigate the risks associated with SEU faults, fault-secured DSP hardware IP cores are used in many data-intensive applications [32, 33, 34, 35]. However, the integration of third-party IP vendors in the modern CE system design process makes the fault-secured DSP hardware IP cores vulnerable to IP piracy threats [36], [37]. An adversary present in the third-party design house may attempt to pirate the design illegally without the knowledge of the genuine designer. Pirated fault-secured DSP cores can lead to the loss of confidential information, safety and integrity risks, and other potentially serious consequences [38].

It is crucial to verify the authenticity against piracy of a fault-secured hardware IP core supplied by an untrustworthy third-party vendor before integrating it into a CE system. The reason is, an adversary may

attempt to replicate the embedded secrets mark of an authentic IP core and embed it into fake, unreliable IP cores to evade piracy detection (in case the embedded security mark is vulnerable). Such pirated fault-secured IP cores undergo little to no quality checks and testing, posing a significant risk to end consumers in terms of safety hazards. To combat this issue, our proposed unified biometric-driven hardware security methodology offers a robust detective control mechanism that provides digital evidence for the authentication of genuine IPs. This methodology comprises multiple security parameters that actively enable robust unified biometrics signature generation, making it impossible for an adversary to relocate and reproduce the secret signature. By safeguarding end consumers from unreliable CE systems with pirated IP cores, our proposed work provides assured detection and isolation of pirated fault-secured IP DSP designs from the design chain, ensuring the safety of CE systems through proactive validation techniques. Furthermore, this approach alleviates any concerns for end consumers using fake unreliable CE systems, as the trustworthiness of the hardware IP cores has been ensured at the system integration level of the design cycle. Mass production of authentic CE systems using our proposed methodology will also lead the sustained goodwill and reputation for the product and manufacturer in the market.

Threat Model:

The focus of the proposed approach is to safeguard fault-secured DSP IP designs from potential hardware threats such as 'IP piracy' and 'Evading pirated IP detection processes'.

1. IP piracy:

One of the major challenges faced by the designers/vendors of DSP IP cores is the threat of IP piracy, which can occur at any stage of the IC design process. An adversary in a third-party design house may illegally pirate the original IP core during the design process, leading to serious implications in terms of hardware security threats. To address this problem,

the proposed methodology provides a seamless detection mechanism for pirated DSP IP cores. This is made possible due to the embedded hardware security constraints in the IP core design based on encoded dictionary-driven unified biometrics signature. With the help of embedded hardware security constraints, the detection of pirated IP cores becomes easier for the original IP vendor (having all the knowledge of security parameters). Therefore, allowing the designers/vendors to take proactive measures to prevent potential IP infringement.

2. Evading pirated IP detection processes

While the detection of pirated IP cores is crucial, the proposed methodology also addresses another important issue - security against evading the detection process. An adversary may attempt to evade the IP piracy detection process by intending to copy the original signature into the fake IP core. This can lead to the creation of a fake IP core that appears to be genuine but is an infringement of intellectual property rights. The presented security methodology thwarts such attempts by making or regenerating the original biometrics signature difficult. This is due to several intricate security features such as the biometric feature generation process, expandable encoded dictionary rules to select hybrid biometrics signature and encoding rules for secret security constraints generation. As a result, an attacker fails to copy and implant the original security mark in the pirated IP core, ensuring the authenticity of the IP and the safety of the end consumer.

5.1 Proposed hardware security methodology for securing fault-secured DSP IP cores

The proposed approach is a hardware security methodology that uses biometric information such as palmprint, facial, and fingerprint data and an encoded dictionary to safeguard the fault-secured DSP IP core designs against IP piracy. This approach can also be applied to regular DSP

IP core designs. The proposed technique takes into account several inputs, including the data flow graph (DFG) of the DSP application, resource constraints, the module library, and biometric information (palmprint, facial, fingerprint information) of the original IP designer. The outcome of this methodology is a protected and fault-secured DSP IP core, which utilises multimodal biometrics and an encoded dictionary. The process of generating a protected fault-secured DSP IP core design involves four main processing blocks: 1) Fault-secured DSP design block, 2) Multimodal biometrics signature block, 3) Encoded dictionary block and 4) Security constraints embedding block.

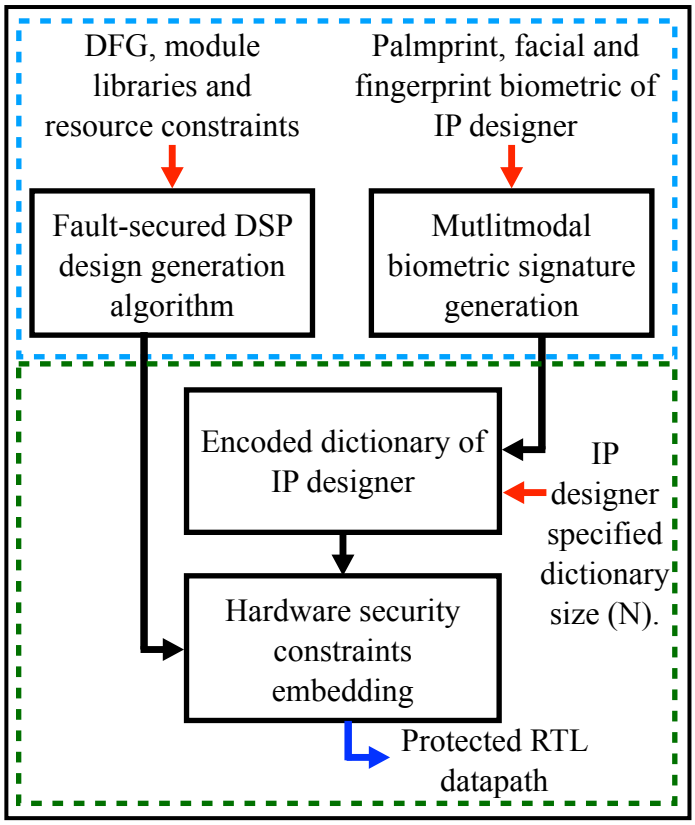


Fig 5.1 Overview of the proposed methodology

As shown in Fig. 5.1, The first processing block, the fault-secured DSP design block, is responsible for generating a fault-secured schedule

using a scheduling algorithm and also allocates the registers to the design using a register allocation algorithm by taking inputs such as the data flow graph (DFG), module library, and resource constraints. This processing block ensures that the design is free from faults or vulnerabilities. The second processing block, the multimodal biometrics signature generation block, generates a unified biometric binary template as digital evidence for the IP designer. This block accepts captured images of the IP designer's palmprint, facial, and fingerprint biometrics and produces a binary template for subsequent processing blocks. The generated binary template is unique to the designer and serves as a form of digital identification.

The third processing block, the encoded dictionary block, produces secret security constraints using the unified biometric binary template. The security constraints are generated based on the selected strength and combination using the designer-created encoded dictionary for embedding. The number of security constraints embedded into the design can be increased by choosing more biometric features, followed by their respective encoding. The IP vendor can vary the strength of embedded security information by varying the number of features of their multimodal biometrics. The size of the encoded dictionary determines the exact set of security constraints and the encoded bits selected by the IP vendor.

The fourth processing block, the security constraints embedding block, generates a secured register transfer level (RTL) datapath using behavioural synthesis. This block accepts the generated fault-secured scheduled design and encoded dictionary-based unified biometric signature of the IP designer as input. It embeds the security constraints into the design and creates a secured RTL datapath that protects the design against IP piracy. Thus, the proposed methodology offers a comprehensive and effective approach to protecting fault-secured DSP IP core designs against hardware IP piracy. The flow of the proposed methodology in terms of the

four processing blocks is shown in Fig. 5.1. In this chapter, we are going to discuss the processing blocks which are highlighted in the blue colour box as shown in Fig. 5.1, and in the next chapter, we are going to discuss the remaining two processing blocks which are highlighted in the red colour box as shown in Fig. 5.1.

In this proposed approach, the process of IP piracy detection is carried out by comparing the extracted security constraints from the DSP RTL design being tested with the original pre-stored biometric image-driven unified digital template-based secret security constraints. A successful 100% match between the two results in the design being deemed genuine, while a mismatch indicates that the design is likely pirated. This approach enables the detection of fake/pirated DSP IPs in the design chain, thereby achieving detective control. Furthermore, the matching process does not require the true IP vendor to recapture their multimodal biometric information. Instead, the original pre-stored biometric image-driven unified digital template of the true IP vendor is used for matching during the detection process. The biometric feature dimensions, its respective digital template, and associated security constraints can be accurately recomputed from the pre-stored palmprint, facial, and fingerprint images for successful IP piracy detection. As a result, factors such as injury marks, grease on the finger and palm, camera variation in resolution, and differences in cropping size have no impact on the proposed IP piracy detection process.

For demonstrating the proposed unified biometrics with the encoded dictionary for hardware security of fault-secured IP core designs, we are going to use the DSP application inverse discrete cosine transform (IDCT) 8-point core as an example DSP IP core application and also a hardware security tool is developed based on the proposed hardware security approach. The hardware security tool has three panels, an input

panel, a status bar and an output panel. The input panel is used the provide the inputs to the tool such as resource constraints, module libraries, biometric information, T_c value, and the encoded dictionary code. The status bar is used to highlight the current status of the proposed approach, for example, if the user provided all the inputs required to generate the scheduled DMR design the status bar associated with the DMR design gets highlighted in orange colour (shown in Fig. 5.2). The output panel consists of buttons which are used to display the intermediate and final results of the proposed approach. In the subsequent sections using the hardware security tool, we are going to discuss the first two processing blocks with detailed insights into the techniques used and their roles in achieving the overall goal of securing the fault-secured DSP IP core designs against IP piracy.

5.2 Generating transient fault-secured DSP designs

Generating a fault-secured design for a DSP application involves taking the application's data flow graph (DFG), a module library containing details of the available hardware units, and the transient fault strength (Tc) as inputs. A dual modular redundant (DMR) design is first constructed based on the DFG of the DSP application. This involves duplicating the operations of the original unit to create a sister unit, which is then designated as the DMR design of the DSP application. The generated DMR design is then scheduled using input resource constraints, represented as $Rc = \{XR_1, XR_2 ..... XR_a\}$, where '$X$' represents the number of hardware units and 'a' represents the type of hardware resource. The LIST scheduling algorithm is employed to schedule the DMR design. After obtaining the scheduled DMR design ($SDFG_{DMR}$), the Tc-cycle fault security rules are applied to the design.

The three fault security rules that are applied to the DMR design are as follows:

1. Allocate operations (opn) of the scheduled DMR design to distinct operators based on availability, such that opn (S) ε N_{OG} and opn (S') ε N_{DP}, where N_{OG} and N_{DP} represent the original and duplicate units, respectively.
2. If distinct operators are not available, keep the same assignment for S' as S in N_{DP} such that $t(S') - t(S) > Tc$.
3. If condition 2 is not met, push S' (and its successors) ε N_{DP} one control step below, and repeat the process until the condition is satisfied.

If any of the three rules are violated, it can result in transient fault hazards between similar operations assigned to similar hardware units, which can lead to incorrect functionality. To resolve these hazards, the affected operations (and their successors) are pushed to the duplicate unit in later control steps, ensuring that the interval between (S) ε N_{OG} and (S') ε N_{DP} is not less than Tc. Note: The above-listed fault security rules are sufficient to safeguard the design against transient faults emanating from single-event upsets (SEU). This is because the above fault security rules also consider the transient fault strength of varying size (Tc = 1, 2 etc.,) which mitigates the impact of worst-case pulse widths (temporal effect) due to multi-cycle transient fault using Tc = 2.

The details for generating a fault-secured IDCT 8-point DSP IP core using the hardware security tool are as follows; initially, we need to load the DSP application core design (in our case IDCT 8-point DSP core) into the hardware tool along with the module libraries. Later on, designer-specified resources are provided such as resource constraints, and the strength of fault (T_c) to the hardware tool (for the sake of demonstration we are considering resource constraints as 1-adder, 2-multipliers and the strength of fault T_c = 2). Once the inputs are provided to the hardware

security tool their respective buttons are enabled in the input panel of the hardware security tool (shown in Fig. 5.2).

In our case, an 8-point IDCT DMR design is created from the inputs provided to the hardware security tool and then scheduled using the LIST scheduling algorithm, where R1 to R16 are the required registers, V0 to V45 are the storage variables used for storing the intermediate values, C1 to C15 are the control steps required to schedule the DMR design and M1, M2 are the multipliers and A1 is the adder as shown in Fig. 5.3. The status bar (shown in Fig. 5.2) of the hardware security tool shows the status of the DMR design highlighted with orange colour once the DMR design is created.

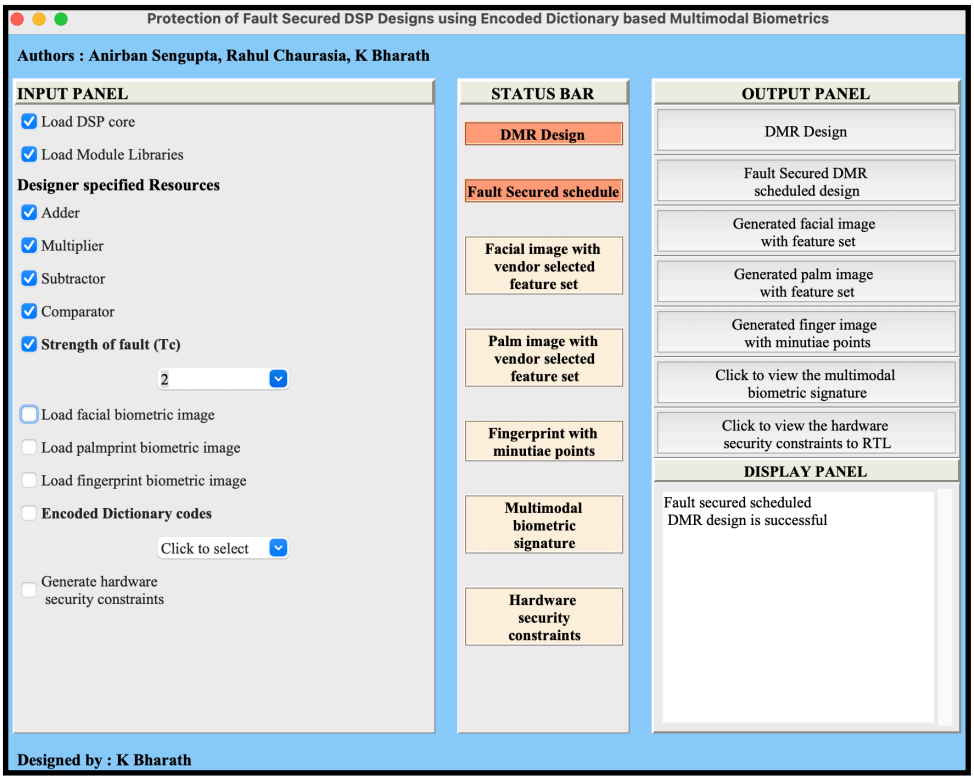


Fig. 5.2 Screenshot of the hardware security tool demonstrating the successful generation of fault-secured DMR design of 8-point IDCT DSP core

Then the design is subjected to the T_c-cycle fault security rules, which ensure that operations are allocated to distinct hardware units based on

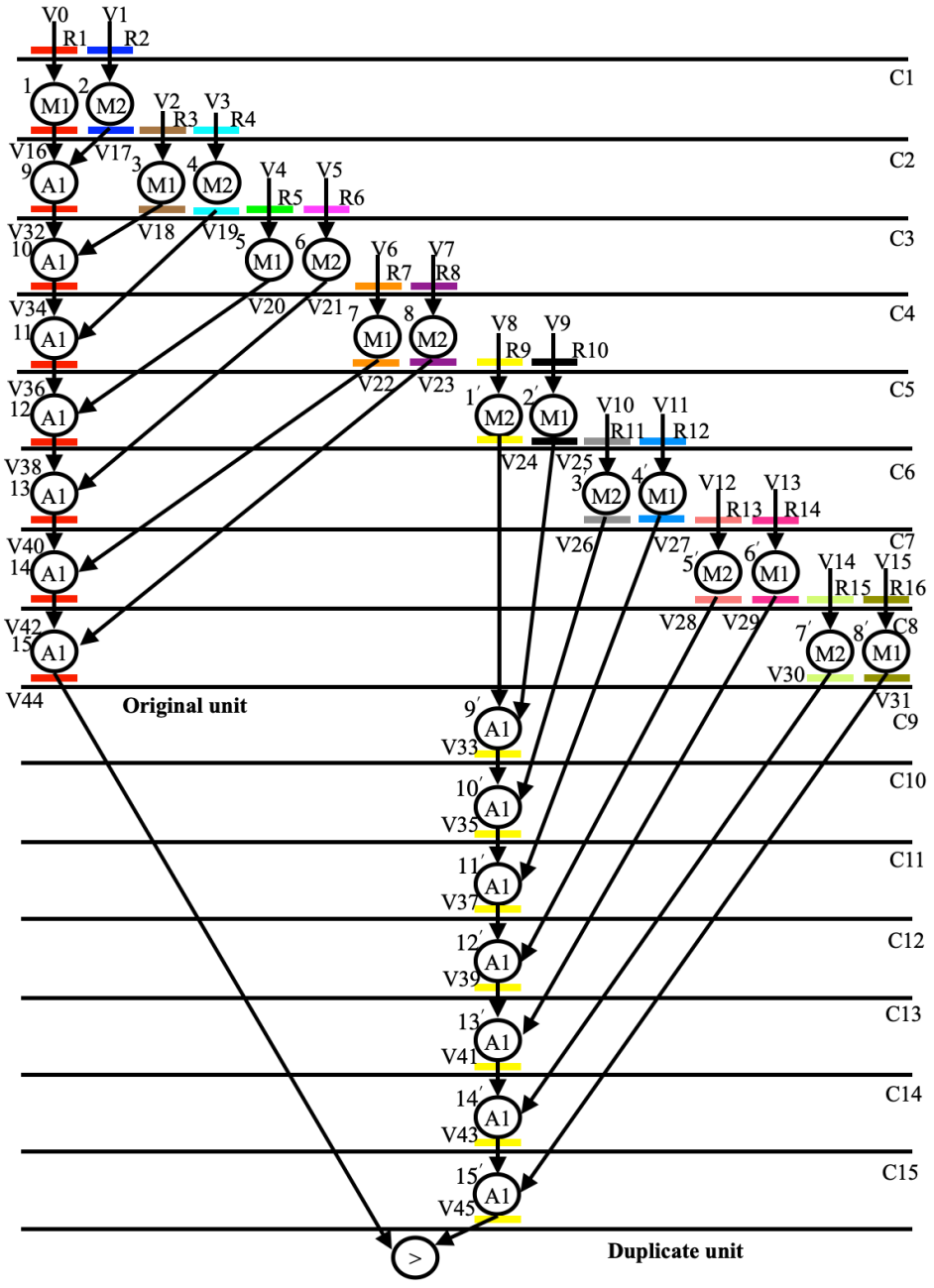


Fig 5.3 Fault-secured scheduled of IDCT filter design (pre-embedding security constraints)

availability and that any violations are resolved by pushing operations in the duplicate unit to later control steps In our example, multiplier operators are allocated distinctively in original and duplicate units of the DMR

design, and since we are restricted to using only one adder (resource constraints provided in the input) in a control step, both the original and duplicate unit in the DMR design of IDCT 8-point DSP core have the same adder operator units Therefore the difference between the control steps of the respective hardware units in the original and duplicate unit in the DMR design should be greater than the strength of the fault (T_c), in our demonstration (shown in Fig. 5.3) the difference between the control steps of the same adder operators in the original and duplicate units is 7 which is greater than the T_c = 2, so there is no need to push the adder operation into the next control step. Following these rules generates the fault-secured scheduled DMR design of the 8-point IDCT DSP IP core (shown in Fig. 5.3). Once the fault-secured 8-point IDCT DSP IP core is generated successfully, the hardware security tool's status bar is highlighted in orange (shown in Fig. 5.2).

5.3 Multimodal biometric signature generation

The proposed unified biometric-driven hardware security methodology involves the integration of three biometric techniques: palmprint biometric [56], facial biometric [57], and fingerprint biometric [58]. Further, for generating the security signature, the biometric information belonging to palmprint, facial and fingerprint can be obtained from the same IP vendor. Further, the biometric information can also be obtained from different IP vendors in case the legal rights of the design belong to more than one IP vendor, for embedding into the design for hardware security. The proposed approach demonstrates the security of fault-secured design against IP piracy using the biometric information from different IP vendors. In the case of palmprint and facial biometrics, nodal point features and in the case of fingerprint minutiae feature points (ridge ending and ridge bifurcation) are exploited for generating multimodal biometric signature. The process of generating the digital signature

corresponding to each biometric using the hardware security tool is discussed in detail below:

Generating facial signature:

The process of generating a facial biometric signature begins by capturing the facial image of the IP designer using a high-resolution imaging device. The captured image is then provided as input to the hardware security tool by enabling the load facial biometric image button in the input panel (shown in Fig. 5.4). The captured facial image is then subjected to a specific grid size and spacing. Once the grid is applied, nodal points are designated on the facial image based on the IP designer's chosen facial feature set. A total of 18 nodal points (P1 to P18), marked in red (as shown in Fig. 5.4 output display panel) are designated to determine the facial features. The coordinate points associated with the 18 points are P1 (240, 120), P2 (240, 250), P3 (170, 280), P4 (310, 280), P5 (130, 285), P6 (205, 285), P7 (275, 285), P8 (345, 285), P9 (105, 325), P10 (375, 325), P11 (240, 360), P12 (195, 375), P13 (220, 375), P14 (265, 375), P15 (290, 375), P16 (185, 440), P17 (305, 440) and P18 (240, 520). After the nodal points are designated, a facial image with all the facial features are generated by the hardware security tool (shown in the output display panel of Fig. 5.4), where each feature is represented as the distance between the

Table. 5.1 Signature generation corresponding to the facial features

Facial features	Naming convention	Feature dimension	Binary representation
HFH	(P1) → (P2)	130	10000010
IPD	(P3) → (P4)	140	10001100
BOB	(P5) →(P8)	215	11010111
IOB	(P6) → (P7)	70	1000110
OB	(P5) → (P6)	75	1001011
WNR	(P2) → (P11)	110	1101110
WF	(P9) → (P10)	270	100001110
HF	(P1) → (P18)	400	110010000
WNB	(P13) → (P14)	45	101101
NB	(P12) → (P15)	95	1011111
OCW	(P16) → (P17)	120	1111000

two nodal points (shown in Table. 5.1). In the facial biometric image, a total of 11 facial features have been marked, as shown in Table. 5.1. The IP designer selected facial feature sets are HFH (Height of forehead), IPD (Inter-pupillary distance), BOB (Bio ocular breadth), IOB (Inter ocular breadth), OB (Ocular breadth), WNR (Width of the nasal ridge), WF (Width of the face), HF (Height of the face), WNB (Width of the nasal bridge), NB (Nasal breadth) and OCW (Oral commissure width). Each of these features is then processed to derive their binarised information. To derive the binarised information, the first step is to determine the feature dimension corresponding to each facial feature using the Manhattan distance. This results in a decimal value corresponding to each feature which represents the magnitude of each feature, it is transformed into its binarised form. The feature dimension and its binary representation of the facial features are shown in Table. 5.1.

Finally, the binarised signature of each facial feature is concatenated to generate the facial biometric signature. The concatenation order can be decided by the IP designer to generate the desired facial biometric signature combination.

Generating palmprint signature:

The first step in generating a palmprint signature in the proposed approach is to capture the palmprint biometric of the IP vendor using a high-quality and high-resolution digital camera. Then the captured image is provided to the hardware security tool as input by enabling the load palmprint biometric image button in the input panel of the hardware security tool (shown in Fig. 5.5). The captured image is then subjected to a

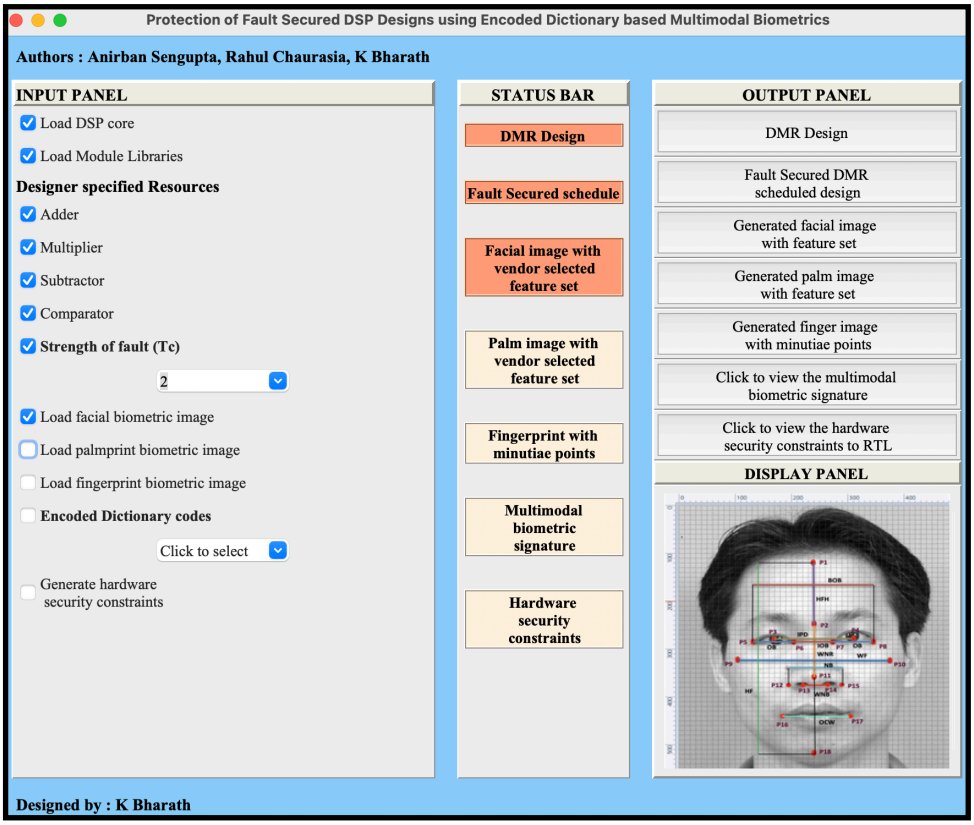


Fig. 5.4 Screenshot of hardware security tool corresponding to the facial image with the vendor-selected feature set on the display panel

specific grid size and spacing to enable the generation of precise nodal points and the coordinates of palmprint features on the palmprint image. Next, nodal points are generated based on the feature set selected by the IP designer. There are a total of 25 nodal points. The coordinate points associated with the nodal points are P1 (350, 5), P2 (300, 30), P3 (415, 50), P4 (350, 110), P5 (285, 130), P6 (415, 160), P7 (495, 170), P8 (350, 220), P9 (285, 230), P10 (415, 245), P11 (495, 265), P12 (285, 320), P13 (350, 325), P14 (495, 335), P15 (415, 355), P16 (230, 390), P17 (495, 405), P18 (70, 470), P19 (180, 480), P20 (495, 490), P21 (120, 495), P22(165, 520), P23 (405, 520), P24 (285, 650) and P25 (350, 650). Each palm feature is a measure of the respective distance between the two nodal points, marked in red (shown in the output display panel of Fig. 5.5). Subsequently, an image of the palm with the IP designer's selected palm feature is generated by the

Table. 5.2 Signature generation corresponding to the palmprint features

Feature name	Naming convention	Feature dimension	Binary representation
DL	(P16) → (P24)	267.75	100001001.11
DHL	(P23) → (P24)	176.91	10110000.111010001111010111
WP	(P16) → (P20)	283.24	100011011.0011110101110000101
LP	(P13) → (P25)	325	101000101
DFF	(P2) → (P5)	101.11	1100101.00011100001010001111
DSF	(P5) → (P9)	100	1100100
DTF	(P9) → (P12)	90	1011010
DFM	(P1) → (P4)	105	1101001
DSM	(P4) → (P8)	110	1101110
DTM	(P8) → (P13)	105	1101001
DFR	(P3) → (P6)	110	1101110
DSR	(P6) → (P10)	85	1010101
DTR	(P10) → (P15)	110	1101110
DFL	(P7) → (P11)	95	1011111
DSL	(P11) → (P14)	70	1000110
DTL	(P14) → (P17)	70	1000110
DFT	(P18) → (P21)	55.90	110111.1110011001100110011
DST	(P21) → (P22)	51.45	110011.0111001100110011
DTT	(P19) → (P22)	42.72	101010.1011100001010001111

hardware security tool as shown in Fig. 5.5. There are a total of 19 palm features (shown in Table. 5.2) selected by the IP designer which are DL (Distance between the start of the life line and end of the life line), DHL (Distance between datum points of head line and life line), WP (Width of palm), LP (Length of palm), DFF (Distance between the first consecutive intersection points of forefinger), DSF (Distance between the second consecutive intersection points of forefinger), DTF (Distance between third consecutive intersection points of forefinger), DFM (Distance between first consecutive intersection points of middle finger), DSM (Distance between second consecutive intersection points of the middle finger), DTM (Distance between third consecutive intersection points of middle finger), DFR (Distance between first consecutive intersection points of ring finger), DSR (Distance between second consecutive intersection points of ring finger), DTR (Distance between third consecutive intersection points of ring finger), DFL (Distance between first consecutive intersection points of the little finger), DSL (Distance between second consecutive intersection

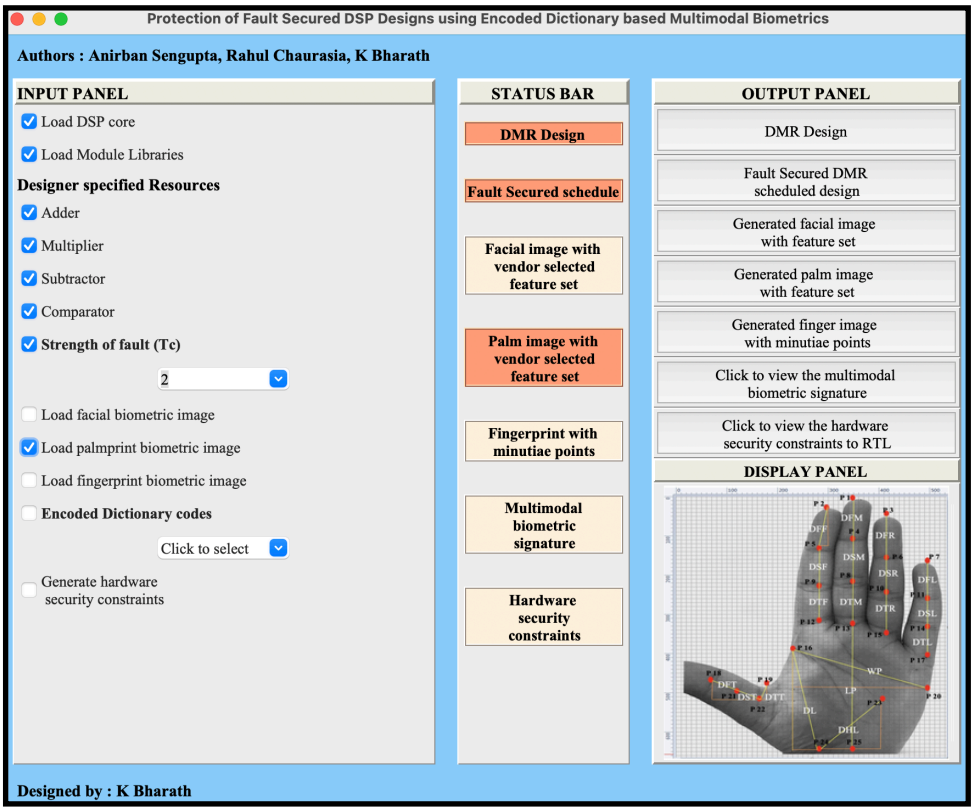


Fig. 5.5 Screenshot of hardware security tool corresponding to the palmprint image with the vendor-selected feature set on the display panel

points of the little finger), DTL (Distance between third consecutive intersection points of the little finger), DFT (Distance between first consecutive intersection points of thumb finger), DST (Distance between second consecutive intersection points of thumb finger) and DTT (Distance between stardust point and the third intersection point of thumb). This image contains all the necessary details to generate the palmprint signature. To do so, we first determine the feature dimensions of all the selected features using Manhattan distance as shown in Table. 5.2. Next, each feature is transformed into its corresponding binarised form (shown in Table. 5.2) and finally, by concatenating the binarised information of each palm feature, the palmprint signature is generated. However, the IP designer can choose from several signature combinations based on different concatenation orders.

Generating fingerprint signature:

To generate a fingerprint signature, the first step is to capture the impression of the fingerprint using an optical scanning device. This fingerprint image is used as input to the security tool by loading it into the hardware security tool by clicking the load fingerprint biometric image button as shown in Fig. 5.6. The captured image then undergoes pre-processing, which involves three sub-processes. The first sub-process is image enhancement using Fast Fourier Transform (FFT) to magnify and reconnect the broken ridges, enhancing image quality.

The second sub-process is binarization, where the image is represented with only two intensity levels (‘0’ for low and ‘255’ for high) by comparing with the threshold intensity of pixels. The third sub-process is thinning, which reduces the thickness of ridge lines to one-pixel width. After pre-processing, the thinned image is used to extract minutiae points, the unique features that define an IP designer’s fingerprint. Minutiae points are the locations where ridge lines end abruptly (termed ridge ending, shown in red in the output display panel image (d) of Fig. 5.6) and where a ridge line bifurcates into branches (termed as ridge bifurcation, shown in blue in the output display panel image (d) of Fig. 5.6). Each minutiae point is then represented in its corresponding binary form as shown in Table. 5.3, which dictates the signature corresponding to each minutiae point. The output of the hardware security tool gives the images of the outputs of each sub-processes (shown in the output display panel of Fig. 5.6) along with the signature for each minutiae point consisting of coordinates (x, y), crossing number (CN) value, minutiae (’n’), and ridge angle in degrees (‘θ’) (shown in Table. 5.3). Finally, a digital template is obtained by concatenating the signatures of each minutiae point. The number of minutiae points and concatenation order can be adjusted by the IP designer to derive a fingerprint signature of the desired strength.

Table. 5.3 Signature generation corresponding to fingerprint minutiae points

CN	x	y	Minutiae type number	Angle in degree	Binary representation
1	190	45	3	34	10111110-101101-11-100010
2	139	46	3	9	10001011-101110-11-1001
3	126	54	1	189	1111110-110110-1-10111101
4	79	64	1	327	1001111-1000000-1-10100 0111
5	181	83	1	38	10110101-1010011-1-100110
6	219	84	3	225	11011011-1010100-11-11100001
7	159	98	1	214	10011111-1100010-1-11010110
8	136	110	1	15	10001000-1101110-1-1111
9	118	115	1	334	1110110-1110011-1-101001110
10	248	130	3	50	11111000-10000010-11-110010
11	192	134	1	46	11000000-10000110-1-10 1110
12	117	137	1	135	1110101-10001001-1-10000111
13	132	150	3	138	10000100-10010110-11-10001010
14	111	164	1	267	1101111-1010 0100-1-100001011
15	149	169	1	239	10010101-10101001-1-11101111

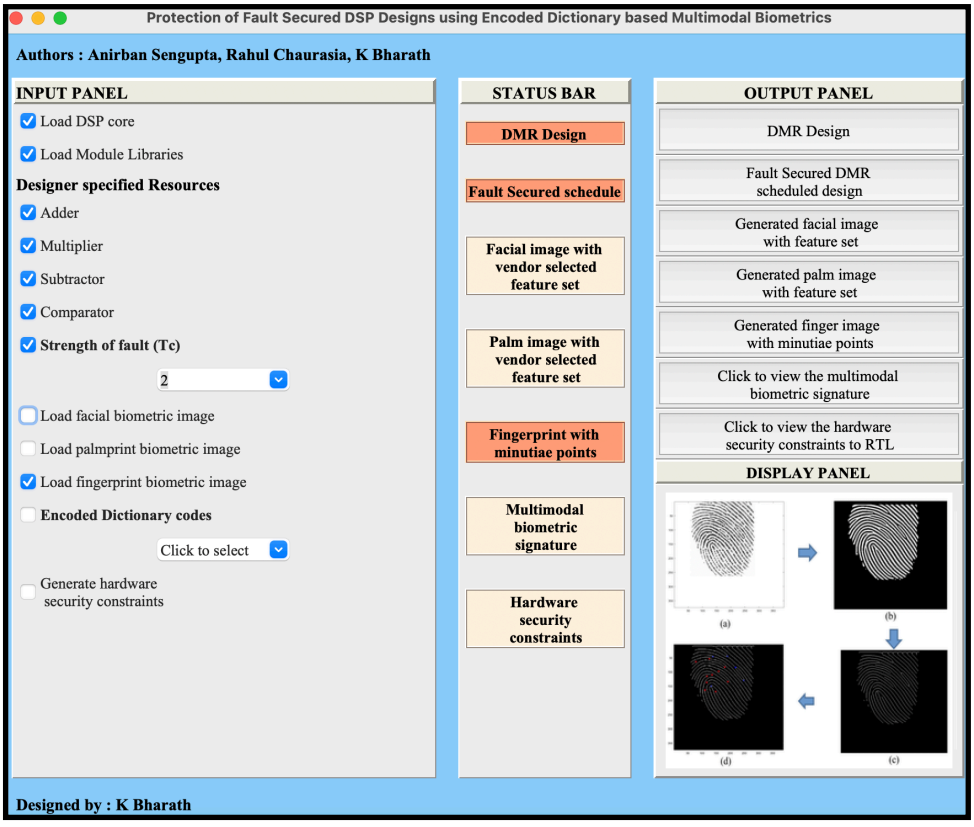


Fig. 5.6 Screenshot of hardware security tool corresponding to the fingerprint image with minutiae points on the display panel

The multimodal biometric signature is generated by concatenating the individual signatures of each biometric using the encoded dictionary. The following are the individual biometric facial, palmprint and fingerprint signatures generated using the hardware security tool. The facial biometric signature is “10000010100011001101011110001101001011110111010000 11101100100001011011011111111000” (83bits), the palmprint signature is “10000100111101100001110100011111010111100011011001 1101011100001011010001.....10101011100001010001 111” (253 bits) and the fingerprint biometric signature is “10111110101101111000101000101110111011100111111110 110110110111101.....10010101101010011111011 1” (350 bits). In the next chapter, we are going to discuss the encoded dictionary block and security constraints embedding block (shown in Fig. 5.1). A unified biometric signature is generated from the encoded dictionary block using proposed encoded dictionary rules. This unified biometric signature is converted into hardware security constraints and embedded into the scheduled fault-secured DSP IP design using the security constraints embedding block. In the next chapter, we also discuss the detection of pirated designs using the proposed methodology.

Chapter 6

Unified Biometrics signature generation using expandable encoded dictionary and signature embedding and detection process

The proposed methodology introduces a novel approach to protect fault-secured hardware IP core designs against IP piracy using a unified biometric-driven hardware security system with an encoded dictionary. This methodology is based on the concept of exploiting unified biometrics to extract hardware security constraints and enable detective control against the use of pirated IP cores. The proposed approach unifies an IP vendor's palmprint, facial and fingerprint biometric signatures to generate a unique and non-replicable hybrid feature set that is used to produce an invisible unified biometric security mark. The proposed approach also includes an expandable encoded dictionary that adds additional layers of security to the generation of unified biometric-driven secret security constraints for embedding into the design.

In the previous chapter, we discussed the generation of biometric signatures of palmprint, facial and fingerprint and also demonstrated it using a DSP IP core (i.e., IDCT 8-point DSP IP core) with the help of the hardware security tool which is designed based on the proposed approach. In this chapter, we will discuss the expandable encoded dictionary and its significance, the generation of unified biometric-driven secret security constraints, and the embedding process of security constraints into the design. For demonstration purposes, we are going to continue with the example IDCT 8-point DSP IP core which we used in the previous chapter and the hardware security tool is used to generate the unified biometric signature using the proposed expandable encoded dictionary and also to

generate the secret hardware security constraints from the unified biometric signature.

6.1 Proposed Expandable Encoded Dictionary

The proposed methodology for protecting fault-secured DSP IP core designs against IP piracy includes an encoded dictionary block that plays a crucial role in generating the final signature to be embedded in the design. The encoded dictionary is created by the IP designer and consists of encoding rules and encoding bits, it is designed to accept the generated unified biometrics signature and select the final signature to be embedded with the designer-selected strength and combination. The encoded dictionary is expandable, and the size can be adjusted based on the need of the designer. The encoding rules corresponding to encoding bits can be created to generate a unique combination of unified biometrics signatures of various strengths. In Table. 6.1, an example of an encoded dictionary is shown, which displays eight different encoding rules for selecting a unique combination of unified biometrics signature of 75-bit signature strength (signature chosen for demonstration). An IP designer can choose the target unified biometrics signature of the desired strength and combination based on the selection of the encoding bits. Once the designer has selected the signature, it is embedded into the target design. The details of the signature embedding process are discussed in the next subsection.

Considering, an IP designer has chosen an encoded unified biometric signature with encoding bits as “001”, by selecting it in the input panel of the hardware security tool (shown in Fig. 6.1). Once encoding bits are selected from the input panel, based on the rule associated with the encoding bits as shown in the Table. 6.1, the final encoded unified biometric signature is generated. In our case, the rule associated with the encoding bit “011” is to concatenate the first even 25 bits of all three

Table. 6.1 Encoded dictionary for 3-bits (N=3) (expandable upto 2^N encoding rules)

Encoding bits	Encoding rules
000	Concatenate first 25 bits of all three (palmprint, facial and fingerprint) biometric signatures.
001	Concatenate first even 25 bits of all three (palmprint, facial and fingerprint) biometric signatures.
010	Concatenate all three palmprint, facial and fingerprint signatures and consider the first 75 prime indexed positions in the final signature.
011	Concatenate first odd 25 bits of all three (palmprint, facial and fingerprint) biometric signatures.
100	Concatenate all three palmprint, facial and fingerprint signature in bitwise manner and consider the first 75 prime indexed positions in the final signature.
101	Concatenate last even 25 bits of all three (palmprint, facial and fingerprint) biometric signatures.
110	Concatenate last odd 25 bits of all three (palmprint, facial and fingerprint) biometric signatures.
111	Concatenate last 25 bits of all three (palmprint, facial and fingerprint) biometric signatures.

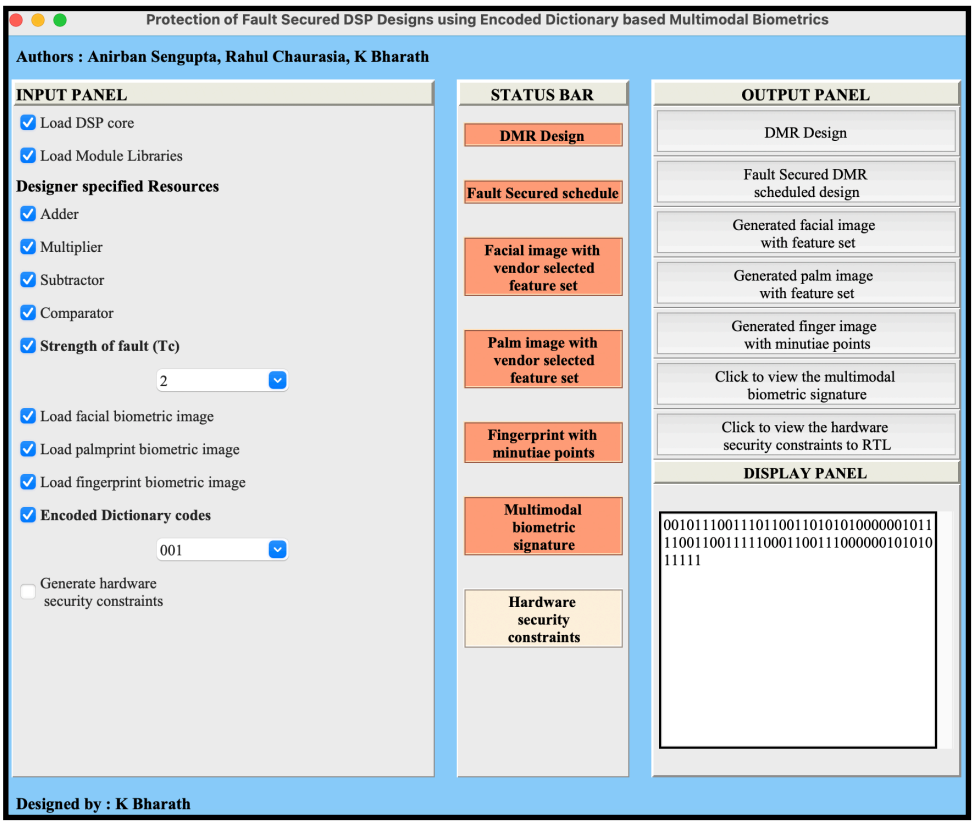


Fig. 6.1 Screenshot of hardware security tool corresponding to the encoded unified biometric signature

(palmprint, facial and fingerprint) biometric signatures (The generation of all three biometric signatures is described in the previous chapter and we are going to use the biometric signatures generated in the previous chapter for generating the encoded unified biometric signature). As shown in the output panel of Fig. 6.1, the final encoded unified biometric signature consists of 75 bits and is represented by the following binary sequence: "001011100111011001101010100000010111100110011111000110011100 000010101011111". The sequence contains 36 zeros and 39 ones. Then, the resulting multimodal biometric signature is incorporated into the target design. Further details regarding the embedding process of the signature are discussed in the following section.

6.2 Embedding unified biometric signature of IP vendor into the design

To safeguard the fault-secured DMR design from IP piracy, the IP designer embeds an encoded dictionary-based unified biometric signature into the target design which is also fault secured. The first step in the process of embedding the signature is generating hardware security constraints corresponding to the biometric signature. The hardware security constraints are generated based on the encoding rule specified by the IP designer and the DFG of the fault-secured DMR design schedule. The number of storage variables in the DFG dictates the number of security constraints formed based on the encoding rule. For example, if the signature bit '0' corresponds to embedding security constraints between even-even storage variable pairs (Vx, Vy), then the resulting security constraints for 36 zeros of the biometric signature are V(0, 2), V(0, 4), V(0, 6), V(0, 8), V(0, 10), V(0, 12), V(0, 14), V(0, 16), V(0, 18), V(0, 20), V(0, 22), V(0, 24), V(0, 26), V(0, 28), V(0, 30), V(0, 32), V(0, 34), V(0, 36), V(0, 38), V(0, 40), V(0, 42), V(0, 44), V(2, 4), V(2, 6), V(2, 8), V(2, 10), V(2, 12), V(2, 14), V(2, 16), V(2, 18), V(2, 20), V(2, 22), V(2, 24), V(2, 26), V(2, 28), V(2, 30). Similarly, if the signature bit '1' corresponds to

embedding security constraints between odd-odd storage variable pairs of the scheduled DFG, then the resulting security constraints for 39 1's of the biometric signature are V(1, 3), V(1, 5), V(1, 7), V(1, 9), V(1, 11), V(1, 13), V(1, 15), V(1, 17), V(1, 19), V(1, 21), V(1, 23), V(1, 25), V(1, 27), V(1, 29), V(1, 31), V(1, 33), V(1, 35), V(1, 37), V(1, 39), V(1, 41), V(1, 43), V(1, 45), V(3, 5), V(3, 7), V(3, 9), V(3, 11), V(3, 13), V(3, 15), V(3, 17), V(3, 19), V(3, 21), V(3, 23), V(3, 25), V(3, 27), V(3, 29), V(3, 31), V(3, 33), V(3, 35), V(3, 37). As shown in Fig. 6.3, the hardware security constraints are generated and displayed on the output display panel of the hardware security constraints after clicking the button "Generate hardware security constraints" in the input panel. Then these hardware security constraints are embedded into the target design during the resister allocation phase of behavioural synthesis to minimise the design overhead.

In the next step, the designer constructs the register allocation table comprising the details of storage variables, control steps, and register allocation information for the unprotected fault-secured DMR design. The designer then feeds the generated hardware security constraints and register allocation information as input to the security constraints embedding block, which outputs the unified biometric signature-protected RTL datapath of the fault-secured design. Local alterations are made among the registers to accommodate the security constraints, as per the distinct register assignment rule. If any security constraint is not adjustable amongst the available registers, a new register is allocated. The register allocation table of fault-secured IDCT-8 point DSP IP core before embedding the hardware security constraints is shown in Table. 6.2, as evident there are sixteen control steps (C0 - C15), sixteen registers (R1 - R16) and 46 storage variables (V0 - V45). After embedding the hardware security constraints generated from the dictionary-encoded unified biometric signature, local alterations take place in the register allocation table to resolve the raised conflict between any of the two registers. For

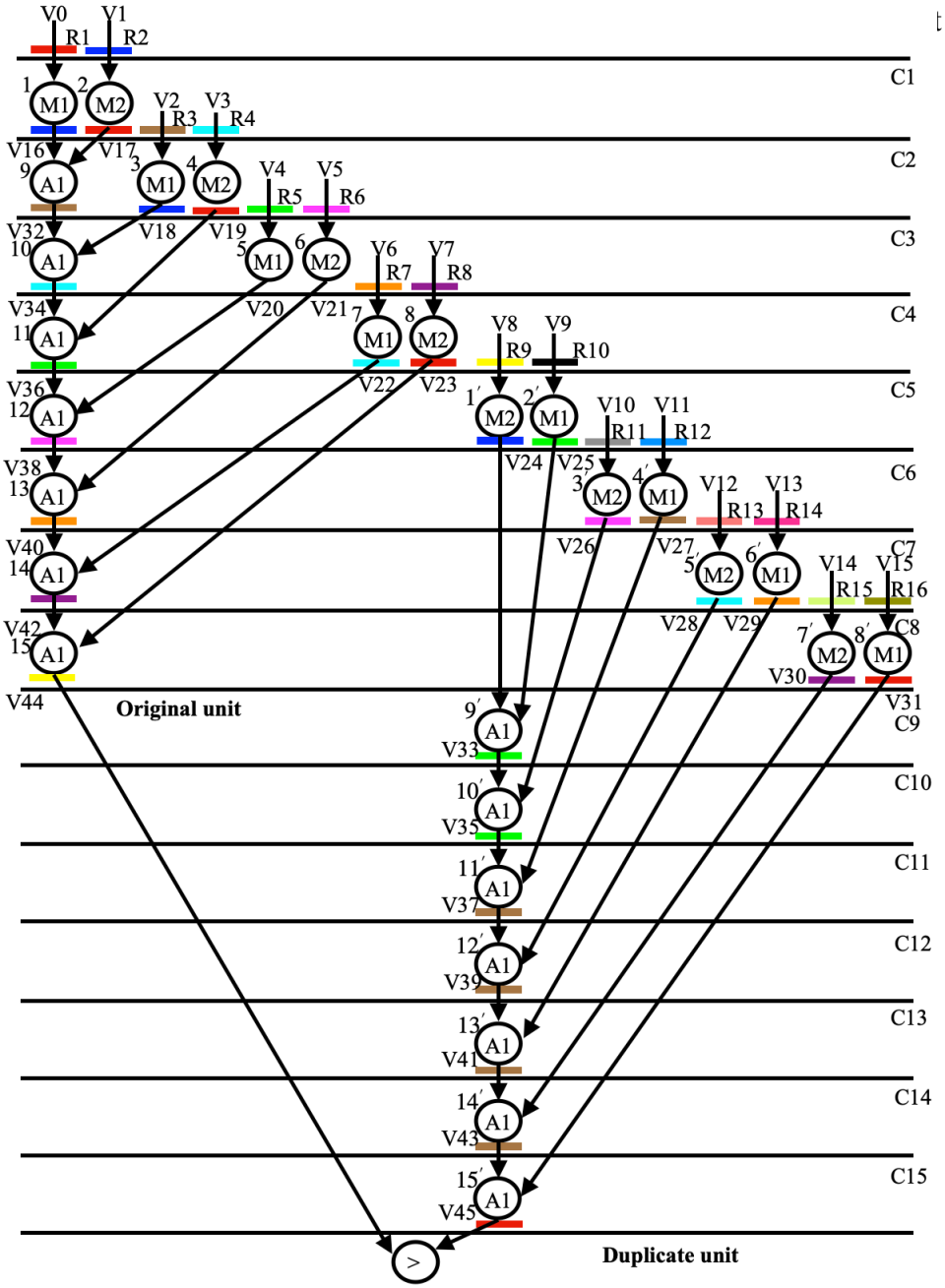


Fig. 6.2 Fault-secured scheduled IDCT filter design (post-embedding security constraints)

hence a conflict has been raised. To resolve this conflict local alterations take place between the storage variables V16 and V17, now V16 is assigned to register R2 (Blue) and V17 is assigned to register R1 (Red). Similarly, the designer embeds all the security constraints by making local alterations, and the resultant register allocation information is presented in

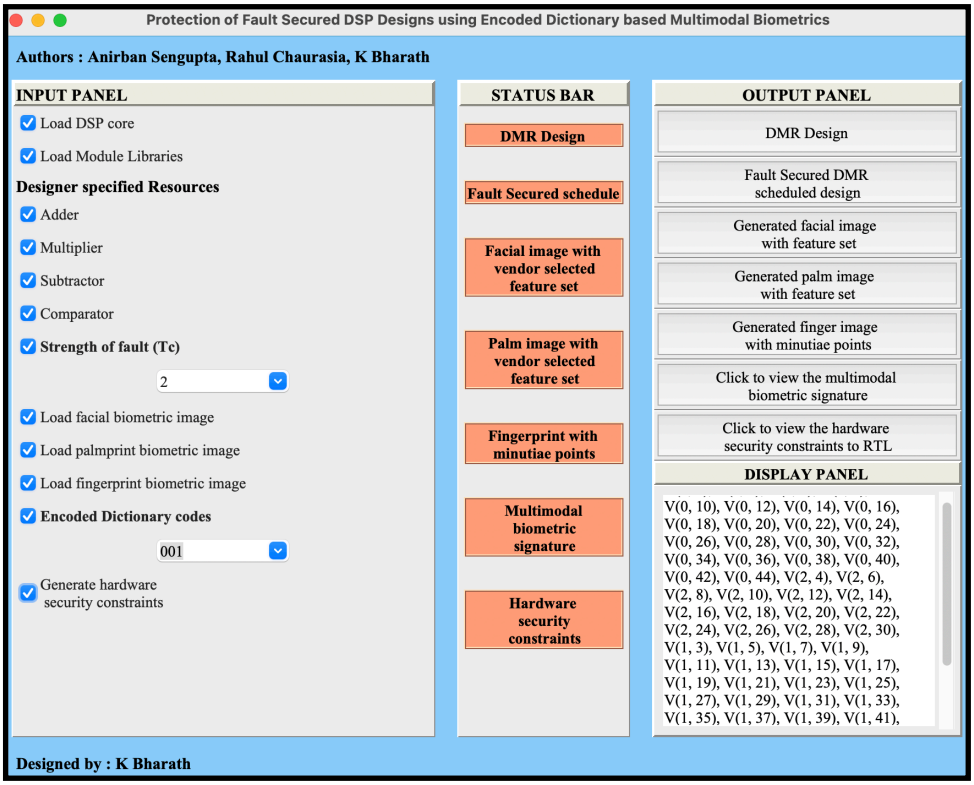


Fig. 6.3 Screenshot of hardware security tool corresponding to the hardware security constraints

Table. 6.2 Register allocation table for pre-embedding unified biometric signature into the design

Registers	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16
C0	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
C1	V16	V17	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
C2	V32	—	V18	V19	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
C3	V34	—	—	V19	V20	V21	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
C4	V36	—	—	—	V20	V21	V22	V23	V8	V9	V10	V11	V12	V13	V14	V15
C5	V38	—	—	—	—	V21	V22	V23	V24	V25	V10	V11	V12	V13	V14	V15
C6	V40	—	—	—	—	—	V22	V23	V24	V25	V26	V27	V12	V13	V14	V15
C7	V42	—	—	—	—	—	—	V23	V24	V25	V26	V27	V28	V29	V14	V15
C8	V44	—	—	—	—	—	—	—	V24	V25	V26	V27	V28	V29	V30	V31
C9	—	—	—	—	—	—	—	—	V33	—	V26	V27	V28	V29	V30	V31
C10	—	—	—	—	—	—	—	—	V35	—	—	V27	V28	V29	V30	V31
C11	—	—	—	—	—	—	—	—	V37	—	—	—	V28	V29	V30	V31
C12	—	—	—	—	—	—	—	—	V39	—	—	—	—	V29	V30	V31
C13	—	—	—	—	—	—	—	—	V41	—	—	—	—	—	V30	V31
C14	—	—	—	—	—	—	—	—	V43	—	—	—	—	—	—	V31
C15	—	—	—	—	—	—	—	—	V45	—	—	—	—	—	—	—

the Table. 6.3. The storage variables marked in red represent the local alteration performed after embedding the encoded dictionary-based unified biometrics signature (shown in Table. 6.3). Thus, the embedding of all the security constraints is performed to protect the fault-secured DMR design against IP piracy. The final fault-secured scheduled IDCT 8-point design post-embedding with encoded dictionary-based unified biometric signature is shown in Fig. 6.2.

Table. 6.3 Register allocation table for pre-embedding unified biometric signature into the design

Registers	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16
C0	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
C1	V17	V16	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
C2	V19	V18	V32	—	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
C3	V19	V20	V21	V34	—	—	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
C4	V23	V20	V21	V22	V36	—	—	—	V8	V9	V10	V11	V12	V13	V14	V15
C5	V23	V24	V21	V22	V25	V38	—	—	—	—	V10	V11	V12	V13	V14	V15
C6	V23	V24	V27	V22	V25	V26	V40	—	—	—	—	—	V12	V13	V14	V15
C7	V23	V24	V27	V28	V25	V26	V29	V42	—	—	—	—	—	—	V14	V15
C8	V31	V24	V27	V28	V25	V26	V29	V30	V44	—	—	—	—	—	—	—
C9	V31	—	V27	V28	V33	V26	V29	V30	—	—	—	—	—	—	—	—
C10	V31	—	V27	V28	V35	—	V29	V30	—	—	—	—	—	—	—	—
C11	V31	—	V37	V28	—	—	V29	V30	—	—	—	—	—	—	—	—
C12	V31	—	V39	—	—	—	V29	V30	—	—	—	—	—	—	—	—
C13	V31	—	V41	—	—	—	—	V30	—	—	—	—	—	—	—	—
C14	V31	—	V43	—	—	—	—	—	—	—	—	—	—	—	—	—
C15	V45	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

6.3 Detection of pirated design using the proposed methodology

The proposed approach for piracy detection involves regenerating security constraints from register allocation information of the target RTL design, followed by matching the extracted secret constraints with the original security constraints of the true IP designer. The multimodal biometric information of the IP vendor does not need to be recaptured during the matching process, as the original pre-stored biometric information is used instead. The biometric features dimensions, digital

template, and associated security constraints can be accurately recomputed from the pre-stored images, making the detection process independent of recapturing biometric information. If a 100% match is found between the extracted security constraints and the original pre-stored unified hardware security constraints, the design is considered genuine; otherwise, it is considered to be a pirated design.

The proposed technique ensures that an adversary cannot evade the piracy detection process by regenerating the exact unified biometric security constraints the security parameters are unknown to them. The IP vendor does not need to store their digital template (secret hardware security constraints), and the captured biometrics are safely stored in a secure vault. Even if the pre-stored biometric images of the IP vendor are leaked to an adversary, the exact regeneration of the digital template and its respective secret hardware security constraint is not possible without knowledge of the security parameters.

6.4 Security properties of encoded dictionary-based unified biometrics

The proposed methodology includes multiple security parameters for the generation of unified biometric security constraints and their embedding into the target design. These parameters enhance the overall security of the target DSP design against IP piracy. The security parameters include:

- **Non-replicability**:

The proposed methodology incorporates security measures that make it impossible for an adversary to replicate the naturally unique biometrics-driven secret hardware security constraints. This sets it apart from non-biometric approaches like hardware watermarking and steganography, which generate arbitrary security constraints. The use of palmprint, facial and fingerprint features to generate a unique biometric signature further complicates matters for adversaries attempting to embed a

fake IP and evade piracy detection. The robustness of the proposed unified biometrics-driven signature makes it highly challenging for an adversary to regenerate. This is because the security parameters required for regeneration are unknown and inaccessible to them. The following are the security parameters:

- *Grid size/spacing:* After capturing the palmprint, facial and fingerprint biometric, it is subjected to a specific grid size and spacing for generating the biometric information accurately. The details of the original grid size are not known to an adversary.
- *The number of biometric features and their concatenation order used for a signature generation:* An adversary is not aware of the total number of palm features selected for palmprint biometric, facial features selected for facial biometric, and the number of minutiae points selected for fingerprint biometric in the generation of the unified biometric driven signature. Additionally, the feature concatenation order used for generating the digital template is unknown to an adversary.
- *Encoding rule:* The original encoding rule used for generating the secret security constraints corresponding to the unified biometric-driven signature is not known to an adversary. This encoding rule is a key factor in ensuring the uniqueness and non-replicability of the generated security constraints, making it difficult for an adversary to regenerate the same constraints for embedding into a fake IP and evading piracy detection.
- *Encoded dictionary bit size and applied encoding rule:* The details of the encoded dictionary, including the size of the encoding bits (N), coded data bits (2N), and the encoding rules used to derive the unified biometric signature bitstream that is embedded into the design, are all unknown to an adversary.

Therefore, the security constraints generated using multiple parameters and encoding rules are unknown to an adversary, making it highly difficult for them to replicate the original security constraints embedded into the

design. This ensures that the IP piracy detection process is robust and an adversary cannot evade it.

- **Robustness against the compromising of biometric image data*:***

  Even if an adversary gains access to the stored original multimodal biometric images, they would not be able to regenerate the exact unified biometric signature that was embedded into the design. This is because they would not know the specific grid size and spacing used for generating the biometric information, the total number of features selected for each biometric, the feature concatenation order chosen for generating the digital template, the original encoding rule used for generating the secret security constraints, and the details of the encoded dictionary such as the size of encoding bits and coded data bits. Without this information, an adversary cannot replicate the original unified biometric signature, making it impossible to evade the IP piracy detection process.

- **Robustness against key-based attacks*:***

  The proposed unified biometric approach for DSP design security does not depend on secret keys for its operation, unlike other hardware security approaches such as digital signature and hardware steganography. The security is achieved through the use of a unified biometric signature that is generated from the palmprint, facial and fingerprint biometric information of the IP vendor. This signature is unique and highly robust, making it difficult for an adversary to replicate or regenerate. The approach incorporates several security parameters that are unknown to an adversary, making it highly challenging for them to evade the piracy detection process.

- The unified biometric-driven signature proposed here offers a higher level of resistance to tampering and a lower probability of coincidence, thereby providing strong protection against tampering attempts and enabling the detection of counterfeit IP cores.

Chapter 7

Results and Discussion/Analysis

7.1 Results and analysis of the proposed quadruple phase watermarking approach

The proposed approach was subjected to a thorough analysis of its security and design cost. To evaluate its security, two measures were used — the probability of coincidence and tamper tolerance ability. These measures help to determine the effectiveness of the proposed approach in detecting and resisting malicious attacks or attempts to alter the data. The design cost of the proposed approach was analysed in terms of trade-offs between cost and partitioning, and the cost overhead as compared to the baseline design. A 15 nm open-cell library [39] was used to calculate the design cost. This library is a commonly used resource for designing integrated circuits and offers a range of design options and optimisation techniques.

The proposed approach was implemented and tested on various Digital Signal Processing (DSP) benchmarks. The discrete cosine transform (DCT) core, for example, is a DSP algorithm used in the Joint Photographic Experts Group (JPEG) compression process to convert image data from the spatial domain to the frequency domain. Similarly, finite impulse response (FIR) and infinite impulse response (IIR) filters are DSP algorithms used for noise cancellation or denoising to improve signal quality in telecommunication. The experimental results of the proposed approach were evaluated to assess its efficiency and effectiveness in securing data. The implementation run time (or time overhead) of the proposed approach was found to be around 2.5 ms, indicating that it could be implemented relatively quickly. Furthermore, the proposed approach

was found to be amenable to other DSP and multimedia applications. This is because these applications also have algorithmic descriptions, and their corresponding intellectual property (IP) can be designed using the high-level synthesis (HLS) process. Hence, the proposed security algorithm can easily be employed to secure such IPs, making it a useful tool for securing a wide range of applications.

Security analysis:

The proposed approach provides security by incorporating a strong signature or digital watermark into the design to facilitate authentic IP verification. The quality of the watermark, which is essentially the strength of the digital evidence embedded into the design and the strength of the proof of IP ownership, is evaluated in terms of a metric called the probability of coincidence (P_c). This metric helps to measure the effectiveness of the watermarks by assessing the probability of coincidence, which is a measure of how difficult it is for an attacker to create a false watermark that matches the original one. The P_c is given as follows:

$$P_c = (1 - \frac{1}{c})^{f1} * (\frac{1}{\pi_{i=1}^{K} U(Zi)})^{f2} * (\frac{1}{2})^{f3} * \pi_{j=1}^{f4} (\frac{1}{\mu(x_j)}) \quad (1)$$

In equation (1), the first, second, third, and fourth terms represent the probability of coincidence (P_c) with respect to register binding, function unit (FU) binding, interconnect binding, and scheduling phases, respectively. In the first term, ‘c’ and ‘f1’ represent the number of colours or registers in the coloured-interval-graph (CIG) pre-embedding register binding constraints and the number of constraint edges, respectively. In the second term, ‘K’, ‘U(Zi)’, and ‘f2’ represents the number of types of FU resources, the number of instances of FU type Zi, and the number of FU binding constraints, respectively. In the third term, ‘f3’ represents the number of interconnect binding constraints, and in the fourth term, ‘f4’

represents the number of scheduling constraints. The symbol '$\mu(x_j)$' represents the mobility of operation 'x_j' which is subject to the imposition of the jth scheduling constraint, and 'x_j' indicates the corresponding operation.

Table. 7.1.1 presents the value of P_c achieved using the proposed watermarking technique for varying signature sizes. Tables. 7.1.1 and 7.1.2 compare the P_c value obtained using the proposed approach with the related approaches [40, 11, 12, 15, 17] for the same signature size. The tables show that the proposed approach achieves lower P_c values than the related works. This is because the proposed watermarking constraints are embedded into the form of different phases of the high-level synthesis (HLS) process, unlike the related approaches. The low P_c value obtained using the proposed approach indicates a high quality of the embedded watermark and a higher strength of digital evidence embedded into the designs for IP ownership verification or piracy detection. Additionally, Fig. 7.1.1 shows the variation in P_c of the proposed approach with varying numbers of embedding phases. The figure demonstrates that the P_c value gradually decreases as the number of embedding phases increases. Further, the strength of the watermark is evaluated based on its ability to withstand tampering, which is measured by a metric known as tamper tolerance (T^P), as defined below:

$$T^P = Q^L * 2^B * M \tag{2}$$

where Q and L are variables that represent the number of variable types and the length of encoding, respectively, with both being set to eight in the proposed approach. B denotes the total number of bits in the signature and M represents the number of mapping rules, also set to eight

in the proposed approach. The length of encoding (L) depends on the design size, while the total bits in the signature (B) depend on the chosen

Table. 7.1.1 Probability of coincidence (Pc) analysis of proposed approach w.r.t. related approaches [11,12,15].

DSP benchmarks	Signature size (# of triads	P_c				#times lower P_c than [15]	#times lower P_c than [12]	#times lower P_c than [11]
		Proposed	[15]	[12]	[11]			
DCT	20	7.6e-7	6.9e-2	1.0e-5	6.9e-2	9.0e+4	1.3e+1	9.0e+4
	25	6.4e-8	3.5e-2	1.5e-6	3.5e-2	5.4e+5	2.3e+1	5.4e+5
	30	1.7e-10	1.8e-2	2.8e-6	1.8e-2	1.0e+8	1.6e+4	1.0e+8
FFT	20	2.3e-6	2.7e-1	5.4e-5	2.7e-1	1.1e+5	2.3e+1	1.1e+5
	26	1.1e-8	1.8e-1	5.1e-7	1.8e-1	1.6e+7	4.6e+1	1.6e+7
	32	6.0e-11	1.2e-1	4.9e-9	1.2e-1	2.0e+9	8.1e+1	2.0e+9
IIR	20	3.1e-4	2.2e-1	1.9e-2	2.2e-1	7.0e+2	6.1e+1	7.0e+2
	26	1.6e-5	1.4e-1	6.5e-3	1.4e-1	8.7e+3	4.0e+2	8.7e+3
	32	1.oe-5	9.3e-2	1.2e-3	9.3e-2	9.3e+3	1.2e+2	9.3e+3
FIR	22	1.3e-9	5.3e-2	1.3e-6	5.3e-2	4.0e+7	1.0e+3	4.0e+7
	34	3.8e-13	1.0e-2	1.9e-8	1.0e-2	2.6e+10	5.0e+4	2.6e+10
	46	7.4e-20	2.1e-3	1.0e-13	2.1e-3	2.8e+16	1.3e+6	2.8e+16
ARF	22	1.0e-9	2.4e-1	2.4e-8	2.4e-1	2.4e+8	2.4e+1	2.4e+8
	34	3.0e-15	1.1e-1	4.7e-13	1.1e-1	3.6e+13	1.5e+2	3.6e+13
	46	1.5e-17	5.1e-2	2.8e-14	5.1e-2	3.4e+15	1.8e+3	3.4e+15
1D-DWT	20	9.2e-6	2.6e-2	4.2e-5	2.6e-2	2.8e+3	4.5e+0	2.8e+2
	25	5.9e-7	1.0e-2	2.7e-6	1.0e-2	1.6e+4	4.5e+0	1.6e+4
	30	3.8e-8	4.2e-3	1.7e-7	4.2e-3	1.1e+5	4.4e+0	1.1e+5
MPEG	20	3.3e-11	2.2e-1	2.1e-10	2.2e-1	6.6e+9	6.3e+0	6.6e+9
	30	1.1e-15	1.0e-1	1.3e-14	1.0e-1	9.0e+13	1.1e+1	9.0e+13
	40	4.0e-20	5.1e-2	8.7e-19	5.1e-2	1.2e+18	2.1e+1	1.2e+18

Table. 7.1.2. Comparison of P_C of the proposed approach with [40, 17].

Benchmarks	Proposed	[17]	[40]
DCT	1.7e1-0	1.8e-2	2.6e-1
FFT	6.0e-11	1.2e-1	5.2e-1
IIR	1.0e-5	9.3e-2	4.7e-1
FIR	7.4e-20	2.1e-3	2.6e-1
ARF	1.5e-17	5.1e-2	5.2e-1
1D-DWT	3.8e-8	4.2e-3	1.6e-1
MPEG	4.0e-20	5.1e-2	4.7e-1

signature size. The three terms in the T^P formula indicate security due to eight-variable encoding, hashing, and eightfold mapping, respectively. The

proposed approach's tamper tolerance ability has presented in the Table. 7.1.3 and compared with other approaches [11, 12, 15]. The results show that the proposed approach achieves higher tamper tolerance compared to the related approaches, making it more difficult for attackers to deduce or tamper with the author's signature. This prevents attackers from claiming IP ownership by circumventing counterfeit detection processes by embedding authentic signatures in counterfeit designs.

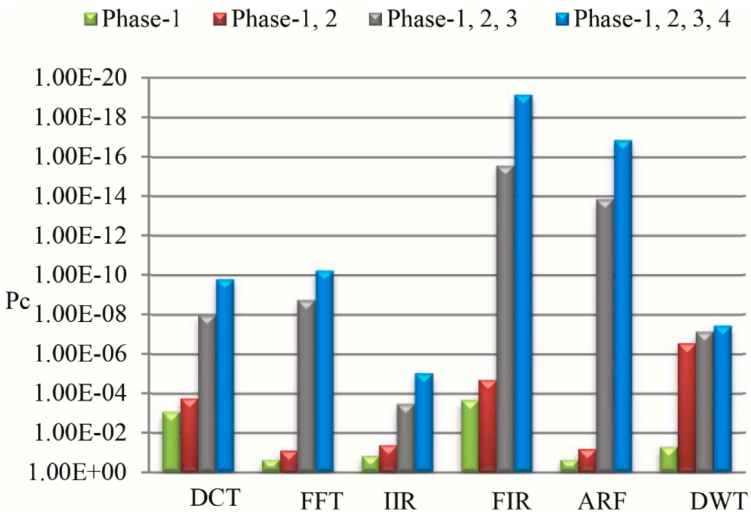


Fig. 7.1.1 Variation in Pc due to embedding watermark during different phases.

Design cost analysis and security-cost tradeoff

The design cost C_t is evaluated as follows:

$$C_t = a_1 \frac{L_h}{L_m} + a_2 \frac{A_h}{A_m} \tag{3}$$

Where A_h, L_h, A_m and L_m are the design area, latency, maximum area and maximum latency respectively. a_1 and a_2 are the weight contribution of latency and area in the design cost.

The design area is calculated as follows:

$$A_H = \sum_{i=1}^{K} U(Zi) * A_{Zi} \quad (4)$$

Where K and U(Zi) denote the number of types of FU resources and the number of instances of FU type Zi respectively.

Table. 7.1.3 Tamper tolerance (T^P) analysis of proposed approach w.r.t. related approaches [11], [12], [15]

DSP benchmarks	Signature size (# of triads)	Tamper tolerance (T^P)				#times lower T^P than [15]	#times lower T^P than [12]	#times lower T^P than [11]
		Proposed	[15]	[12]	[11]			
DCT	20	9.9e+27	1.1e+12	7.9e+16	1.0e+6	9.0e+15	1.2e+11	9.9e+21
	25	3.2e+32	1.1e+15	1.3e+21	3.3e+7	2.9e+17	2.4e+11	9.7e+24
	30	1.4e+45	1.1e+18	2.2e+25	1.1e+9	1.2e+27	6.3e+19	1.2e+36
FFT	20	1.2e+27	1.1e+12	7.9e+16	1.0e+6	1.0e+15	1.5e+10	1.2e+21
	26	1.0e+37	4.5e+15	9.3e+21	6.7e+7	2.2e+21	1.0e+15	1.4e+29
	32	2.9e+51	1.8e+19	1.1e+27	4.2e+9	1.6e+32	2.6e+24	6.9e+41
IIR	20	4.7e+21	1.1e+12	7.9e+16	1.0e+6	4.2e+9	5.9e+4	4.7e+15
	26	6.3e+29	4.5e+15	9.3e+21	6.7e+7	1.4e+14	6.7e+7	9.4e+21
	32	5.8e+48	1.8e+19	1.1e+27	4.2e+9	3.2e+29	5.2e+21	1.3e+39
FIR	22	5.4e+39	1.7e+13	3.9e+18	4.2e+6	3.1e+26	1.3e+21	1.2e+33
	34	2.4e+52	2.9e+20	5.4e+28	1.7e+10	8.2e+31	4.4e+23	1.4e+42
	46	6.7e+66	4.9e+27	7.5e+38	7.0e+13	1.3e+39	8.9e+27	9.5e+52
ARF	22	5.4e+39	1.7e+13	3.9e+18	4.2e+6	3.1e+26	1.3e+21	1.2e+33
	34	3.7e+50	2.9e+20	5.4e+28	1.7e+10	1.2e+30	6.8e+21	2.1e+40
	46	1.0e+65	4.9e+27	7.5e+38	7.0e+13	2.0e+37	1.3e+26	1.4e+51
1D-DWT	20	7.9e+28	1.1e+12	7.9e+16	1.0e+6	7.1e+16	1.0e+12	7.9e+22
	25	2.6e+33	1.1e+15	1.3e+21	3.3e+7	2.3e+18	2.0e+12	7.8e+25
	30	8.5e+37	1.1e+18	2.2e+25	1.1e+9	7.7e+19	3.8e+12	7.7e+28
MPEG	20	1.0e+37	1.1e+12	7.9e+16	1.0e+6	9.0e+24	1.2e+20	1.0e+31
	30	1.1e+46	1.1e+18	2.2e+25	1.1e+9	1.0e+28	5.0e+20	1.0e+37
	40	1.2e+55	1.2e+24	6.3e+33	1.1e+12	1.0e+31	1.9e+21	1.0e+43

The design latency is determined by analysing the scheduling information of the operations that are scheduled in various control steps. The calculation of design latency is based on the following formula:

$$L_h = R_L + \sum_{i=1}^{T} (L_m^i + R_L) \quad (5)$$

Where R_L represents the delay of a register, T denotes the total number of control steps and L_m^i indicates the delay of the FU with the maximum latency in the *i*th control step. Table. 7.1.4 presents the design cost before and after watermark embedding for a fixed signature size. The table shows zero design cost overhead for most DSP benchmarks. However, for some designs, there may be a slight increase in design cost due to an increase in latency after embedding scheduling constraints. Fig. 7.1.2 illustrates the trade-off between design cost and security (measured in P_c) for a fixed partition type and varying signature size. The figure demonstrates that the P_c value significantly decreases with increasing signature size, with little to no impact on design cost.

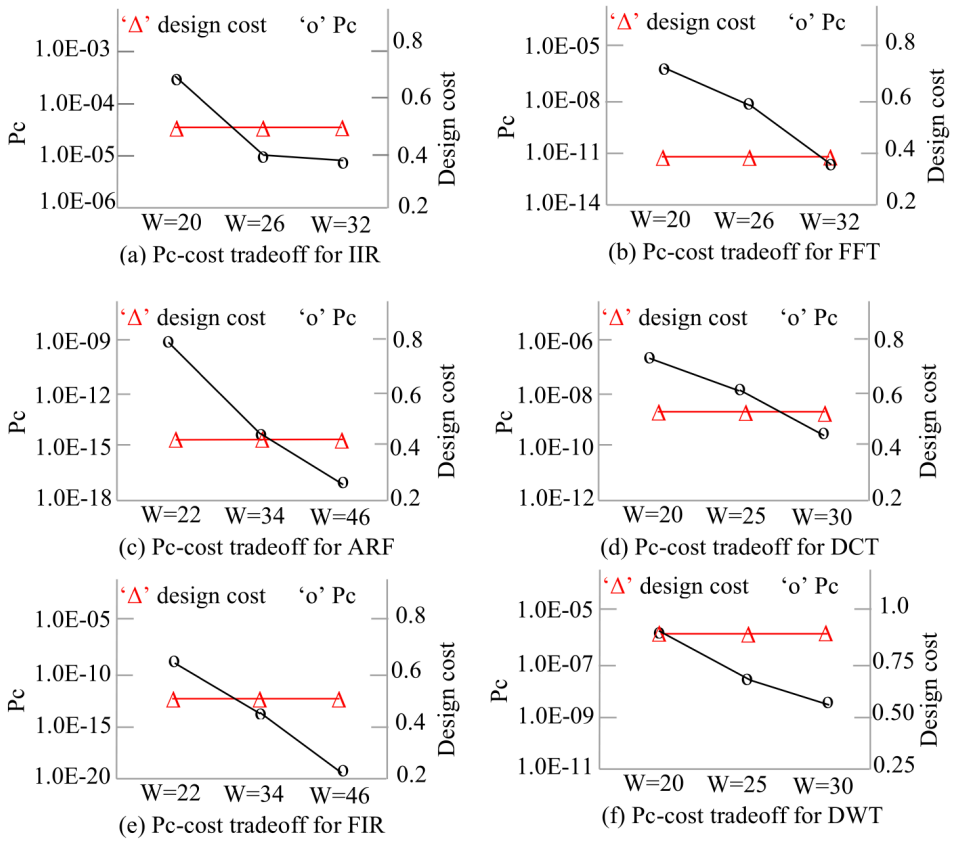


Fig. 7.1.2 Security (in terms of Pc)-cost tradeoff for various benchmarks.

Table. 7.1.4 Design cost pre and post-embedding of the proposed watermark.

Benchmarks	Design cost of baseline	Design cost of proposed	%cost overhead
DCT	0.497	0.537	8.0%
FFT	0.395	0.395	0.0%
IIR	0.522	0.522	0.0%
FIR	0.461	0.494	7.1%
ARF	0.408	0.408	0.0%
1D-DWT	0.851	0.851	0.0%
MPEG	0.370	0.370	0.0%

Impact and analysis of portioning on P_c and design cost

Fig. 7.1.3 depicts the impact of selecting three different partition types (X, Y, and Z) on design cost and P_c. The figure illustrates that choosing different partition types can have varying effects on design costs. However, there is a negligible impact on P_c for a fixed signature size. This allows designers to select the partition type that results in the least design cost overhead.

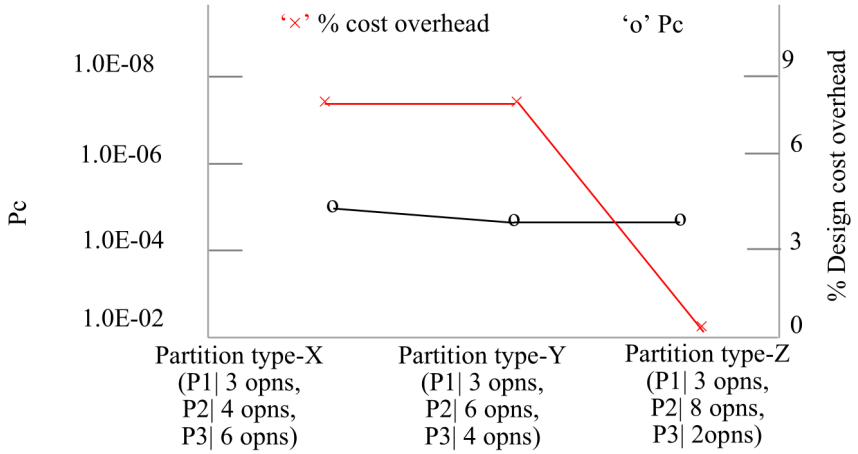


Fig. 7.1.3 Partitioning-cost trade-off for IIR filter core for signature size=32

Impact of proposed mapping of signature triads into corresponding constraints on security and design cost.

The signature is transformed into watermarking constraints using an eightfold mapping proposal. The constraints are then embedded into four different phases of HLS, resulting in a significant improvement in security concerning P_c and tamper tolerance. The mapping of signature triads to FU vendor binding and interconnect binding constraints does not affect design cost as no additional resources are required. However, the mapping of triads into register binding constraints may result in a minimal increase in design overhead due to the potential need for additional registers. The mapping of signature triads into scheduling constraints may sometimes cause a delay overhead, thereby affecting the latency of the design.

7.2 Results and analysis of the proposed unified biometric driven hardware security methodology

In this section, the outcomes of the proposed hardware security method, which utilises an encoded dictionary-based unified biometric approach for safeguarding fault-secured DSP IP cores, are examined. The method was developed using Python programming language and implemented on a processor with a 2.40 GHz frequency.

Security analysis:

The unified biometric signature that is embedded in the DSP design using an encoded dictionary-based approach is non-replicable. This hardware security methodology utilising a unified biometric approach provides strong protection against IP piracy and prevents an adversary from evading the piracy detection process. The reason for this is that it is not feasible for an adversary to reproduce the exact signature and corresponding security constraints, due to several security parameters that are integrated during the embedding process of the unified biometric signature. The proposed approach's ability to protect against the threat of IP

piracy is evaluated by examining the probability of coincidence (Pb_c) and tamper tolerance (TT). The probability of coincidence (Pb_c) is measured using the following metric [11]:

$$Pb_c = (1 - \frac{1}{k})^w \tag{6}$$

'k' represents the number of registers required to store all the input, intermediate, and output variables of the target design before implanting secret constraints, while 'w' represents the number of covert security constraints generated for the proposed unified biometric signature embedded in the design. A low probability of coincidence is desirable as it indicates a lower likelihood of detecting security constraints in an unsecured design. The comparison of the Pb_c values achieved using our proposed unified biometric-driven hardware security methodology with IP watermarking [11], hardware steganography [17], unimodal palmprint biometric [26], and unimodal fingerprint biometric [28] approaches for various DSP frameworks are presented in Table. 7.2.1 and Table. 7.2.2.

Table. 7.2.1 Comparison of Pb_c of proposed unified biometrics approach w.r.t related works [11], [17].

Benchmarks	Proposed approach		Watermarking [11]		Steganography [17]	
	Security constraints	Pb_c	Security constraints	Pb_c	Security constraints	Pb_c
DCT-8 point	450	2.4e-13	15	3.7e-1	43	6.2e-2
IDCT-8 point	450	2.4e-13	30	1.4e-1	125	3.1e-4
JPEG sample	686	2.1e-13	60	7.7e-2	116	7.1e-3
MESA	686	7.5e-4	120	2.8e-1	159	1.8e-1
WDF	686	1.1e-26	240	8.5e-10	86	5.6e-4

As shown in Table. 7.2.1, the proposed encoded unified biometric approach generates a greater number of secret security constraints, resulting in a lower Pb_c value compared to related approaches such as IP watermarking [11] and hardware steganography [17]. Similarly, as illustrated in Table. 7.2.2, the proposed approach achieves a lower Pb_c

value compared to related approaches like unimodal palmprint biometrics [26] and unimodal fingerprint biometrics [28].

Table. 7.2.2 Comparison of Pb_c of proposed unified biometrics approach w.r.t related works [26], [28].

Benchmarks	Proposed approach		Palmprint biometric [26]		Fingerprint biometric [28]	
	Security constraints	Pb_c	Security constraints	Pb_c	Security constraints	Pb_c
DCT-8 point	450	2.4e-13	255	7.1e-8	350	1.5e-10
IDCT-8 point	450	2.4e-13	255	7.1e-8	350	1.5e-10
JPEG sample	686	2.1e-13	255	1.9e-5	350	3.3e-7
MESA	686	7.5e-4	255	6.9e-2	350	2.5e-2
WDF	686	1.1e-26	255	2.3e-10	350	6.0e-14

The tamper tolerance metric is used to evaluate the security against tampering attempts aimed at determining the exact signature combination. Our proposed multi-modal biometric signature approach achieves higher tamper tolerance compared to related approaches. The tamper tolerance ability (TT) is assessed using the following metric [11]:

$$TT = (\tau)^{\omega} \tag{7}$$

The tamper tolerance ability of our proposed approach is compared with related methodologies in Table. 7.2.3 and Table. 7.2.4, where 'τ' represents the number of signature variables used in the multimodal biometric signature. Due to the generation and embedding of a significantly higher number of secret security constraints through the proposed approach, the tamper tolerance ability is much stronger than the related methodologies [11, 17, 26, 28]. Table. 7.2.5. shows the impact of varying the multi-modal signature on Pb_c and TT. The proposed approach generates a greater number of secret security constraints, resulting in a lower Pb_c value and higher TT value, providing strong digital evidence against IP piracy and robust security against tampering aimed at determining the exact embedded 'encoded dictionary-based unified biometrics signature'.

Table. 7.2.3 Comparison of TT of proposed unified biometrics approach w.r.t related works [11], [17].

Benchmarks	Proposed approach		Watermarking [11]		Steganography [17]	
	Security constraints	TT	Security constraints	TT	Security constraints	TT
DCT-8 point	450	2.9e+135	15	3.2e+4	43	8.7e+12
IDCT-8 point	450	2.9e+135	30	1.0e+9	125	4.2e+37
JPEG sample	686	3.2e+206	60	1.1e+18	116	8.3e+34
MESA	686	3.2e+206	120	1.3e+36	59	5.7e+17
WDF	686	3.2e+206	240	1.7e+72	86	7.7e+25

Table. 7.2.4 Comparison of TT of proposed unified biometrics approach w.r.t related works [26], [28].

Benchmarks	Proposed approach		Palmprint biometric [26]		Fingerprint biometric [28]	
	Security constraints	TT	Security constraints	TT	Security constraints	TT
DCT-8 point	450	2.9e+135	255	5.7e+76	350	2.2e+105
IDCT-8 point	450	2.9e+135	255	5.7e+76	350	2.2e+105
JPEG sample	686	3.2e+206	255	5.7e+76	350	2.2e+105
MESA	686	3.2e+206	255	5.7e+76	350	2.2e+105
WDF	686	3.2e+206	255	5.7e+76	350	2.2e+105

Table. 7.2.5 PC, TT of the proposed approach corresponding to varying signature size for 8-point DCT application.

Security constraints	Pb_c	TT
75	7.9e-3	3.77e+22
200	2.4e-6	1.6e+60
350	1.5e-10	2.2e+105
500	9.6e-15	3.2e+150
686	5.91e-20	3.2e+206

Analysis of embedded design cost:

The cost of the unified biometric-driven signature embedded design is analysed in this subsection, specifically about the unsecured baseline design. The design cost $D_c(s_n^t)$ of the fault-secured DSP design embedding proposed security constraints is computed using the following [17]:

$$D_c(s_n^t) = \tau 1 \frac{D_A}{D_{maxA}} + \tau 2 \frac{D_T}{D_{maxT}} \quad (8)$$

where 's_n^t' indicates the resource constraints (where 'n' specifies the number of resources and 't' specifies the type of resources), 'D_A' and 'D_T' indicates the design area and latency respectively, 'D_{maxA}' and 'D_{maxT}' indicates the maximum design area and latency of the design. The weighing factors '$\tau 1$' and '$\tau 2$' are used to determine the relative importance of normalised design area and latency in the cost function, and they indicate the priority given by the IP vendor to these factors during the cost evaluation process. In this case, the weighting factors for design area and latency are both assumed to be 0.5. The cost of generating the protected fault-secured design is presented in Table. 7.2.6, which provides details on the functional units, the number of required registers, and the design cost of embedding an encoded dictionary-based unified biometric signature. The use of this signature incurs no additional design cost for any DSP design. To estimate the delay and area of the design, a 15nm open-cell library is used.

Table. 7.2.6 Comparison of the design cost pre and post-embedding encoded dictionary-based unified biometric signature

	Pre-embedding unified biometrics signature into the design			**Post-embedding unified biometrics signature into the design**			**% overhead**
Benchmarks	**FUs**	**# of registers**	**Design cost**	**FUs**	**# of registers**	**Design cost**	
DCT-8 point	1+, 2*, 1>	16	0.436	1+, 2*, 1>	16	0.436	0.00%
IDCT-8 point	1+, 2*, 1>	16	0.436	1+, 2*, 1>	16	0.436	0.00%
JPEG sample	2+, 1*, 1>	24	0.522	2+, 1*, 1>	24	0.522	0.00%
MESA	4+, 4*, 1>	96	0.208	4+, 4*, 1>	96	0.208	0.00%
WDF	2+, 1*, 1>	12	0.522	2+, 1*, 1>	12	0.522	0.00%

Table. 7.2.7 Implementation run time of the proposed security methodology corresponding to different benchmarks (fault-secured)

Benchmarks	Implementation run time (ms)
DCT-8 point	6.852
IDCT-8 point	6.810
JPEG sample	19.279
MESA	107.027
WDF	14.526

Table. 7.2.7 presents the implementation run time for the proposed security methodology, which generates a secured version of the design using encoded dictionary-based unified biometric signatures. As shown, the proposed technique is capable of generating fault-secured designs with embedded biometric signatures in a relatively short implementation time.

Chapter 8

Conclusions and Scope for Future Work

The use of intellectual property (IP) cores in modern system-on-chip (SoC) designs has become increasingly prevalent. However, the globalisation of the design supply chain has made IP cores vulnerable to various hardware security threats, such as IP piracy, counterfeiting, and false claims of IP ownership. These threats can result in serious concerns for end consumers such as. To address these concerns, IP watermarking has emerged as a robust detective control mechanism that provides security to IP cores against these hardware security threats. Hardware watermarking involves embedding a unique digital signature, or watermark, into the IP core design, which can be used to identify the rightful owner and detect any attempts to tamper with or copy the IP core.

In this thesis, a novel quadruple-phase watermarking scheme has been proposed to secure digital signal processing (DSP) IP cores. The scheme employs mechanisms such as partitioning, encoding, hashing, and eightfold mapping in the signature generation process, making the signature constraints highly tamper-tolerant. The watermark is embedded into the design during four distinct phases: scheduling, the functional unit (FU), register binding, and interconnect binding, of the high-level synthesis (HLS) process. Embedding the watermark during these phases ensures a robust watermark, providing stronger ownership proof and higher strength of digital evidence embedded into the IP core designs.

Experimental analysis of the proposed quadruple-phase watermarking scheme was conducted in terms of probability of coincidence, tamper tolerance ability, the impact of embedding the signature on design cost overhead, and security-cost tradeoff. The results

(chapter 7) showed that the proposed approach outperformed related state-of-the-art works, achieving a significantly lower probability of coincidence and higher tamper tolerance.

Moreover, pirated IP cores that are integrated into hardware systems of consumer electronics (CE) products may pose a serious concern to the end consumer from the perspective of safety, non-reliability, and confidentiality. Therefore, a unified biometric-driven hardware security methodology has been presented to ensure robust piracy protection of DSP IP cores and safeguard the end consumer and critical systems that may have integrated pirated DSP IP cores.

An adversary may try to evade piracy detection by intentionally integrating fake IP cores in CE systems due to a lack of robust security mechanisms. A unified biometric-driven hardware security approach is presented to provide strong piracy protection for DSP IP cores, protecting both end consumers and critical systems that may have used pirated DSP IP cores. The proposed methodology uses an encoded dictionary to allow for the flexible selection of a robust signature, which significantly complicates the generation of secure security constraints from the attacker's perspective. This renders the attacker unable to extract the embedded signature and copies it into fake IP cores to evade piracy detection, ensuring the safety and reliability of CE systems for end consumers. In summary, the proposed quadruple-phase watermarking scheme and unified biometric-driven hardware security methodology provide a promising solution to protect hardware IP cores against IP piracy, counterfeiting, and false claim of IP ownership threats, ensuring the reliability and safety of consumer electronics products and critical systems.

While hardware watermarking is a useful technique for enabling detective control against IP piracy threats and IP ownership. On the other

hand, obfuscation is another important mechanism for enabling preventive control measures against reverse engineering attacks (RTL design alteration). Obfuscation involves modifying the design structure or implementation to hide the original IP and make it difficult for attackers to extract the functionality or design details [55]. Obfuscation can be achieved through various techniques, such as logic obfuscation or structural obfuscation, and can be applied at different levels of abstraction, from the register transfer level (RTL) to the high-level synthesis (HLS) level. In my future work, I will be focusing on exploring the double line of defence mechanism (ensuring both detective and preventive control) for securing hardware IP cores against hardware security threats. Further, generating low-cost and secure architectural solutions corresponding to different data-intensive hardware IPs and also analysing the trade-offs between security and performance or power consumption by proposing novel approaches to enhance the security of DSP designs while incurring negligible design cost overhead.

References

[1] A. Sengupta, R. Sedaghat, Z. Zeng, “Multi-objective efficient design space exploration and architectural synthesis of an application specific processor (ASP),” *Microprocess. Microsyst.*, vol. 35, no. 4, June 2011, pp. 392–404.

[2] J. Rajendran, H. Zhang, and O. Sinanoglu, “High-level synthesis for security and trust,” in *Proc. IEEE 19th Int. On-Line Testing Symposium*, Chania, Greece, 2013, pp. 1–6.

[3] M. Beaumont, B. Hopkins, and T. Newby, “Hardware trojans—Prevention, detection, countermeasures (a literature review),” Dept. of Defense, Defense Sci. and Technology Org., Australia, DSTO-TN-1012, 2011.

[4] A. Sengupta, “Protection of IP-core designs for CE products,” *IEEE Consum. Electron. Mag.*, vol. 5, pp. 83–89, Dec. 2015.

[5] 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.*, to be published. doi: 10.1109/TCAD.2016.2597232.

[6] Cui A, Chang CH, Tahar S. IP watermarking using incremental technology mapping at logic synthesis level. *IEEE Trans Comput Aided Des Integr Circuits Syst* 2008;27(9):1565–70.

[7] Cui A, Chang C. Watermarking for IP protection through template substitution at logic synthesis level. *In: Proceedings of the ISCAS;* 2007. p. 3687–90.

[8] Cui A, Chang C, Tahar S, Abdel-Hamid AT. A robust FSM watermarking scheme for IP protection of sequential circuit design. *IEEE Trans Comput Aided Des Integr Circuits Syst* 2011;30(5):678–90.

[9] Karmakar R, Chattopadhyay S. Hardware IP protection using logic

encryption and watermarking. *In: Proceedings of the IEEE international test conference (ITC)*. IEEE; 2020. p.
1–10.

[10] Karmakar R, Jana SS, Chattopadhyay S. A cellular automata guided finite-state-machine watermarking strategy for IP protection of sequential circuits. *IEEE Trans. Emerg. Top. Comput*. 2022;10(2):806–23. 1 April-June.

[11] Koushanfar F, Hong I, Potkonjak M. Behavioral synthesis techniques for intellectual property protection. *ACM Trans Des Autom Electron Syst* 2005;10(3): 523–45.

[12] Sengupta A, Roy D, Mohanty S P. Triple-phase watermarking for reusable IP core protection during architecture synthesis. *IEEE Trans Comput Aided Des Integr Circuits Syst* 2018; 37 (4): 742 – 55.

[13] Le Gal B, Bossuet L. Automatic low-cost IP watermarking technique based on output mark insertions. *Des Autom Embed Syst* 2012;16(2):71–92.

[14] Hong I, Potkonjak M. Behavioral synthesis techniques for intellectual property security. *In: Proceedings of the DAC*; 1999. p. 849–54.

[15] Sengupta A, Bhadauria S. Exploring low-cost optimal watermark for reusable IP cores during high level synthesis. *IEEE Access* 2016;4:2198–215.

[16] Sengupta A, Rathor M. Enhanced security of DSP circuits using multi-key based structural obfuscation and physical-level watermarking for consumer electronics systems. *IEEE Trans Consum Electron* 2020;66(2):163–72.

[17] Sengupta A, Rathor M. IP core steganography for protecting DSP kernels used in CE systems. *IEEE Trans Consum Electron* 2019;65(4):506–15.

[18] S. Rai, A. Rupani, P. Nath and A. Kumar, “Hardware Watermarking Using Polymorphic Inverter Designs Based On Reconfigurable

Nanotechnologies," 2019 *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 663-669.

[19] M. Shayan, K. Basu and R. Karri, "Hardware Trojans Inspired IP Watermarks," *IEEE Design & Test*, vol. 36, no. 6, pp. 72-79, Dec. 2019.

[20] J. Kuai, J. He, H. Ma, Y. Zhao, Y. Hou and Y. Jin, "WaLo: Security Primitive Generator for RT-Level Logic Locking and Watermarking," *2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 2020, pp. 01-06.

[21] T. Kean, D. McLaren, and C. Marsh, "Verifying the authenticity of chip designs with the designtag system," *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 59–64.

[22] G. T. Becker, M. Kasper, A. Moradi, and C. Paar, "Side-channel based watermarks for integrated circuits," *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 30–35.

[23] M. Rathor and A. Sengupta, "IP Core Steganography Using Switch Based Key-Driven Hash-Chaining and Encoding for Securing DSP Kernels Used in CE Systems," *IEEE Trans. Consum. Electron.*, vol. 66, no. 3, pp. 251-260, Aug. 2020.

[24] S. S. Zalivaka, A. A. Ivaniuk and C. -H. Chang, "Reliable and Modeling Attack Resistant Authentication of Arbiter PUF in FPGA Implementation with Trinary Quadruple Response," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 1109-1123, April 2019.

[25] Y. Lao, B. Yuan, C. H. Kim and K. K. Parhi, "Reliable PUF-Based Local Authentication With Self-Correction," *IEEE Trans. on Comput.-Aided Design of Integr. Circuits and Syst.*, vol. 36, no. 2, pp. 201-213, Feb. 2017.

[26] A. Sengupta, R. Chaurasia and T. Reddy, "Contact-Less Palmprint Biometric for Securing DSP Coprocessors Used in CE Systems," IEEE Trans. Consum. Electron., vol. 67, no. 3, pp. 202-213, Aug. 2021.

[27] A. Sengupta and R. Chaurasia, "Secured Convolutional Layer IP Core in Convolutional Neural Network using Facial Biometric," *IEEE Trans.*

Consum. *Electron*., 2022.

[28] A. Sengupta and M. Rathor, “Securing hardware accelerators for CE systems using biometric fingerprinting,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 2020.

[29] F. Koushanfar, S. Fazzari, C. McCants, W. Bryson, P. Song, M. Sale, M. Potkonjak, Can EDA combat the rise of electronic counterfeiting? *Proceedings of the DAC design automation conference, San Francisco, CA* (2012), pp. 133-138.

[30] Plaza SM, Markov IL. Solving the third-shift problem in IC piracy with test-aware logic locking. *IEEE Trans Comput Aided Des Integr Circuits Syst* 2015;34(6): 961–71.

[31] “Single event upsets”, Intel [online]. Available: https://www.intel.com/content/www/us/en/support/programmable/suppo rt-resources/quality/seu.html, Jan. 2022.

[32] A. Sengupta, S. P. Mohanty, F. Pescador and P. Corcoran, “Multi-Phase Obfuscation of Fault Secured DSP Designs With Enhanced Security Feature,” *IEEE Trans*. *Consum*. *Electron*.*,* vol. 64, no. 3, pp. 356-364, Aug. 2018.

[33] P. Qiu, D. Wang, Y. Lyu and G. Qu, "VoltJockey: Breaking SGX by Software-Controlled Voltage-Induced Hardware Faults," 2019 *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, 2019, pp. 1-6.

[34] S. Park, S. Jeon, B. Kim and J. Lee, “Methods for Improving the Reliability of Intelligent Semiconductor,” 2021 *IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia),* 2021, pp. 1-4.

[35] B. Yuce, C. Deshpande, M. Ghodrati, A. Bendre, L. Nazhandali and P. Schaumont, “A Secure Exception Mode for Fault-Attack-Resistant Processing,” *IEEE Trans*. *Dependable Secure Comput*.*,* vol. 16, no. 3, pp. 388-401, 1 May-June 2019.

[36] M. T. Arafin, A. Stanley and P. Sharma, “Hardware-based anti-counterfeiting techniques for safeguarding supply chain integrity,” 2017

IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1-4.

[37] N. Khan, S. Nitzsche, A. G. López and J. Becker, "Utilizing and Extending Trusted Execution Environment in Heterogeneous SoCs for a Pay-Per-Device IP Licensing Scheme," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2548-2563, 2021.

[38] A. Syed and R. M. Lourde, "Hardware Security Threats to DSP Applications in an IoT Network," 2016 *IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, 2016, pp. 62-66.

[39] Martins M, Matos JM, Ribas RP, Reis A, Schlinker G, Rech L, Michelsen J. Open cell library in 15nm freePDK technology. In: Proceedings of the *ISPD*; 2015.
p. 171–8.

[40] Castillo E, Parrilla L, Garcia A, Meyer-Baese U, Botella G, Lloris A. Automated signature insertion in combinational logic patterns for HDL IP core protection. *In: Proceedings of the 4th southern conference on programmable logic. IEEE*; 2008.

[41] D. Ziener and J. Teich, "Power signature watermarking of IP cores for FPGAs," *J. Signal Process. Syst.*, vol. 51, no. 1, pp. 123–136, 2008. doi 10.1007/s11265-007-0136-8.

[42] 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. doi: 10.1109/MCE.2016.2640622.

[43] K. Wu, R. Karri, Algorithm level recomputing—a register transfer level concurrent error detection technique, *Proc. IEEE/ACM Int. Conf. Comput. Aided Des.* Nov 2001, pp. 537–543.

[44] A. Sengupta, R. Sedaghat, Swarm intelligence driven design space exploration of optimal k-cycle transient fault resilient datapath during high-level synthesis based on user power-delay budget, *Elsevier J. Microelectron. Reliab.* 55 (6) (2015) 990–1004 (May 2015).

[45] K. Wu, R. Karri, Fault secure datapath synthesis using hybrid time and

hardware redundancy, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 23 (10) (2004) 1476–1485.

[46] C. Rusu, et al., Multiple events transient induced by nuclear reactions in CMOS logic cells, *13th IEEE International On-Line Testing Symposium* 2007, pp. 137–145.

[47] Natasa Miskov-Zivanov, Multiple transient faults in combinational and sequential circuits: a systematic approach, *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 29 (10) (2010) 1614–1627.

[48] S. Chen, Z. Guo, J. Feng, and J. Zhou, "An improved contact-based high-resolution palmprint image acquisition system," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 9, pp. 6816–6827, Sep. 2020, doi: 10.1109/TIM.2020.2976081.

[49] J. P. Patil, C. Nayak, and M. Jain, "Palmprint recognition using DWT, DCT and PCA techniques," in *Proc. ICCIC*, 2015, pp. 1–5, doi: 10.1109/ICCIC.2015.7435677.

[50] M. Pudzs, R. Fuksis, R. Ruskuls, T. Eglitis, A. Kadikis, and M. Greitans, "FPGA based palmprint and palm vein biometric system," in *Proc. BIOSIG*, 2013, pp. 1–4.

[51] S. Priya and M. Ezhilarasan, "A novel palmprint authentication system using level 3 pore feature," in *Proc. ICCSP*, Chennai, India, 2018, pp. 623–626, doi: 10.1109/ICCSP.2018.8524278.

[52] S. K. Panigrahy, D. Jena, S. B. Korra, and S. K. Jena, "On the privacy protection of biometric traits: Palmprint, face, and signature," in *Proc. Int. Conf. Contemp. Comput. (CCIS)*, vol. 40, 2009, pp. 182–193.

[53] D. P. Gaikwad and S. P. Narote, "Multi-modal biometric system using palm print and palm vein features," in *Proc. IEEE India Conf. (INDICON)*, Mumbai, India, 2013, pp.1–5, doi: 10.1109/INDCON.2013.6726010.

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

[55] Y. Lao and K. K. Parhi, “Obfuscating DSP Circuits via High-Level Transformations,” *IEEE Trans. Very Large Scale Integration Sys*., vol. 23, no. 5, pp. 819–830, May 2015.

[56] CASIA Palmprint Database, NIST, Chinese Academy of Sciences, Accessed: November. 2022. http://biometrics.idealtest.org/dbDetailForUser.do?id=5#/.

[57] Multimedia Laboratory Datasets. Accessed: November. 2022. [Online]. Available: http://mmlab.ie.cuhk.edu.hk/datasets.html

[58] V. K. Alilou. Accessed: November. 2022. Fingerprint Matching: A simple approach MATLAB Central File Exchange. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/44369-