HARDWARE SECURITY AND IP CORE PROTECTION OF CE SYSTEMS

Ph.D. Thesis

By

DIPANJAN ROY

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

OCTOBER, 2018

HARDWARE SECURITY AND IP CORE PROTECTION OF CE SYSTEMS

A THESIS

Submitted in Partial Fulfillment of the requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

by

DIPANJAN ROY

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

OCTOBER, 2018

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **HARDWARE SECURITY AND IP CORE PROTECTION OF CE SYSTEMS** in the partial fulfillment of the requirements for the award of the degree of **DOCTOR OF PHILOSOPHY** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from December 2015 to October 2018 under the supervision of Dr. Anirban Sengupta, Associate Professor, Indian Institute of Technology Indore.

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

Signature of the Student with date
(DIPANJAN ROY)

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

Signature of Thesis Supervisor with date
(Dr. ANIRBAN SENGUPTA)

DIPANJAN ROY has successfully given his Ph.D. Oral Examination held on

Signature of Chairperson (OEB)
Date:

Signature of External Examiner
Date:

Signature of Thesis Supervisor
Date:

Signature of PSPC Member #1
Date:

Signature of PSPC Member #2
Date:

Signature of Convener, DPGC
Date:

Signature of Head of Discipline
Date:

Acknowledgements

I express my sincere gratitude to **Dr. Anirban Sengupta**, my Ph.D. Supervisor in Computer Science and Engineering, at Indian Institute of Technology Indore, India, for his stimulating guidance, continuous encouragement and supervision throughout my research work. Without his support, this challenge would not have been completed.

I am extremely thankful to **Dr. Abhishek Srivastava** and **Dr. Manavendra Mahato**, PSPC members, at Indian Institute of Technology Indore, India, for their feedback on the progress on my research throughout these years.

I am extremely thankful to Head, Discipline of Computer Science and Engineering for the support.

I also wish to extend my thanks to **Prof. Pradeep Mathur**, Director, IIT Indore, for providing infrastructural facilities to work in.

Special thanks are due to my lab colleague, **Deepak** Kachave for his moral support during this work.

I would like to thank Indian Institute of Technology Indore, for giving me this wonderful opportunity to pursue Ph.D.

I cannot end this list without thanking my sister **Shravani** and my **parents** without whose honest opinions, feedback, complaints and suggestions this thesis would not have been completed.

Dipanjan Roy

Dedicated To My Parents

Abstract

In the modern era of Consumer Electronics (CE), use of Intellectual Property (IP) cores in global supply chains have become an inexorable part of complex System on Chip (SoC) design process. Use of IP cores not only speed up the design productivity massively but also decrease the design period immensely. Importing these IP cores from third-party IP (3PIP) vendors by the system integrator or SoC designer has become a standard industry de facto. Earlier, IP vendors have mainly prioritized on the performance and functionality of an IP core but neglected IP security. As evident, in a typical IP design flow, specification only includes performance and functionality but totally overlooks an important dimension i.e., security/protection. With the increasing popularity and prevailing usage of IP cores in SoC, rivalry between multiple IP core vendors have also multiplied, which invites security threats like infringement of IP core, IP piracy, dishonest claim of ownership, insertion of hardware Trojan, etc. Similar to an untrustworthy 3PIP vendor, an adversary may also be present in a foundry. He/she can launch reverse engineering attack to extract the gate-level netlist of an IP core. The objective is to counterfeit the IP core, insert Trojan logic, etc. Sometimes an attack can also be launched on the secured IP core by removing its security features. Therefore this mandates robust security and protection mechanisms.

Strong security and protection of IP core are expected to become the key focus in recent years due to the involvement of globalization in the SoC design process. This dissertation proposes different solutions to secure and protect hardware/IP cores of CE systems from multiple attacks. To be specific, this thesis provides seven different methodologies for generating secured IP cores for CE systems at the architectural level. It solves the problem of **(a)** fraudulent claim of ownership, IP piracy and IP cloning for DSP IP core by proposing a novel watermarking methodology which is easily adaptable by any (Computer Aided Design) CAD tool. The experimental results over the standard applications indicate an average reduction in the final embedding cost of 6% compared to a recent approach. **(b)** Tampering or removal of the implanted signature in the IP cores by inserting a robust and distributed triple-phase watermarking methodology which is encoded through 7 variables and dispersed over three different phases. The experimental results over the standard applications indicate it yields zero delay and minimal area overhead compared to the baseline and achieves average cost reductions of 7.38% and 6.25% compared to two similar approaches. Further, it achieves 3.4×10^{43} and 2.8×10^{19} times more tamper tolerant than similar approaches. **(c)** Abusing the rights of the IP core buyer and seller for DSP kernel applications by implanting IP seller's watermark and IP buyer's fingerprint in the design, thus provides symmetrical IP core protection. Experimental results indicate an average 1% design cost overhead compared to baseline (unprotected) design and $<$1% design cost overhead compared to a non-symmetrical approach. **(d)** Reverse engineering attack for DSP kernel by transforming the architecture of an application into a non-obvious one through a robust multi-stage structural obfuscation methodology. Experimental results over the standard applications indicate an improvement in the power of obfuscation by 22% and reduction in obfuscated design cost by 55% is achieved compared to recent prior work. **(e)** SAT (Boolean Satisfiability) and removal attacks of the functionally obfuscated DSP IP core by integrating a custom light-weight Advance Encryption Standard (AES) core in the design. The AES module utilizes $<$1% of the available design logic elements of the FPGA. **(f)** Reverse engineering attack on the JPEG (Joint Photography Expert Group) IP core by generating a low-cost structurally obfuscated design. The design cost of the proposed obfuscated JPEG CODEC IP core is reduced by 5% and enhanced the robustness by 76% compared to a non-obfuscated design. **(g)** Hardware Trojan detection by generating a low-cost Trojan secured datapath architecture for DSP kernel, which explores the optimal vendor allocation and loop unrolling factor through Particle Swarm Optimization (PSO) based Design Space Exploration (DSE). The results generated indicate that a detection rate of 100% was achieved while handling such Trojans. Therefore, all approaches incur very minimal design overhead compared to the baseline (unprotected) design and achieve a significant reduction in design overhead and improvement in security/robustness compared to the current state-of-the-art.

List of Publications

SCI Journals (12)

P1. A. Sengupta, D. Roy, S. P Mohanty, ”Triple-Phase Watermarking for Reusable IP Core Protection during Architecture Synthesis”, **IEEE Transactions on Computer Aided Design of Integrated Circuits & Systems**, vol. 37 (4), Apr 2018, pp. 742 755, Impact factor $\sim$ 2.

P2. A. Sengupta, D. Roy, S. P Mohanty, P. Corcoran, ”DSP Design Protection in CE through Algorithmic Transformation Based Structural Obfuscation”, **IEEE Transactions on Consumer Electronics**, vol. 63 (4), Nov 2017, pp. 467 476, Impact factor $\sim$ 1.7.

P3. A. Sengupta, D. Roy, S. P Mohanty, P. Corcoran, ”Low-Cost Obfuscated JPEG CODEC IP Core for Secure CE Hardware”, **IEEE Transactions on Consumer Electronics**, Accepted, Aug 2018, Impact factor $\sim$ 1.7.

P4. A. Sengupta, D. Kachave, D. Roy, ”Low Cost Functional Obfuscation of Reusable IP Cores used in CE Hardware through Robust Locking, **IEEE Transactions on Computer Aided Design of Integrated Circuits & Systems**, Accepted, 2018, Impact factor $\sim$ 2.

P5. D. Roy, A. Sengupta, ”Obfuscated JPEG Image Decompression IP Core for Protecting Against Reverse Engineering”, **IEEE Consumer Electronics**, vol.7 (3), May 2018, pp. 104 - 109 , Impact factor $\sim$ 1.15.

P6. D. Roy, P. Sarkar, A. Sengupta, MK Naskar, ”Optimizing DSP IP Cores using Design Transformation”, **IEEE Consumer Electronics**, vol.7 (4), May 2018, pp. 104- 109 , Impact factor $\sim$ 1.15.

P7. D. Roy, A. Sengupta, ”Low-overhead multi-level watermark for Protecting DSP Core, **IEEE Consumer Electronics**, Accepted, Impact factor $\sim$ 1.15.

P8. D. Roy, A. Sengupta, ”Low Overhead Symmetrical Protection of Reusable IP Core using Robust Fingerprinting and Watermarking during High Level Synthesis, **Elsevier Journal on Future Generation Computer Systems**, vol. 71, Jun 2017, pp. 89101, Impact factor $\sim$ 4.8.

P9. A. Sengupta, D. Roy, ”Protecting an Intellectual Property Core during Architectural Synthesis using High-Level Transformation Based Obfuscation **IET Electronics Letters**, vol. 53 (13), Jun 2017, pp. 849 - 851, Impact factor $\sim$ 1.23.

P10. A. Sengupta, D. Roy, ”Anti-Piracy aware IP Chipset Design for CE Devices: Robust Watermarking Approach, **IEEE Consumer Electronics**, vol. 6(2), Apr 2017, pp. 118 - 124 Impact factor $\sim$ 1.15.

P11. A. Sengupta, D. Roy, ”Automated Low Cost Scheduling Driven Watermarking Methodology for Modern CAD High-Level Synthesis Tools, **Elsevier Journal of Advances in Engineering Software**, vol. 110, Aug 2017, pp. 26-33, Impact factor $\sim$ 3.0.

P12. A. Sengupta, D. Roy, S. Bhadauria, ”Low cost optimized Trojan secured schedule at behavioral level for single & Nested loop control data flow graphs”, **Elsevier Journal on VLSI Integration**, vol. 58, Jun 2017, pp. 378-389, Impact factor $\sim$ 1.0.

Invited Book Chapter (1)

P13. A. Sengupta, D. Roy, Low Cost Dual-Phase Watermark for Protecting CE Devices in IoT Framework, **Springer Book: Security and Fault Tolerance in Internet of Things**, 2018.

Peer- Reviewed Conferences (3)

P14. D. Roy, A. Sengupta, Reusable Intellectual Property Core Protection for Both Buyer and Seller, **Proc. 36th IEEE Int. Conf. on Consumer Electronics**, Las Vegas, Jan 2018, pp. 1-3.

P15. D. Roy, A. Sengupta, Low overhead symmetrical protection of reusable IP core using robust fingerprinting and watermarking during high level synthesis, **Cyber Security Awareness Applied Research, IIT Kanpur- New York University**, Accepted, Nov 2017.

P16. A. Sengupta, D. Roy, Multi-Phase Watermark for IP Core Protection, **Proc. 36th IEEE Int. Conf. on Consumer Electronics**, Las Vegas, Jan 2018, pp. 1-3.

Table of Contents

List of Figures

List of Tables

Nomenclature

X_i	Resource constraint
R_D	D^{th} Resource type
$N(R_D)$	Number of resource type R_D
U/UF	Unrolling factor
μ_m	Mobility of an operation
D_w	Watermarked IP core design
A_T & L_T	Total area and execution time of symmetrically protected design
A_{cons} & L_{cons}	User provided area and execution time constraints
A_T^{OBF} & L_T^{OBF}	Total area and execution time of obfuscated design
$A(R_i)$	Area of resource type R_i
$A(mux)$	Area of mux R_i
$N(mux$	Number of mux used in the design
$A(buffers)$	Area of storage units
$N(buffers)$	Number of storage units used in the design
T_{body}^{OBF}	Single iteration execution delay of the loop body of obfuscated CDFG
T_{first}^{OBF}	delay to execute the first iteration of obfuscated CDFG
I	total loop iteration (loop count)
A_{max}^{OBF} & L_{max}^{OBF}	Maximum possible obfuscated area and latency in the design space
ϕ_1/w_1 & ϕ_2/w_2	User specified area and latency weightage factor
$C_f(X_i)$	Design cost of the IP core with resource configuration X_i
I_{max}	maximum number of pre-defined iteration count
δ	number of iterations to check if no improvement in global best is found
r_1 & r_2	random number
b_1 & b_2	social and cognitive factor
ω	Inertia weightt
$V_{d_i}^+$ & V_{d_i}	New and current velocity of i^{th} particle in d^{th} dimension
$R_{d_{lb_i}}$	local best position of i^{th} particle in d^{th} dimension
$R_{d_{gb}}$	Global best position of d^{th} dimension
$R_{d_i}^+$ & R_{d_i}	New and current resource value/UF of i^{th} particle in d^{th} dimension
p	Population size in PSO
P^{obf} P_i^{obf}	Total Power of Obfuscation and after i^{th} stage transformation
n_i	unique number of modified nodes due to i^{th} stage transformation
n_T	total number of nodes before applying any transformation method

$N(HLT)$	Total number of HLT applied
A_{max} & L_{max}	Maximum possible area and latency in the design space
W/w	Size of watermarking constraints
F	Size of fingerprint constraints
P_c	probability of coincidence
C	Number of colors used in colored interval graph before watermark
$L_T^{proposed}$	Total execution latency of the proposed approach
T^{A1} & T^{M1}	Total execution latency of resource $A1$ and $M1$
T_t	Tamper tolerance ability
V	Number of variables used in encoding
A_{max} & L_{max}	Maximum possible area and latency in the design space
A_T^{JPEG} & L_T^{JPEG}	Total area and execution time of obfuscated JPEG design
A_{max}^{JPEG} & L_{max}^{JPEG}	Maximum possible obfuscated JPEG area and latency in the design space
T	2D-DCT coefficient matrix
M	Input image matrix
i & j	i^{th} row, j^{th} column component of the 2D-DCT matrix
X	Transformed matrix after DCT processing
D	Matrix $(d_{11}, d_{12}, ..d_{88})$ transforms the rows of the M block
N	Size of the DCT matrix
Q	Quantization matrixx
X'	2D quantized 8x8 block
X''	decompressed 8x8 block
O	IDCT block of X block
E	Matrix $(e_{11}, e_{12}, ..e_{88})$ transforms the rows of the X" block
$PSNR(A, B)$	Peak Signal to Noise ratio for two images A and B of size M x N
MAX_A	Maximum pixel value in image A
$MSE(A, B)$	Mean square error between image A and B
U_1 & U_n	Unrolling factor value for outer most and inner most nested loop
A_v	Vendor allocation mode
V_1 & V_2	Two distinct vendors
A_T^{DMR} & L_T^{DMR}	Total area and total execution delay of the DMR design
δ	Latency of one CS
C_{body}^{DMR}	Number of CS required to execute the loop body of the DMR CDFG
C_{first}^{DMR}	Number of CS required to execute the first iteration of the DMR CDFG
$A(R_i^{V_j})$	Area of resource type R_i corresponding to vendor V_j
$N(R_i^{V_j})$	Number of resource R_i corresponding to vendor V_j
U^{OG} & U^{DP}	operations of the original unit and the duplicate unit

Acronyms

3PIP	Third-party IP
AES	Advance Encryption Standard
ALAP	As Late As Possible
ARF	Auto Regression Filter
ASAP	As Soon As Possible
ATPG	Automatic Test Pattern Generator
CAD	Computer Aided Design
CDFG	Control and Data Flow Graph
CE	Consumer Electronics
CED	Concurrent Error Detection
CIG	Color Interval Graph
CODEC	Compression and decompression
CS	Control Step
DCT	Discrete Cosine Transformation
DFG	Data Flow Graph
DMR	Dual Modular Redundant
DSE	Design Space Exploration
DSP	Digital Signal Processor
DWT	Discrete Wavelet Transformation
EDA	Electronics Design Automation
ESL	Electronic System-level
EWF	Enhance Write Filter
FDS	Force Direct Scheduling
FFT	Fast Fourier Transformation
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
GDS	Graphic Database System
HDL	Hardware Description Language
HLS	High-level Synthesis
IC	Integrated Circuit
ICT	Information and Communication Technology
IDCT	Inverse Discrete Cosine Transformation

IIR	Infinite Impulse Response
ILB	IP core Locking Block
IP	Intellectual Property
ITS	Intelligent Transportation System
JPEG	Joint Photography Expert Group
LFSR	Linear-Feedback Shift Register
LICM	Loop Invariant Code Motion
LMS	Least Mean Square
LT	Logic Transformation
LUT	Lookup Table
LU	Loop Unrolling
MERO	Multiple Excitation of Rare Occurrences
MPEG	Moving Picture Expert Group
MSE	Means Squared Error
NC	Noise Cancellation
PoO	Power of Obfuscation
PSNR	Peak Signal-to-Noise Ratio
PSO	Particle Swarm Optimization
RE	Reverse Engineering
ROE	Redundant Operation Elimination
RTL	Register Transfer Level
SAT	Boolean Satisfiability
SCPA	Semiconductor Chip Protection Act
SoC	System on Chip
THT	Tree Height Transformation
UF	Unrolling Factor
VLSI	Very Large Scale Integration

Chapter 1

Introduction

The colossal influence of modern CE on the daily lives of citizens around the world continues to expand. From smartphones to smart TV, home gateways and routers, set-top box to smart kitchen appliances followed by multi-spectral cameras to smart-speakers is taking the world by storm [60, 27]. Continuous innovations in CE industries accompanied by Information and Communication Technology (ICT) make smart city, smart healthcare, Intelligent Transportation System (ITS) a reality thereby making human lifestyle smarter, more sophisticated and fast-paced.

These CE systems are designed using SoC platforms which integrate various system modules such as memory (SRAM, Flash), A-to-D converter, custom processor or co-processor, application specific processors (such as DSP), A/V codecs, wireless modems, etc. (shown in Fig. 1.1). Among these modules, DSP kernels in the form of reusable IP core is at the heart of every SoC based CE devices. (*NOTE: All the abbreviations and their corresponding full form are listed in 'Acronyms' section.*)

1.1 IP Cores: Heart of the CE Systems

A reusable IP core (also known as soft IP core) is a piece of reusable logic block/cell. CE systems are heavily reliant on several DSP IP cores for data-intensive or control-intensive functions. These IP cores are used as a part of SoC design to achieve data/power intensive computation at high speed, minimal silicon area and low power [8]. For example, a smart-

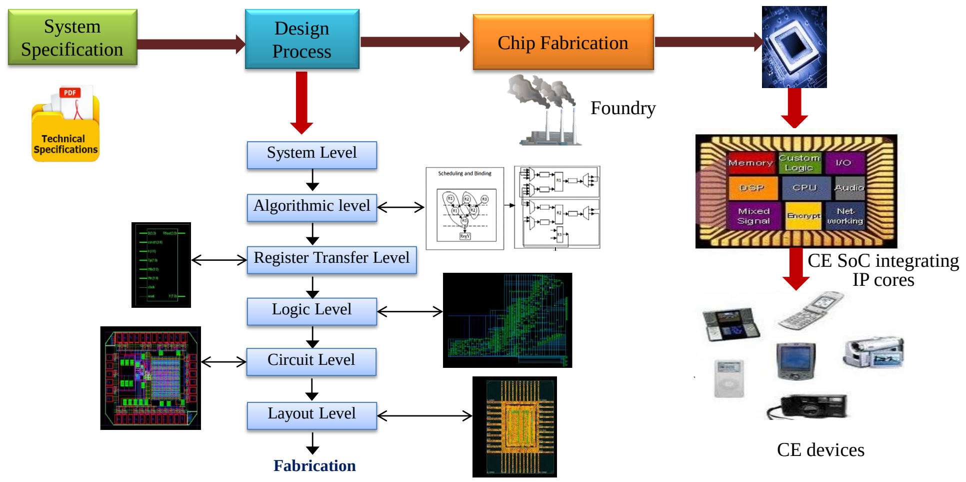


Figure 1.1: Design process of SoC based CE systems

phone always has a dedicated processor for handling calculation of image compression and decompression (CODEC). The job of this application specific processor is to perform repetitive mathematical tasks in real time. Therefore, the main processor of the smartphone can be busy with handling tasks like incoming call, SMS, Email, Wi-Fi etc., while the calculations related to image CODEC are delegated to this dedicated processor. It performs the same calculation for a large set of data. Similar to JPEG CODEC IP core, other DSP IP cores such as Discrete Cosine Transformation (DCT), Fast Fourier Transform (FFT), Finite Impulse Response (FIR), Infinite Impulse Response (IIR) Filters etc., also play a critical role in the successful operation of a CE system. Moreover, the motivation for using a reusable IP core is to maximize design productivity while minimizing design time. The current generation SoC designers amalgamate reusable IP cores which are mass-produced, tested and verified by various companies thus the IP supply chain is distributed worldwide.

Importing these IP cores from third-party IP vendors by the system integrator or SoC designer has become common industry practice. Previously, IP vendors primarily focused on IP performance and IP functionality such as hardware optimization, power optimization, energy efficiency, throughput while neglecting/overlooking the security aspects of an IP cores. As evident, it can be observed that in a typical IP design flow, performance and functionality is only included in the specification. However, prevailing usage of IP cores in the SoC design process increases rivalry among multiple IP core vendors which invites

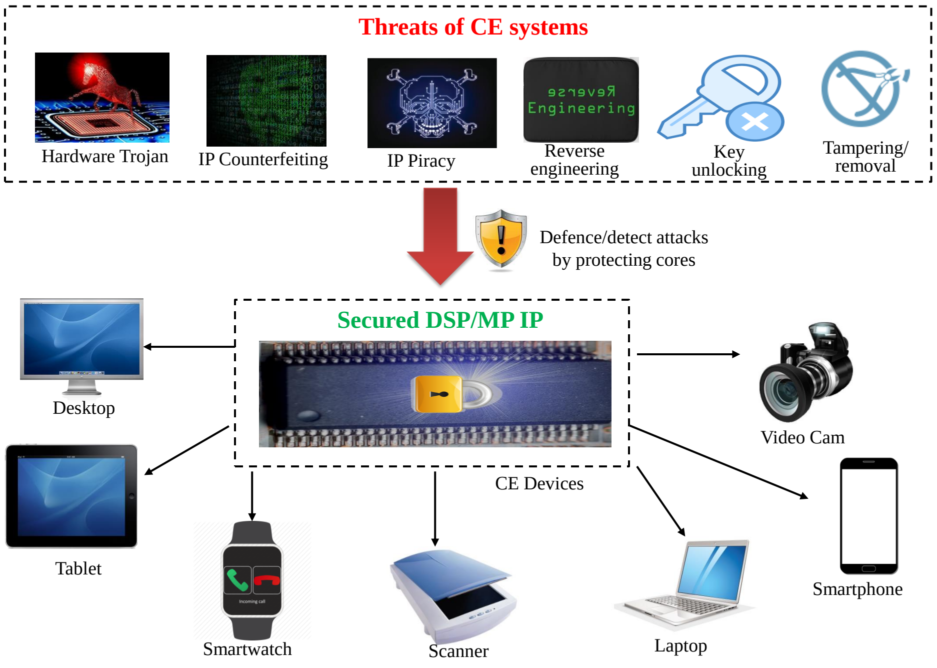


Figure 1.2: Design process of SoC based CE systems

security threats of system breaching, Reverse Engineering (RE), fraudulent claiming of IP core ownership/rights, IP piracy/counterfeiting, system trustworthiness, malfunctioning of hardware etc. (shown in Fig. 1.2). Consequently, looking from the security perspectives, an IP core has plenty of access points for possible attacks for an adversary. Hence, for a trusted and secured SoC design, defense of IP cores is of ultimate significance [49, 54].

1.2 VLSI Design Abstraction Levels

Now the question is among all the levels of the design abstraction of an IP core, what is the best possible level of providing security and protection to a DSP IP core. Before explaining that, let's discuss the abstraction levels in IP core design.

The Very Large Scale Integration (VLSI) consists of multiple design and test levels to satisfy the required design specification. A design engineer accepts the user requirements as constraints and translates them into specifications. The designing is performed once the specifications have been determined. This process includes system level, high level, gate or logic level, transistor or circuit level and physical and layout level. Each of this level is described below (shown in Fig. 1.1) [36].

1. *System level:* A system is represented as a combination of processes, tasks, hardware and software in this highest level of abstraction. It handles the overall execution of the system and the flow of information within the system.

2. *Architectural or Behavioral or Algorithmic level:* The computations of each individual processor executed in a system is controlled at this level. The mapping sequences between the inputs and the outputs of the processors are monitored here.

3. *Register Transfer level:* A system is specified as a combination of storage elements, functional units and interconnecting units in this level. As this level represent each element as a module, it is also known as the module level.

4. *Logic or Gate level:* A system is represented as a network of logic gates and flip-flops in this level. This network of logic gates and flip-flops is known as gate-level netlist or netlist of gates. In this level, the behavior of the system is specified in terms of logic equations.

5. *Circuit or Transistor level:* A circuit is represented as a netlist of transistors in this level. It mainly focuses on the issues related to the nature and number of transistors used in a system.

6. *Physical or Layout level:* A circuit or a system specification is represented in terms of individual transistors. It generates the Graphic Database System (GDS) file which is a database file format used in fabrication to manufacture the Integrated Circuit (IC).

The design process of an IC/IP proceeds from the higher level to the lower level. The automation process of each abstraction level is known as synthesis. Specifying the design at a higher abstraction level is an effective and beneficial way of handling design complexity.

The DSP IP cores used in CE systems are highly complex in nature as they execute complex algorithms. Therefore, designing such complex DSP cores heavily relies on high-level synthesis (HLS) or architectural synthesis. It works as the backbone of the design framework. The reason being such complex DSP cores comprises of more than 10-100 thousand logic gates thus initiating the design process of such applications from the lower abstraction level is neither easy nor an industry de facto. Further, selecting/controlling the architecture of these IP cores is equally important to generate a low-cost, power/area/latency efficient design. This can be achieved through HLS framework with the integration of DSE module (driven through intelligent evolutionary algorithms). This would not have been possible if these design process were started at lower levels.

The next section gives a brief overview of the higher design abstraction levels i.e., HLS. As the target of this thesis is to propose several security/protection approaches for DSP IP cores therefore design process of these IP cores must start from either architectural level or Register Transfer Level (RTL).

1.3 Background on HLS

HLS, also referred to as C synthesis, Electronic System-level (ESL) synthesis, algorithmic synthesis, or behavioral synthesis is an automated design process that accepts the algorithmic description of an application in the form of Data Flow Graph (DFG) or high-level languages such as ANSI C, C++, SystemC and generates the RTL hardware description language (HDL) that implements the same behavior [12, 42, 35, 65]. The hardware description comprises two units viz. datapath unit and control unit. The datapath unit includes functional units (adder, multiplier etc.), storage unit (register) and interconnecting units (multiplexer, de-multiplexer) while the control unit is responsible for the coordination of the data flow between different modules of the datapath [56].

In the early years, the primary objective of HLS was to give a better control over the optimization of the design architecture [36, 12]. It helped the hardware designers to build an efficient and optimized design with the relatively easy verification process.

HLS analyses the input code of the algorithm, performs constraint-based scheduling, allocation and binding to generate the datapath architecture and the controller architecture of the algorithm where each component is represented as a block/module in the design. This is further synthesized to generate synthesized RTL netlist. Multiple academic tools (GAUT [11], LegUp [7] etc.) as well as commercial tools (Vivado, Cyber-Workbench, Bluespec etc.) are available as example. Each step of HLS is described below [11, 36, 21, 28].

1.3.1 Scheduling

It is the process of mapping the operations of the algorithmic description of the input application onto a set of discrete time steps in such a way that all data dependencies/precedence constraints are satisfied. In other words, scheduling divides the algorithmic behavior (can be represented as a DFG) into multiple control steps where each step executes a part of the input application. The number of execution steps increases with the decrement of available resources and vice versa. Several scheduling algorithms are available in the literature such as, As Soon As Possible (ASAP), As Late As Possible (ALAP), List scheduling [21], Force Direct Scheduling (FDS) [43] etc. Discussion on these algorithms is not relevant considering the scope of the thesis.

1.3.2 Allocation

Allocation is the process of assigning available resources/operators to the corresponding operations of the design. In other words, allocation decides which operation has been executed through what hardware.

1.3.3 Resource binding

Binding is the process of providing interconnection between operators and registers through buses, muxes and demuxes. This interconnection between operators/register can be realized by generating a multiplexing scheme for each unit. Several algorithms are available for performing binding such as clique partitioning, left edge algorithm etc [57]. Discussions on these algorithms is available in the literature.

The protection/security protocols of complex DSP IP cores must be combined during HLS design process. In other words, another dimension in the form of security/protection

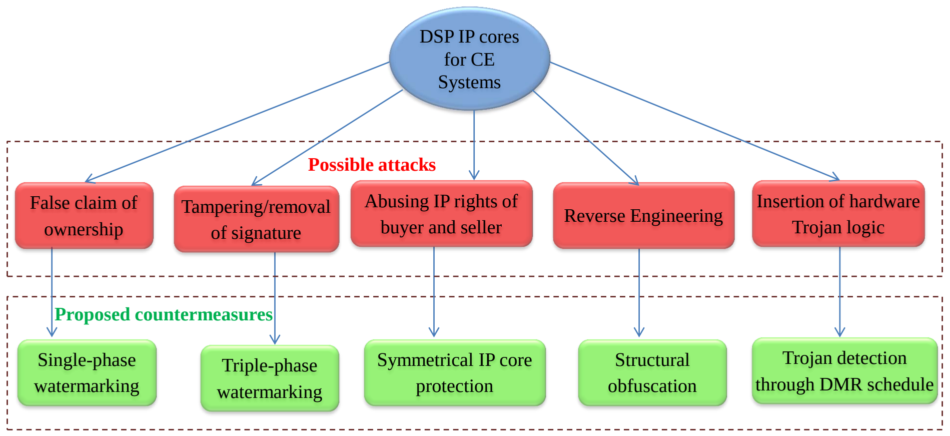

DSP IP cores for CE Systems
Possible attacks
False claim of ownership
Tampering/removal of signature
Abusing IP rights of buyer and seller
Reverse Engineering
Insertion of hardware Trojan logic
Proposed countermeasures
Single-phase watermarking
Triple-phase watermarking
Symmetrical IP core protection
Structural obfuscation
Trojan detection through DMR schedule

Figure 1.3: Possible attacks on DSP IP core based CE systems and their proposed defense mechanisms

must be added to the existing HLS framework. The reasons being (a) the design overhead due to security constraints/features can be handled easily during DSE, (b) the design complexity of implanting security is lesser in HLS, (c) the protection/security algorithm implanted in the higher design level will automatically propagate to the lower design level, (d) the security algorithms merged with the HLS tool can always automatically generate a low-cost protected DSP core.

1.4 Security of IP Cores

The significance of IP core protection at the architectural level is mentioned in the subsequent sections. However, there is no single defense mechanism available to combat all types of attacks. The thesis discusses the details of several attacks for reusable IP cores (more specifically, DSP IP core) used in different CE devices and their possible defense/detection techniques at the architectural level. The overview of typical attacks on DSP IP cores based CE system and their proposed defense mechanism is shown in Fig. 1.3.

1.4.1 Fraudulent claim of ownership

The designing process of an IP core for multi-modal CE designs involves a lot of man-hours of research, investment, validation and effort. However, it can be infringed easily causing a huge financial loss for an original IP owner. Thus IP piracy/forgery and false claim of ownership is a surging security concern. The standard IP protection mechanisms such as copyright, patent, trademark, industrial design rights etc., are not applicable to reusable IP cores designs. In such cases implanting designers signature secretly in the IP core during its design process is the most effective way of defense for an IP vendor. The process of embedding the designer's signature in an IP core design is known as hardware watermarking. Unlike watermarking of the multimedia artifacts (i.e. image, audio, video etc.), IP watermarking has zero tolerance towards the modification of the functionality of the design. This makes IP watermarking more challenging than the watermarking of other artifacts.

1.4.2 Tampering/removal of implanted signature

Embedding signature to protect IP ownership is vulnerable to tampering and removal attack. In this process, an adversary tries to remove the additional design constraints implanted as owner signature completely or partially from the design. The objective is to thwart its effect. Therefore for a rightful owner of an IP core, just implanting his/her signature into the IP core design is not enough. It should also have higher robustness and aforementioned attack-resilient properties.

1.4.3 Abusing rights of IP buyer and seller

In the process of manufacturing an IP core, two entities are involved viz., seller and buyer. An IP seller (also known as IP vendor) is the creator of an IP core whereas an IP buyer (also known as IP user) is the purchaser of an IP core. In an IP core, an IP buyer may claim exclusive buyer privilege so that the same IP copy could not be sold/distributed to his/her rivals in the market. This is possible when an IP buyer obtains customized specifications of an IP core from an IP seller, thus creating a unique one-to-one mapping between both the entities. Therefore, a protected IP core design is needed which facilitates detection of unlawfully redistributed/resold duplicates of an IP core by a deceitful IP seller as well as protect the design from infringements such as piracy and false claim of ownership.

1.4.4 Reverse engineering of IP core

RE of an IP core is a process of identifying its design, structure and functionality. Using RE one can identify the device technology, extract the gate-level netlist, and infer the IP functionality. Though according to Semiconductor Chip Protection Act of 1984 (SCPA) RE is a legal process for teaching, analysis and evaluation purpose, however, an attacker can illegally use RE process for IP piracy, insertion of malicious Trojan logic, etc.

1.4.5 Unlocking a functionally obfuscated netlist

Functional obfuscation approach is a well-established countermeasure for RE attack for combinational/sequential circuits. However, functional obfuscation for DSP IP core is a new research direction now. In this process, locking blocks are inserted into the design to generate a functionally locked DSP circuit/netlist. However, this approach is vulnerable to SAT attack [58] and removal attack [68]. In the SAT attack, the objective is to identify the correct key value of the locked netlist and in the removal attack, the objective is to remove all the locking blocks to generate an unlocked circuit.

1.4.6 Insertion of malicious hardware logic

For reducing time to market of CE systems, major IP cores such as from digital signal procession and multimedia are imported from third-party IP (3PIP) vendors. As these 3PIP vendors are untrustworthy, there is a high possibility of containing malicious logic called ‘Hardware Trojan’ in their IP cores. These malicious logics causes malfunctioning/failure of the complete CE system.

1.5 Organization of the Thesis

Rest of the thesis is organized as follows: **chapter 2** discusses the current state-of-the-art on DSP IP core defense/detection mechanism used in CE systems, followed by the objective and summary of contributions of the thesis. **Chapter 3** proposes an approach to solve the problem of the fraudulent claim of ownership, IP piracy and IP cloning for DSP IP core used in CE systems. **Chapter 4** enhances the defense mechanism by solving the problem of tampering or removal of the implanted signature. **Chapter 5** proposes an approach to solve the problem of abusing the rights of the IP core buyer and seller. The solution of reverse engineering attack for DSP kernel is proposed in **chapter 6**. Further, **chapter 7** enhances the defense mechanism by solving the problem of nullifying the protection through SAT and removal attacks of the functionally obfuscated DSP IP core. **Chapter 8** discusses the solution to the problem of reverse engineering attack on JPEG IP core. **Chapter 9** proposes a solution to the problem of designing a low-cost Trojan secured datapath architecture for IP cores. The results of all the proposed approaches are tested and verified through various well known benchmarks and implemented in CAD tool. The design cost analysis, security analysis and comparative study (with baseline and related approaches) of all the proposed approaches are reported in **chapter 10**. Finally, **chapter 11** concludes the research work presented in the thesis and provides the future scope of the work.

Chapter 2

State-of-the-Art

This chapter presents the state-of-the-art of various IP core attacks and their corresponding defense/detection mechanisms. Thereafter, objective and summary of contributions of the thesis are presented. (*NOTE: All the abbreviations and their corresponding full form are listed in 'Acronyms' section.*)

2.1 State-of-the-Art on IP Core Attacks

The standard IP protection mechanisms such as copyright, patent, trademark, industrial design rights etc., are not applicable for reusable IP cores designs [50]. In this context, IP watermarking is one of the most effective ways to countermeasure IP piracy, IP infringements and false claim of ownership. Most of the IP watermarking approaches that are available in the literature implant additional constraints during lower design abstraction level. For example, authors in [71] have proposed a power watermark technique for the FPGA cores. In this approach, the signature is detected at the power supply pins of the FPGA. With this approach, it is possible to read the watermark only for a given device without any other information from the vendor of the product. In [40] a hierarchical watermarking method for FPGA, IP protection is proposed. The authors implanted the watermark into the netlist and bitstream of the design. In this way, the implanted watermark propagates entirely through the design flow. Authors in [72] have proposed a power watermarking method that detects the signature (watermark) at the power supply pins of the FPGA. They integrated the signature into functional parts of the watermarked core and detected it from a voltage trace with high reliability.

As mentioned earlier, these approaches are not useful for protecting IP core digital signal processing and multimedia application as they embed the watermark in lower design steps. In that context, the advantages of implanting watermark at the higher abstraction level are mentioned earlier. Nevertheless, very limited literature exists for IP core watermarking where the designer's secret signature is implanted at the higher design abstraction level [30, 50, 33]. In [30], initially, the owner's secret signature has been transformed into its equivalent binary form where each digit (0 or 1) indicates a watermarking constraint. After that, each constraint is added as an additional edge in the equivalent colored interval graph ([22]) which represents the sharing of registers of the IP core design. Unlike dual variable (0 and 1) signature in [30], authors in [50] have proposed a multi-variable (combination of four) encoded watermarking methodology to enhance the protection of a reusable IP core. In this approach, each variable which carries a unique encoding rule adds an edge between two node pair in the colored interval graph. Therefore, both [30] and [50] implant watermarking constraints during a single phase of HLS. In [33], an in-synthesis IP watermarking technique is presented. The approach uses marking based on mathematical relationships between numeric values as inputs and outputs at specified times. Both in [30] and [50], the watermarking constraints are implanted during a single phase i.e., register allocation phase of HLS. Hence it could be more vulnerable to signature tampering. Additionally, area overhead due to insertion of watermarking constraints at the register allocation phase could incur high area overhead. Further, [33] is an in-synthesis process, which increases the watermark embedding time. Therefore, all these approaches incur high design overhead which is a major bottleneck for designs working under area/delay constraints and cost budget.

As discussed earlier, securing the rights of the IP buyers as well as the IP seller is one of the important goals of IP core protection. There is only one approach in the literature that has protected the rights of both the entities [31]. In [31], a hidden encrypted mark has embedded into the physical layout of a digital circuit when it is placed and routed onto the FPGA. This mark not only uniquely identifies the source of the circuit but also detect the original recipient of the circuit. Further, this mark is extremely robust and tamper resistant. However, this approach provides protection for both the entities in the lower design abstraction level, i.e. layout level, which is impractical for complex DSP IP cores. Therefore, IP cores such as DSP kernels are not addressed. Moreover, no design optimization algorithm has been used to minimize the design overhead due to the insertion of secret marks.

RE attack is another way to launch several attacks: IP piracy, insertion of malicious Trojan logic etc. Using RE, an adversary can identify the device technology, extract the gate-level netlist, and infer the IP functionality. Hardware obfuscation is the process of counter-measuring the RE attack. It enhances the complexity of RE attack for an adversary. Hardware obfuscation can be classified into three parts, (a) by transforming the HDL code, known as 'source code-based obfuscation' (b) by locking the functionality, known as 'functional obfuscation' (c) by obscuring the structure of a design, known as 'structural obfuscation'. To thwart an attacker from intercepting the Hardware Description Language (HDL) code, it either alters [5] the source code into a compound version or encrypts it [2]. In [5], HDL code obfuscation is achieved using diverse alteration approaches which are challenging for reverse engineering. In [2], the source code is encrypted using the Dead Code Addition approach on a Crypter.

Functional obfuscation locks the design by implanting additional components known as key components into it. These components are controlled by key bits. Applying the correct key will only produce the functionally correct output of the design. There are multiple functional obfuscation approaches available in the literature for the sequential and combinational circuits. For these circuits, the additional components could be dummy states [9], Code-Word [16], XOR/XNOR gates [48, 44, 47], AND/OR gates [19], Lookup Table (LUT) [3], muxes [63, 69] or combination of these elements [34]. However, all these approaches are not applicable for complex DSP IP cores. There is only one approach [25] in the literature which has provided functional obfuscation of DSP IP core. In that approach IP core Locking Block (ILB) based locking technique is used to obfuscate a DSP design where each ILB is inserted in the output of each functional unit. It accepts 8 key bits for each ILB. Further, as the insertion of additional component incurs design overhead, PSO based DSE is performed to obtain the low-cost functionally obfuscated DSP IP core. However, several attack strategies have been available against functional obfuscation techniques, such as key-sensitization [47], SAT [58] and removal attack. Though this approach is capable of handling key-sensitization attack, however, failed to defend against rest of the attacks.

Structural obfuscation changes the standard architecture of an IP core into a non-obvious one while preserving its functionality. This can be achieved through design transformations, arbitrary assignment of logic components, inserting dummy wires or irregular routing [62]. It does not insert any additional component as key-gate into the design, thus does not incur design overhead. There is only one approach in the literature which has provided structural obfuscation of DSP IP core [32]. In that approach, folding based high-level transformation technique is used to obfuscate a DSP design. This transformation technique is capable of generating various meaningful modes thus increasing the complexity to identify the correct functionality within different modes. Further, different equivalent circuits can be generated based on different folding factor which enhances the ambiguity for an attacker. Additionally, in folding based transformation, circuits with different functionality may have the same structure and vice versa. This single-stage structural obfuscation is not secured enough against RE attacks. Moreover, the folding factor is a crucial design parameter to control the final design cost and hence the intelligent selection of proper folding factor may lead to an optimized DSP circuit design. However, no optimal folding factor evaluation process has been proposed in this approach which leads to higher design overhead due to random/uncalculating selection of folding factor.

Though RE attack is very common for CE systems, the design process of standard applications such as JPEG CODEC IP core does not aim at providing security against RE. The current approaches are mainly focused either on the optimization of the software design [1] or on the improvement of the CODEC technique [67]. However, there is no process that designs a dedicated JPEG CODEC IP core secured against RE attacks. To design a secured CE system, protection of JPEG CODEC is crucial as it is a very commonly used IP core in almost every CE systems.

Hardware Trojan is another type of attack that can be launched in any CE systems to introduce malfunctioning of the system. There are multiple classes of hardware Trojans [26, 59]. The detection of Trojans that change the functionality/output of DSP IP core is targeted here.

Different types of Trojan detection mechanisms are available in the literature. Trojans may be inserted in a foundry, therefore different types of detection methods are exist (a) side channel analysis method [38, 24], (b) logic testing method [18, 10]. Moreover, Trojans can also be inserted in an IP core, 3PIP or in a design for which detection methods exists [6, 70, 46, 52]]. However, no effort has been made on generating an optimized Trojan security aware schedule for the single loop and nested loops DSP kernels (based on user constraints) capable of providing runtime detection. Authors in [46, 13] duplicate the IPs in order to detect the presence of a Trojan, however, none of the aforesaid approaches design single and nested DSP IP core for Trojan detection.

Multiple Excitation of Rare Occurrences (MERO) [10] based compact test pattern generation technique also exists for Trojan detection which starts with a golden netlist, random patterns, rare nodes and number of times to activate a node to rare value. However, this technique is sensitive to Trojan sample size and number of times a rare condition is satisfied, in order to create a balance between estimate coverage and simulation time as well as coverage and test vector set size respectively. Without carefully selecting the sensitivity parameters, the results may be imprecise. Further, in the case of a 3PIP in HLS, no golden model exists. Additionally, in [18], a procedure to identify circuit sites where Trojan may be inserted in the untrusted foundry is also proposed, however, it is not beneficial for detection of Trojan inserted in 3PIP module of HLS library for DSP core design.

In [6], a simulation-based Trojan detection approach is proposed where the statistical correlation of Trojan logic compared to the rest of the circuit is weak. A weighted circuit graph is generated based on the input and output of the circuit and their interconnection. The reachability plots identify the weak relation, thus identifying the Trojan. In [70] the Trojan detection process is divided into tracer and checker. The tracer identifies those signals which contain un-activated entries. The checker analyzes these signals to determine the signals which contain redundant inputs and hence are potentially affected by Trojan.

Duplication-based Concurrent Error Detection (CED) technique is proposed in [46] to achieve Trojan secured design during HLS. In duplication-based CED technique, replication of the same design is performed and both the designs are fed with the same set of inputs. The incorrect/dissimilar outputs due to activation of Trojan logic can be identified by comparing their outputs. Additionally, two collusion prevention techniques has also been proposed to avoid activation of hardware Trojan. However, this approach is capable of handling only non-loop based applications in the form of DFGs. It does not handle loop-based control DFG. Besides, this approach does not explore a low-cost hardware configuration and vendor allocation procedure for scheduled DMR system. Consequently, it affects the final area-delay of the design resulting in an inferior quality solution with higher cost.

2.2 Objective of the Thesis

The objective of this thesis is to develop a secured hardware design for DSP IP cores used in CE systems. However, as there is no single defense mechanism available to combat all types of attacks, therefore this thesis proposes possible defense/detection techniques for several attacks at the architectural/algorithmic level for reusable IP cores. In order to realize this, the following objectives have been accomplished:

- Proposes a methodology to solve the problem of the fraudulent claim of ownership, IP piracy and IP cloning for DSP IP core used in CE systems.
- Proposes a methodology to solve the problem of nullifying the defense mechanism through tampering or removal of the implanted signature for DSP IP core.
- Proposes a methodology to solve the problem of abusing the rights of the IP Core buyer and seller for DSP IP core used in CE systems.
- Develop a methodology to address the problem of reverse engineering attack for DSP applications.
- Proposes a methodology to address SAT and removal attacks of the functionally obfuscated DSP IP core.
- Proposes a methodology to solve the problem of reverse engineering attack for an

important multimedia application such as JPEG CODEC commonly used in CE systems.

- Proposes a methodology for designing a low-cost Trojan secured datapath architecture for DSP IP core used in CE systems.

2.3 Summary of Contributions

The focus of this thesis is to provide a number of robust solutions to the aforementioned attacks or security breaches in the field of consumer electronics for DSP IP core.

In order to resolve the issues present in the state-of-the-art approaches, the following contributions have been made through this thesis.

- Solve the problem of the **fraudulent claim of ownership, IP piracy and IP cloning** for DSP IP core (discuss in Chapter 3).

  (Outcome: Refer to Publication P7, P10 and P11).

  The contributions of the approach are as follows:

  1. The proposed methodology is devised in such a way that it can be applied to any DSP core and integrated with any HLS tool.
  2. The watermarks are embedded covertly during the scheduling phase.
  3. The watermark provides robust protection while incurring zero hardware overhead and negligible delay overhead.
  4. Experimental results over the standard applications indicate an average reduction in the final embedding cost with higher security compared to recent approach.

- Solve the problem of nullifying the defense mechanism through **tampering or removal of the implanted signature** for DSP IP core (discuss in Chapter 4).

  (Outcome: Refer to Publication P1, P6, P13 and P16).

  The contributions of the approach are as follows:

1. This is the first work in the area of hardware security that protects the DSP IP cores by embedding watermarking constraints in three different phases of HLS.
2. A combination of 7 unique variables encoded IP owner's signature.
3. Achieves higher robustness and tamper resistance ability compared to prior work.
4. Yields zero delay and minimal area overhead.

- Solve the problem of **abusing the rights of the IP Core buyer and seller** for DSP IP core (discuss in Chapter 5).

  (Outcome: Refer to Publication P8, P14 and P15).

  The contributions of the approach are as follows:

  1. Proposes a symmetrical IP core protection during HLS for the first time that incorporates seller watermark and buyer fingerprint simultaneously.
  2. Proposes a multi-variable encoding scheme for fingerprint and watermark.
  3. Offers higher robustness, lower design overhead/embedding cost, fault tolerance, and faster signature encoding/decoding.
  4. Experimental results over the standard applications indicate an average 1% design cost overhead compared to baseline (unprotected) design and $<1\%$ design cost overhead compared to a non-symmetrical approach.

- Solve the problem of **reverse engineering attack** for DSP applications (discuss in Chapter 6).

  (Outcome: Refer to Publication P2 and P9).

  The contributions of the approach are as follows:

  1. Proposes a multi-stage, compiler-driven high-level transformation based structural obfuscation for DSP applications.
  2. The proposed approach is capable of obfuscating loop-based control DFG.

3. Achieves an improvement in the Power of Obfuscation (PoO) by 22% and reduction in obfuscated design cost by 55% compared to recent prior work.

- Solve the problem of nullifying the defense mechanism through **SAT and removal attacks** of the functionally obfuscated DSP IP core (discuss in Chapter 7).

  (Outcome: Refer to Publication P4).

  The contributions of the approach are as follows:

  1. A custom lightweight AES module is proposed and designed as a custom IP core in a standard CAD tool.
  2. The custom AES module is integrated with a locked circuit of a standard DSP application to mitigate SAT and removal attack.

- Solve the problem of reverse engineering attack for a **low-cost JPEG CODEC** application (discuss in Chapter 8).

  (Outcome: Refer to Publication P3 and P5).

  The contributions of the approach are as follows:

  1. Proposes a structural obfuscation methodology for dedicated JPEG CODEC IP core that aims to enhance the reverse engineering complexity by transforming the architecture.
  2. Optimized the design cost of the obfuscated JPEG CODEC IP core using PSO based DSE by 5% compared to a non-obfuscated design.
  3. Enhancement in security in terms of power of obfuscation is achieved by 76% compared to a non-obfuscated design.

- Solve the problem of designing a **low-cost Trojan secured datapath** architecture for DSP IP core (discuss in Chapter 9).

  (Outcome: Refer to Publication P12).

  The contributions of the approach are as follows:

1. Simultaneously explores an optimal hardware resource, loop unrolling factor and mode of vendor allocation to design a low-cost Trojan security-aware DMR schedule for DSP cores.
2. Proposes a unique particle encoding comprises with candidate hardware resources, candidate loop unrolling factor and candidate vendor allocation mode information.
3. A pre-processing algorithm is devised to discard unfit unrolling factors.

Chapter 3

Single-Phase IP Core Watermarking of CE Systems

The flourishing consumer electronics industry is hugely reliant on CAD-based design automation tools for designing IP core of complex DSP applications. Much as the CAD tools can be used in any abstraction levels of a computing system design, however, to automatically generate a schedule and datapath of a DSP application such as JPEG, FIR, IIR, DWT, DCT etc., use of the CAD HLS tools is mandatory. A CAD HLS tool automatically generates a scheduled graph or datapath of an application as a soft IP core. It has multiple advantages such as re-usability, higher productivity and lower design cycle to state a few. These soft IP cores are the product of huge investment, manpower, research and intellectuality. However, not many HLS CAD tools explicitly consider the protection of IP core as an important aspect while generating it. Thus uncovering risks of ownership and counterfeit threats.

This chapter presents a novel solution through a low-cost watermarking methodology for DSP cores. The proposed methodology is integrated with HLS design process to automate the watermarked DSP IP core generation. This has also been fully automated by integrating with an HLS tool. This is the first work in the area of DSP IP core protection that integrates a watermarking algorithm with a CAD HLS to automate the watermark IP generating process.

The chapter is organized as follows: Section 3.1 discusses the problem formulation of the proposed approach; Section 3.2 explains the proposed watermarking methodology; Section 3.3 explains the proposed signature detection approach; Section 3.4 demonstrates the proposed watermarking and signature detection approach through a standard application and finally Section 3.5 summarizes the chapter. (*NOTE:All the abbreviations and taxonomy are listed in 'Acronyms' and 'Nomenclature' section respectively.*)

3.1 Problem Formulation

3.1.1 Threat model

For an input DSP kernel application, design a watermarked IP core to protect against (i) fraudulent ownership (ii) ownership conflict/abuse and (iii) IP infringement.

3.1.2 Input & Output

Inputs: (a) control data flow graph representing a DSP IP core, (b) user resource constraint $(X_i) = N(R_1); N(R_2); :: N(R_D)$, (c) loop unrolling factor and (d) signature. Output: a low-cost overhead watermarked IP core. Where, $N(R_D)$ indicate the number of resources of type R_D.

3.2 Proposed Methodology

The proposed approach implants a low-cost watermark covertly during scheduling phase of HLS. The watermark is extracted from the encoded multi- variable signature for loop-based CDFGs. The IP core seller's watermark is implanted as hidden additional constraint by employing some local scheduling rules. More explicitly, instead of employing priority resolver functions or random break during operation scheduling conflict, proposed approach employs watermark encoding based scheduling rule to select which operation to assign at which control step. This is a very covert way of inserting watermark, as during regular operation scheduling conflict, the watermarking constraints are inserted with zero hardware

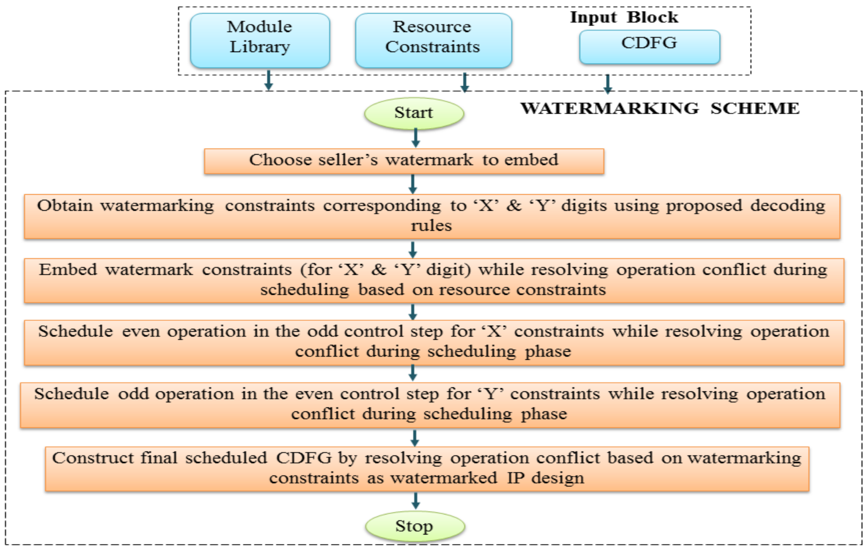

Module Library
Resource Constraints
Input Block
CDFG
Start
WATERMARKING SCHEME
Choose seller's watermark to embed
Obtain watermarking constraints corresponding to 'X' & 'Y' digits using proposed decoding rules
Embed watermark constraints (for 'X' & 'Y' digit) while resolving operation conflict during scheduling based on resource constraints
Schedule even operation in the odd control step for 'X' constraints while resolving operation conflict during scheduling phase
Schedule odd operation in the even control step for 'Y' constraints while resolving operation conflict during scheduling phase
Construct final scheduled CDFG by resolving operation conflict based on watermarking constraints as watermarked IP design
Stop

Figure 3.1: Proposed watermarking methodology

and negligible latency overhead. In other words, watermarking constraints are embedded in scheduling step by forcing specific operations to specific control steps during the process of schedule conflict resolution.

3.2.1 Signature encoding rules

The proposed signature creation for seller watermark consists of two unique encoded variables X and Y where each variable carries an encoded meaning. An IP seller must provide his/her encoded signature as a combination of these two variables. The signature can be consist of any number of 'X' and 'Y' variable. However, it is recommended to provide a signature size with 15 or higher digits to embed a robust watermark. The encoded meaning of each variable is as follows:

- X = Force even operation in odd control step while resolving scheduling conflict in scheduling phase.

- Y = Force odd operation in even control step while resolving scheduling conflict in scheduling phase.

3.2.2 Watermark embedding process

Fig. 3.1 represents the flow diagram of the proposed watermarking methodology. As shown in the figure it accepts module library, applications in the form of CDFG or DFG, user provided hardware constraints and loop unrolling factor. Additionally, a desired signature of the IP seller is selected as a combination of X and Y digits. To implant the signature as watermark each signature digit is converted to its equivalent watermarking constraints using proposed signature encoding rules.

Initially, a loop unrolled CDFG is generated based on the user specified loop unrolling factor. Then scheduling of the operations are performed in each step using the watermarking constraints abiding by the user specified resource constraints. As mentioned earlier, these watermarking constraints are implanted during scheduling by acting as a priority resolver if there is a conflict between operations. In other words, for each scheduling conflict, a watermarking constraint is used to determine which operation gets priority over the other. This process will continue until all the operations of the input CDFG has been scheduled for generating a watermarked IP core. *(Note: in case further watermarking constraints are not available for embedding during additional conflict resolution in scheduling, then conflict breaking is done based on sorted ordering of operations).*

Each step of the proposed watermarking process is mentioned below:

1. Accept module library, CDFG/DFG and user provided resource constraints as input.

2. Create a sorted list of operations of the CDFG/DFG as per their operation number in ascending order.

3. Accept an IP seller signature as combination of X & Y digits.

4. Extract the watermarking constraints from the signature using encoding rules.

5. Based on the resource constraints (in step 1) initiate scheduling of CDFG.

6. In case of operation conflict, use decoded watermarking constraints (in step 4) to resolve as follows:

   (a) For 'X' constraint schedule even numbered operation in the odd numbered control step.

   (b) For 'Y' constraint schedule odd numbered operation in the even numbered control step.

7. Continue step 6 until all the operations have been scheduled.

8. Finally, watermarked IP core is generated where additional constraints are implanted by resolving conflicts.

3.2.3 Properties of proposed watermark

The proposed watermarking methodology satisfies all the following desirable properties:

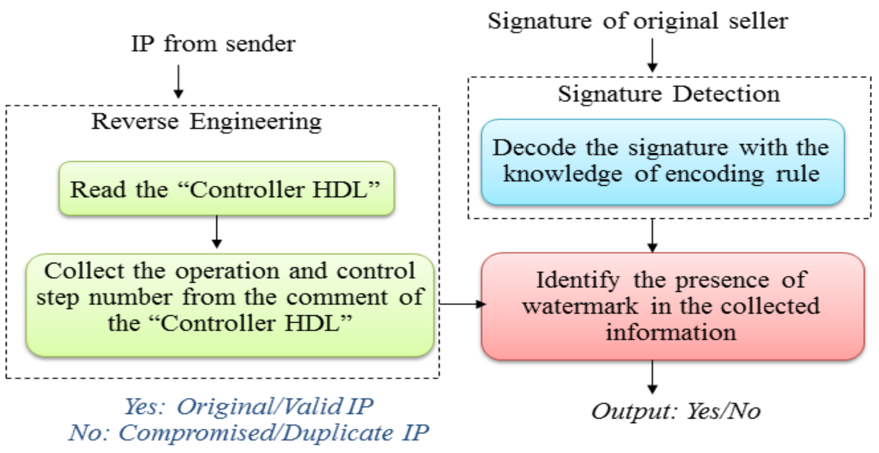
(a) **Low Implanting Cost:** The proposed watermark incurs zero area and minimal latency overhead. Thus incurs very low design cost overhead.

(b) **Strong Resiliency:** The proposed approach implant watermarking constraints as multi-variable encoded signature in the IP core design. Hence incorporating strong resiliency against partial or complete removal of signature.

(c) **Higher Tamper Tolerance Ability:** As the proposed approach distributes the watermarking constraints throughout the IP core design, therefore, the ownership remains preserved even after partial removal or tampering by an attacker.

(d) **Lesser Watermark Implanting & Detection Time:** The proposed watermarking scheme is integrated with the HLS CAD to automate the watermarking process making signature implanting time fast. Moreover, for the original owner with complete knowledge of encoding rules, the decoding process is simple and straightforward. However, for an outsider identifying the presence of watermark is complex and difficult.

(e) **Integration with Current HLS CAD Tools:** Any HLS tool performs the following major steps: scheduling, hardware allocation, register allocation and operation binding to generate a datapath of an application before RTL synthesis. As the proposed



IP from sender
Reverse Engineering
Read the "Controller HDL"
Collect the operation and control step number from the comment of the "Controller HDL"
Signature of original seller
Signature Detection
Decode the signature with the knowledge of encoding rule
Identify the presence of watermark in the collected information
Output: Yes/No
Yes: Original/Valid IP
No: Compromised/Duplicate IP

Figure 3.2: Proposed signature detection process

approach inserts watermark during operation conflict in scheduling phase it can be integrated seamlessly with the scheduling algorithm of the tool.

3.3 Proposed Signature Detection Process

Signature detection process is a must for any signature encoding process as it is crucial to identify the actual owner of the IP in case of any ownership conflict. In the proposed approach, signature detection is a two-step process: (a) Inspection and (b) Verification.

3.3.1 Inspection

The controller design file of an IP core is written using any standard HDL such as VHDL or Verilog. This file of the watermarked IP core is first inspected. Based on the inspection the relevant information such as operation number and its corresponding allocated control step number have to be extracted.

3.3.2 Verification

The watermarking constraints must be decoded from the IP seller signature through proposed encoding rules. The existence of seller signature is verified by matching each additional watermarking constraints with the information extracted during inspection step. The block diagram of proposed signature detection process is shown in Fig. 3.2.

3.4 Motivational Example

This section gives a demonstration of proposed watermarking approach through a standard benchmark and discusses the resolution of threat scenario.

3.4.1 Demonstration of proposed watermarking approach

Let us assume FFT application is taken an demonstration example in the form of an unscheduled CDFG. The user provided resource constraints as: 2 adder, 2 multiplier, 1 subtractor, 1 comparator and UF=2. The schedule length of the application may be restricted due to loop UF, however it results in lower latency due to repetition of loop body. As in the proposed approach UF value is user specified, therefore the proposed watermark scheme is independent of UF value. The unscheduled CDFG of FFT with UF=2 is shown in Fig. 3.3 **(step 1)**. The two consecutive iterations corresponding to UF = 2, are shown in dotted boundary lines.

As shown in the figure, each node number is indicated through a integer value. The resource type of a node is indicated through its corresponding color. For example, green node indicates adder, blue node indicates multiplier, orange node indicate subtractor and red node indicate comparator. The interconnecting arrows as top-down flow represents the data dependencies between nodes. A sorted list of operation of the generated CDFG is created based on the their operation number in ascending order **(step 2)**.

Now based on the combination of 'X' and 'Y', an IP seller's signature is assumed as "YXXYXYYX" **(step 3)**. According to the proposed encoding meaning of 'X' and 'Y', the watermarking constraints have to extracted from the IP seller's signature **(step 4)**. Each signature digit and its corresponding decoded constraint is shown in Table 3.1.

According to the sorted list four addition operations are ready to be scheduled in CS 1. They are operation 1, 2, 15 and 16. However, due to resources constraints provided by the user only 2 adders are available **(step 5)**. Therefore, operation conflict scenario arises

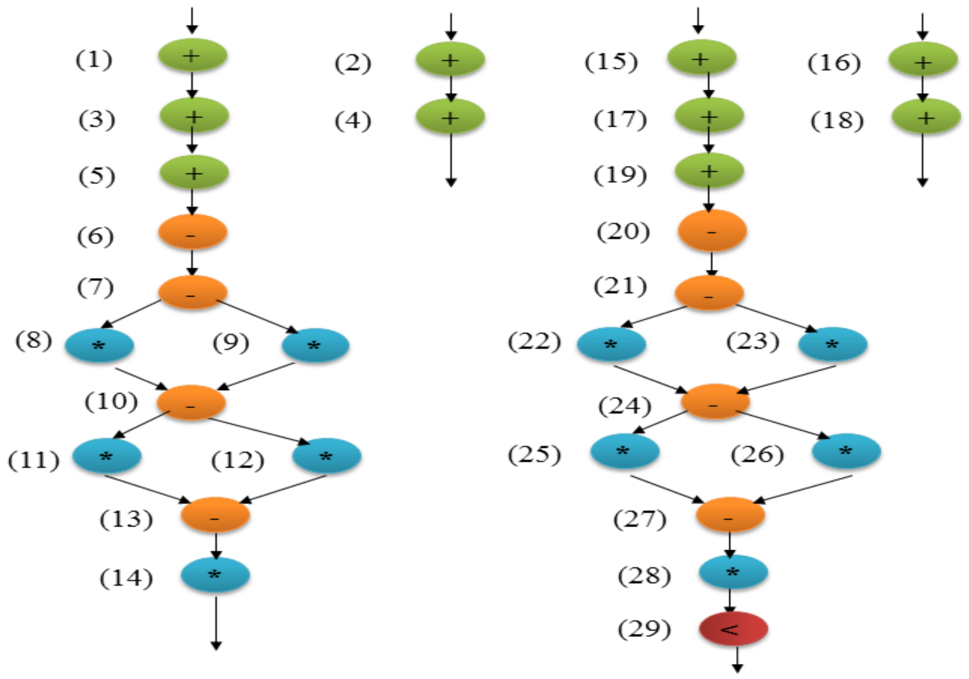


Figure 3.3: Unscheduled CDFG of FFT unrolled twice (UF = 2)

Table 3.1: IP core seller's signature and its decoded constraints

Seller Signature (8 digits)	**Corresponding decoded constraints (assigning of operation in CS)**
Y	Assign operation 1 in CS 2
X	Assign operation 2 in CS 1
X	Assign operation 16 in CS 1
Y	Assign operation 15 in CS 2
X	Assign operation 4 in CS 3
Y	Assign operation 3 in CS 4
Y	Assign operation 17 in CS 4
X	Assign operation 6 in CS 7

during scheduling of CS 1. The conflict is resolved by implanting proposed watermarking scheme. According to the proposed encoding rule the first signature digit i.e. Y forces odd operation i.e. operation 1 into even control step i.e. CS 2. However, operation conflict still exist between operation 2, 15, and 16. The second signature digit i.e. 'X' forces even operation i.e. operation 2 into odd control step i.e. CS 1.; the third signature digit X forces even operation i.e. opn 16 to odd control step i.e CS 1 while the fourth signature digit Y forces odd operation i.e. opn 15 into even control step i.e. CS 2 **(step 6)**. Therefore, according to the first four signature digits i.e. 'YXXY', operation 2 & 16 are scheduled in CS 1 and operation 1 & 15 are scheduled in CS 2. This indicates the watermarking constraints corresponding to the first four signature digits is implanted into the design by resolving the operation conflicts. Similarly, all the signature should be implanted into the design of the unrolled CDFG of FFT **(step 7)**. The complete scheduled CDFG after embedding all the signature digits represents the watermark implanted FFT design **(step 8)**. If there are any pending operation conflict available after implanting all the signature digits, in such case operation conflict should be resolved on sorted operation number (lower operation number will get higher priority to be scheduled).

Fig. 3.4 represents the watermark implanted schedule CDFG of FFT. The application is scheduled based on 2 adder, 2 multiplier, 1 subtractor, 1 comparator and unrolled twice. Each control step is separated through horizontal line known as control step line and the corresponding CS number is mention at the left side of each control step. For example, operation 2 and operation 16 (shown as (2) and (16)) are scheduled in CS 1, demarcated with a control step line, operation 1 and operation 15 (shown as (1) and (15)) are scheduled in CS 1.

Parallelization of two consecutive iterations during scheduling can also be observed in Fig. 3.4. Both the iterations start in CS1, however iteration 1 finishes its execution at CS 13 whereas iteration 2 finishes its execution at CS 15. It must be noted that, due to proposed watermarking scheme the area and delay overhead for FFT is **ZERO**. In the proposed approach the area overhead will **always remain ZERO** for all benchmarks; however, possibility of negligible delay overhead is there for some benchmarks.

3.4.2 Threat scenarios

This subsection discusses several threat scenarios to launch false claim of ownership on watermarked IP core.

An original IP seller can not prove his/her legal ownership for a reusable IP core if no watermark is implanted. Thus suffer from financial huge loss. However, attack on watermarked IP core can still be launched as follows: (a) by inserting attacker's own signature,

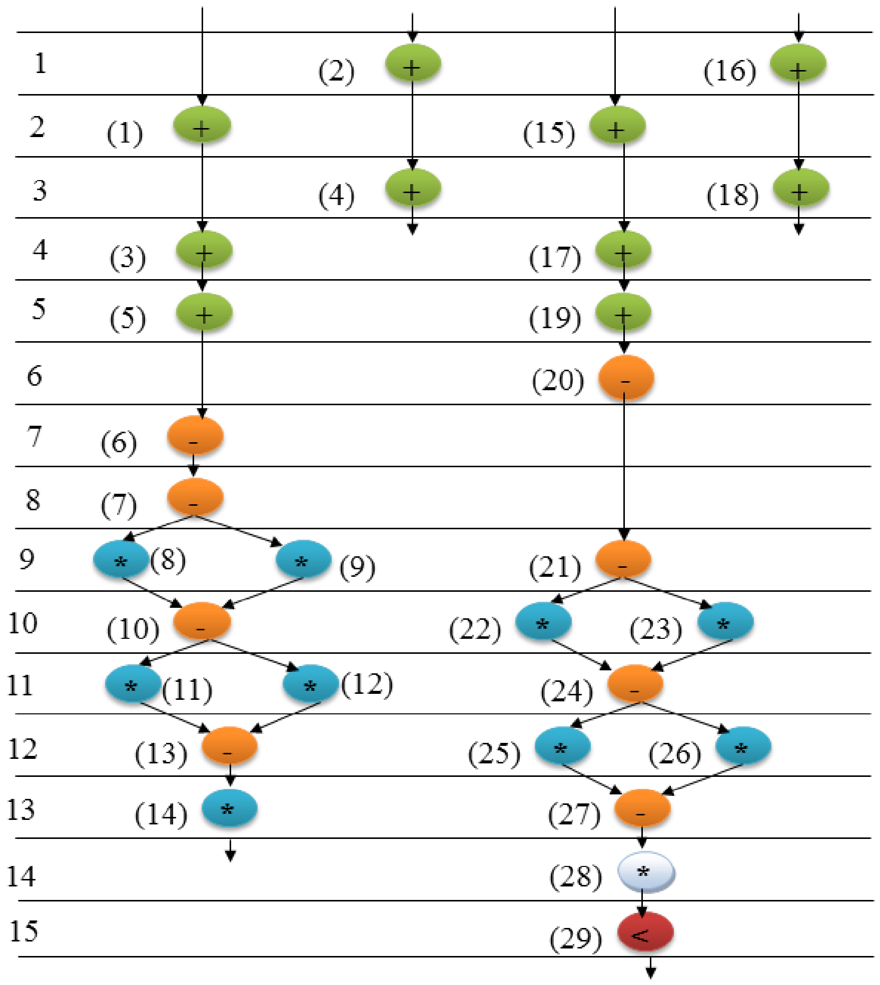


Figure 3.4: Watermark embedded scheduling of FFT CDFG unrolled twice (UF = 2) with two multiplier (blue nodes), two adders (green nodes), one subtractor (orange nodes) and one comparator (red node).

(b) by finding an unintended signature through inverse watermark extraction, (c) by tampering the original signature and then falsely claim for ownership. In such a scenarios, it can be nullified through proposed signature detection approach.

For the first type of attack, when an attacker inserts his/her own signature into the watermarked IP core design, the signature of the original owner still remains intact inside the IP core design. Hence ownership conflict can be resolved easily.

In the second type of attack, when an attacker tries to find an unintended signature in the IP design and claim as his/her watermark, the claimant with a more meaningful and strong signature will be the winner. As the proposed watermarking scheme inserts a strong and meaningful watermark, therefore the ownership resolution will be simple and straightforward.

In the third type of attack, when an attacker tries to tamper or remove the original signature and make it a non-watermarked IP core design. As the proposed watermarking scheme embeds signature throughout the design therefore,complete tampering or removal is extremely challenging. If not all some traces of watermark will always present in the design. Hence ownership conflict can be resolved easily.

Table 3.2: Demonstration of signature detection of watermarked IP core for resolving ownership conflict

Extracted info from the controller HDL through inspection	**Watermarking constraints decoded from the vendor signature**	**Signature verification (comparing extracted info and decoded constraints)**
CS 1: Opn (2), Opn (16)	Y: operation 1 in CS 2	Matches
CS 2: Opn (1), Opn (15)	X: operation 2 in CS 1	Matches
CS 3: Opn (4), Opn (18)	X: operation 16 in CS 1	Matches
CS 4: Opn (3), Opn (17)	Y: operation 15 in CS 2	Matches
CS 5: Opn (5), Opn (19)	X: operation 4 in CS 3	Matches
CS 6: Opn (20)	Y: operation 3 in CS 4	Matches
CS 7: Opn (6)	Y: operation 17 in CS 4	Matches
CS 8: Opn (7)	X: operation 6 in CS 7	Matches
CS 9: Opn (8), Opn (9), Opn (21)		
CS 10: Opn (10), Opn (22), Opn (23)		
CS 11: Opn (11), Opn (12), Opn (24)		
CS 12: Opn (13), Opn (25), Opn (26)		
CS 13: Opn (14), Opn (27)		
CS 14: Opn (28)		
CS 15: Opn (29)		

3.4.3 Resolution of ownership conflict through proposed approach

This subsection discusses how to invalidating false claim of ownership through proposed signature detection method with an example. To mitigate false claim of ownership signature detection of watermarked IP core is performed. To achieve that, each operation and its corresponding control step is extracted from the controller HDL file through inspection of the watermarked IP core. Thereafter, accept the signature of the claiming IP vendor (assuming: 8-digit signature as YXXYXYYX) and decode it based on the proposed encoding rule. Now, match the existence of each watermarking constraints decoded from the vendor's signature with the extracted information of controller HDL file. Table 3.2 represents signature verification using extracted information from the controller HDL file of the watermarked FFT (shown in Fig. 3.4) and the watermarking constraints decoded from the vendor's signature. If all the constraints match, it can be concluded that the IP vendor is the original owner of that IP core. Thus the proposed approach is capable of providing protection and nullifying the threat of false ownership claim.

Note: The corresponding experimental results of the proposed methodology is explained in Chapter 10 Section 10.1.

3.5 Summary

This chapter presents a novel low-cost watermarking algorithm for modern CAD HLS tools. The algorithm is integrated with a modern HLS tool to automate the watermarked DSP IP core generating process. The work proposes a watermarking algorithm during the scheduling phase of HLS to thwart ownership abuse and IP piracy. The embedding watermark satisfies desirable properties such as covertness, robustness, low embedding cost and low signature implanting complexity. This innovation has several unique features: (a) the proposed algorithm is devised in such a way that it can be integrated with any HLS tool (b) watermarks are embedded covertly during the scheduling phase of HLS (c) provides robust protection while incurring zero hardware overhead and negligible delay overhead. Finally, experimental results over the standard applications indicate an average reduction in the final cost of 6% compared to recent approaches. Thus this approach provides an significant advancement over similar IP core protection mechanisms of CE systems.

Chapter 4

Multi-Phase IP Core Watermarking of CE Systems

As discussed in the previous chapter, a watermarked IP core can nullify security threats such as IP core forgery, dishonest claim of ownership, IP piracy etc. Therefore, multiple attacks such as signature tampering, partial/complete signature removal, inserting unauthorized signature can be launched on a watermarked IP core to dwindle its effects. For rightful owner of an IP core, it is not only important to resolve the ownership conflict but also to prove that his/her IP core has been used illegally in a product. This necessitates the need to implant a strong watermark with higher robustness and the aforementioned attack resiliency.

This chapter presents a novel triple-phase watermarking methodology to enhance the robustness of watermarked DSP core for CE systems (shown in Fig. 4.1). It implants IP owner's signature which is encoded through 7 variables in three different but interrelated phases of HLS i.e. scheduling phase, hardware allocation phase and register allocation phase ensuring strong resistance against complete removal or malicious modification of watermark. In case of ownership conflict, the embedded signature can be detected by inspecting the datapath and controller HDL file of the watermarked IP core. This is the first work in the area of IP core protection of CE systems that has implanted watermarking constraints in three different phases.

The chapter is organized as follows: Section 4.1 discusses the problem formulation of the proposed approach; Section 4.2 explains the proposed watermarking methodology; Section 4.3 explains the proposed signature detection approach; Section 4.4 demonstrates the proposed watermarking and signature detection approach through a standard application;finally the summary of this chapter is presented in Section 4.5. (*NOTE: All the abbreviations and taxonomy are listed in 'Acronyms' and 'Nomenclature' section respectively.*)

4.1 Problem Formulation

4.1.1 Threat model

For an input DSP kernel, design a robust watermarked IP core to protect against (i) fraudulent ownership (ii) ownership conflict/abuse and (iii) IP infringement.

4.1.2 Input & Output

Inputs: (i) DSP application, (ii) resource configuration, (iii) module library, (iv) vendor signature. Output: Watermarked IP core design.

4.1.3 Target platform

The proposed watermarking approach is seamlessly adaptable to any EDA tool.

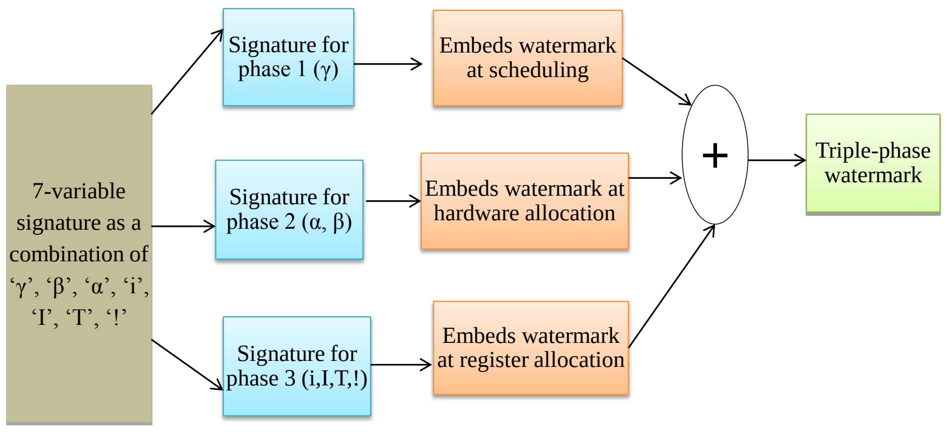

7-variable signature as a combination of 'γ', 'β', 'α', 'i', 'I', 'T', '!'
Signature for phase 1 (γ)
Signature for phase 2 (α, β)
Signature for phase 3 (i,I,T,!)
Embeds watermark at scheduling
Embeds watermark at hardware allocation
Embeds watermark at register allocation
+
Triple-phase watermark

Figure 4.1: Overview of proposed triple-phase watermarking approach

4.2 Proposed Methodology

Fig. 4.2 depicts the overview of the design process of the proposed triple-phase watermarking approach. In this approach, watermark is embedded in three phases of HLS i.e. scheduling, hardware allocation and register allocation. Additionally, the IP owner's signature is encoded through a 7-variable encoding scheme. Thus, making the implanted watermark extremely robust and lowing the possibilities of any malicious signature modification. Therefore, the proposed triple phase watermarking with 7-variable encoding enhances the IP core protection and increases the tamper-tolerance ability. Moreover, it is also extremely difficult for an attacker to identify which HLS phases and how the watermark constraints are embedded in the design.

In this approach, scheduling, resource allocation and register allocation information of an IP core design are presented through three different tables: (a) "Functional unit allocation" table, (b) "Non-critical operations ($\mu_m > 0$)" timing table, and (c) "Register allocation" table (inspired from [50]). As mentioned earlier, the signature of triple-phase watermarking methodology consist of seven different variables where each variable indicates an encoded meaning. These seven variables are 'α', 'β', 'γ', 'i', 'I', 'T' and '!'. Among these variables, 'γ' digits embed the watermarking constraints at scheduling phase by modifying the "non-critical operations ($\mu_m > 0$)" table, 'α' and 'β' digits embed the watermarking constraints at hardware allocation phase by modifying the "functional unit allocation" table and 'i', 'I', 'T' and '!' digits embed the watermarking constraints at register allocation phase by modifying the "register allocation" table through colored interval graph. In a colored interval graph, each node indicates a storage variable. The lifetime of a storage variable lies between the time it is created (value written into it) and the last use (value read from it). An edge between two nodes indicates there is an overlap in lifetime between those storage variables. The color of the node indicates the allocated register of that corresponding storage variable. Therefore, two storage variables whose lifetimes overlap cannot be stored in the same register. In other words, two nodes having a common edge between them cannot have same node color. Thus, adding additional edges in the colored interval graph as watermarking constraints will force to execute two storage variables in two different registers.

4.2.1 Watermark encoding

The encoding rule of all the aforementioned variables are defined below:

- α = *For odd control step:* odd operation will be assigned to hardware of vendor type 1 (U1) and even operation will be assigned to hardware of vendor type 2 (U2).
- β = *For even control step:* odd operation is assigned to hardware of vendor type 2 (U2) and even operation is assigned to hardware of vendor type 1 (U1).
- γ = Move an operation of non-critical path with highest mobility into immediate next CS.
- i = Add an additional edge between (prime, prime) node pair of CIG.
- I = Add an additional edge between (even, even) node pair of CIG.
- T = Add an additional edge between (odd, even) node pair of CIG.
- ! = Add an additional edge between (0, any integer) node pair of CIG.

An IP core owner is free to select any random combination of these seven variables as his desired signature. (NOTE: ”Edge between (prime, prime) node pair” indicates insert an edge between two nodes having prime node number).

4.2.2 Process of implanting watermark

The previous sub-section discusses the signature encoding rule of seven variables while this sub-section discusses the watermark implanting process to generate an watermarked IP core from an input data intensive application (shown in Fig.4.2).

The triple-phase watermark implanting process has four parts, i.e., (a) pre-implanting, (b) 1st phase watermark implanting, (c) 2nd phase watermark implanting and (d) 3rd phase watermark implanting. The detailed steps of each of these parts are as follows:

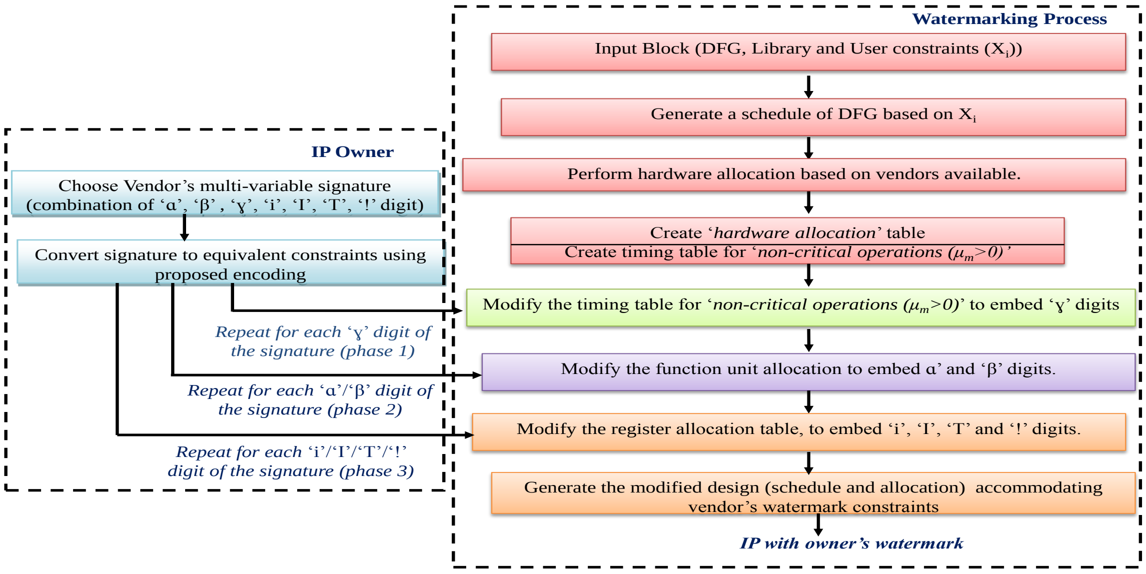

Watermarking Process
Input Block (DFG, Library and User constraints (Xi))
Generate a schedule of DFG based on Xi
Perform hardware allocation based on vendors available.
Create 'hardware allocation' table
Create timing table for 'non-critical operations (μm>0)'
Modify the timing table for 'non-critical operations (μm>0)' to embed 'γ' digits
Modify the function unit allocation to embed α' and 'β' digits.
Modify the register allocation table, to embed 'i', 'I', 'T' and '!' digits.
Generate the modified design (schedule and allocation) accommodating vendor's watermark constraints
IP with owner's watermark
IP Owner
Choose Vendor's multi-variable signature (combination of 'α', 'β', 'γ', 'i', 'I', 'T', '!' digit)
Convert signature to equivalent constraints using proposed encoding
Repeat for each 'γ' digit of the signature (phase 1)
Repeat for each 'α'/'β' digit of the signature (phase 2)
Repeat for each 'i'/'I'/'T'/'!' digit of the signature (phase 3)

Figure 4.2: Design process of proposed triple-phase watermark

A. **Pre-implanting steps:**

(1) Schedule the data intensive application (accepted as an CDFG) based on user resource configuration.

(2) Perform functional unit allocation and storage variable allocation.

(3) Construct the timing table for "non-critical operations ($\mu_m > 0$)", "Functional unit allocation" table, and "Register allocation" table from the scheduled CDFG.

(4) Sort the operations control stepwise based on their number in increasing order.

(5) Accept an n-digit vendor's signature as a combination of 'α', 'β', 'γ', 'i', 'I', 'T' and '!' variables.

B. **1st phase watermark implanting steps:**

(1) For each occurrence of 'γ', shift an operation of the non-critical path into its next control step while checking from control step 1 onward (without repeating), such that:

i. The operation do not have any child operation in its immediate next control step.

ii. Shifting does not violate the hardware constraints.

iii. In case of conflict, the operation with the highest mobility gets priority.

(2) Modify the timing table of "non-critical operations ($\mu_m > 0$)" after embedding all 'γ' variables.

C. **<u>2nd phase watermark implanting steps:</u>**

(1) Accept the 1st phase watermark implanted scheduled design as input to embed 2nd phase watermark.

(2) For each occurrence of 'α' and/or 'β' variable, perform re-allocation of functional units according to the encoding rule of 'α' and 'β'.

(3) Modify the functional unit allocation table after embedding all 'α' and 'β' variables.

D. **<u>3rd phase watermark implanting steps:</u>**

(1) Accept the 1st and 2nd phase watermark implanting design as input to embed 3rd phase watermark.

(2) Construct a colored interval graph using storage variables to identify the minimal registers count.

(3) Generate a list of additional edge to be added in the colored interval graph from the decoded signature by traversing the sorted nodes.

(4) Insert those additional edges in the colored interval graph one by one as watermarking constraints.

(5) Adjust the node color of the graph in such a way that no same color node should have a connecting edge.

(6) Modify the "register allocation" table after embedding all 'i', 'I', 'T' and '!' variables.

4.2.3 Tamper tolerance ability of proposed watermark

In the proposed triple-phase watermark, the 3rd phase watermark is independent of both the 1st and 2nd phase watermarks. Similarly, the 1st phase watermark is independent of both the 2nd and 3rd phase watermarks. Due to these advantages, the proposed watermarking method becomes extremely robust and tamper tolerant against threats like signature tampering attack, partial or complete IP core watermark removal attack, inserting unauthorized signature attack (details provided in Section 4.4) beside ownership abuse.

However, the watermarking variable of the 2nd phase is dependent on watermarking variables of 1st phase. Therefore, watermarking constraints of the 2nd phase may be affected due to alteration of the 1st phase watermark which is a limitation to this approach. Nevertheless, as discussed earlier, it is extremely difficult to identify in which of the phases the watermark has been inserted along with its encoding rule. Additionally, the initial register allocation table and colored interval graph before and after implanting the 1st and 2nd phase watermark is constant. In other words, the 3rd phase watermark is independent of others and capable of determining the real owner. Therefore, despite the possibility of tampering in 1st phase watermark the ownership can be preserved. Though tampering in the 3rd phase is also possible, the 1st and 2nd phase watermark is capable of protecting the ownership of the original IP, as the 3rd phase is independent of these two phases.

4.2.4 Proposed signature detection

In signature based IP core ownership protection mechanisms, determining the original owner to resolve ownership conflict is a crucially compulsory process. To achieve signature detection in the proposed triple-phase based watermarking approach, following inputs are required: a) decoded vendor signature, b) original application, c) watermarked IP core in the form of controller and datapath HDL. In this approach, the presence of IP owner's signature is first identified and then verified in each phase.

To identify the 1st phase watermark, the control steps of modified operations have to be extracted from the decoded signature and to verify, it has to be compared with the controller HDL of the watermarked IP core. To identify the 2nd phase watermark, the allocated hardware of the watermarked operations have to be extracted from the decoded signature and to verify, it has to be compared with the datapath HDL of the watermarked IP core with the help of original application and its corresponding inputs. To identify

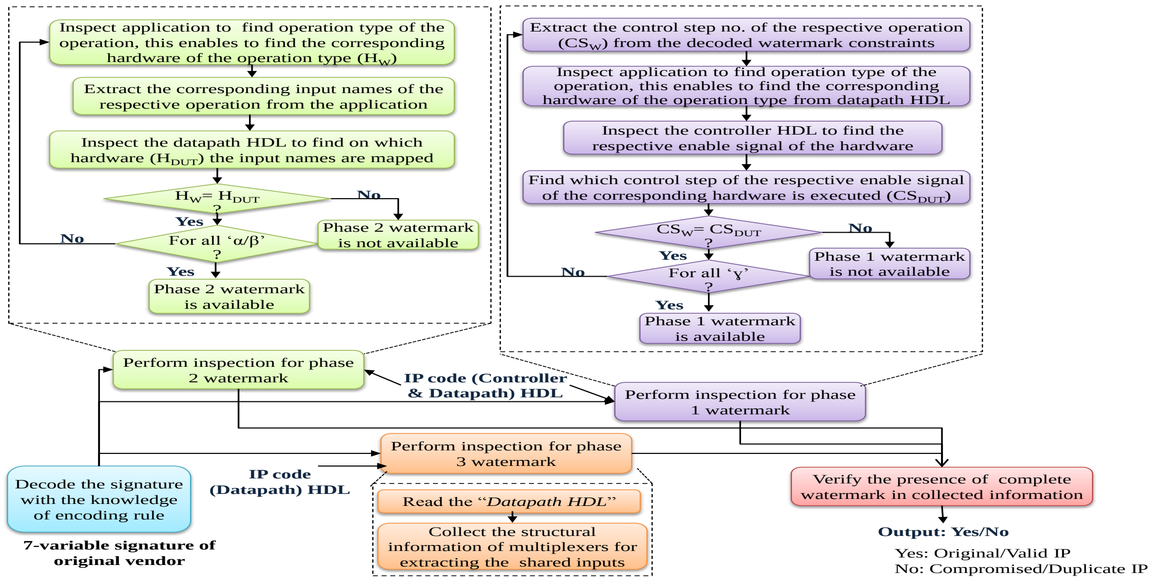

Inspect application to find operation type of the operation, this enables to find the corresponding hardware of the operation type (H_W)
Extract the corresponding input names of the respective operation from the application
Inspect the datapath HDL to find on which hardware (H_{DUT}) the input names are mapped
$H_W = H_{DUT}$?
No
Yes
Phase 2 watermark is not available
No
For all 'α/β' ?
Yes
Phase 2 watermark is available
Extract the control step no. of the respective operation (CS_W) from the decoded watermark constraints
Inspect application to find operation type of the operation, this enables to find the corresponding hardware of the operation type from datapath HDL
Inspect the controller HDL to find the respective enable signal of the hardware
Find which control step of the respective enable signal of the corresponding hardware is executed (CS_{DUT})
$CS_W = CS_{DUT}$?
No
Yes
Phase 1 watermark is not available
No
For all 'γ' ?
Yes
Phase 1 watermark is available
Perform inspection for phase 2 watermark
IP code (Controller & Datapath) HDL
Perform inspection for phase 1 watermark
Perform inspection for phase 3 watermark
IP code (Datapath) HDL
Decode the signature with the knowledge of encoding rule
7-variable signature of original vendor
Read the "Datapath HDL"
Collect the structural information of multiplexers for extracting the shared inputs
Verify the presence of complete watermark in collected information
Output: Yes/No
Yes: Original/Valid IP
No: Compromised/Duplicate IP

Figure 4.3: Signature detection process of proposed triple-phase watermark

the 3rd phase watermark, the shared inputs of each multiplexer should be fetched from the datapath HDL of the watermarked IP and to verify, it has to be compared with the extracted decoded signature of 3rd phase watermarking constraints. If all the constraints of the decoded signature are correctly present in the watermarked IP core then the signature verification is successful. Therefore, it can be concluded that the IP vendor who provided the signature is the original owner of that IP core. Fig.4.3 represents the complete signature detection process of the proposed triple phase watermarking approach.

4.2.5 Properties of generated watermark

The proposed watermarking methodology satisfies all the following desirable properties:

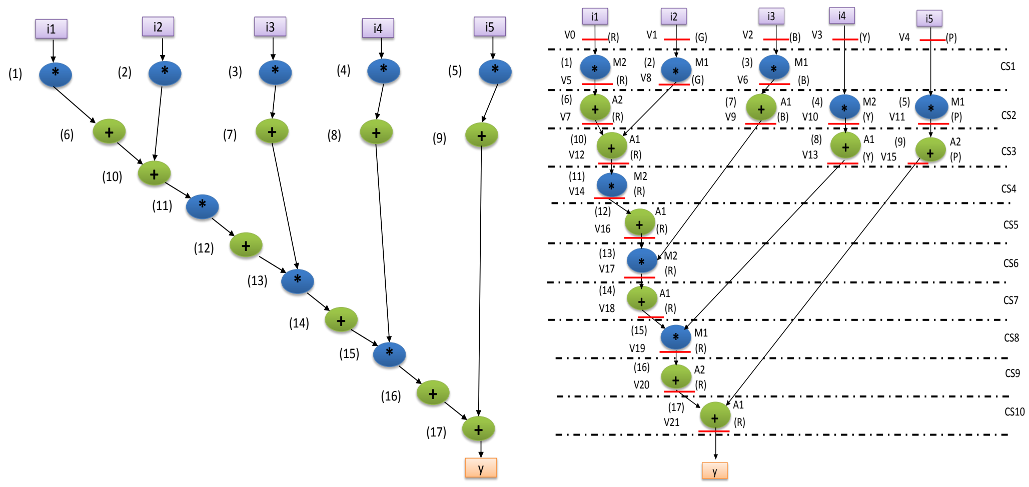
(a) **Low Implanting Cost:** In an ideal watermarking approach, the overall design overhead due to the insertion of additional constraints should be minimal. The proposed triple-phase watermarking incurs low design overhead compared to the original non-watermarked (baseline) design. The overall design cost is calculated based on two crucial design parameters, i.e., design area and execution latency (refer to Table 10.7).

(b) **High Robustness:** A crucial security aspect of an ideal watermarking approach is strong robustness to thwart signature removal attack. As the proposed triple-phase watermarking approach implants watermark in three different design phases of HLS, it ensures extreme robustness. Additionally, IP owner's signature is encoded through seven different variables to add another layer of protection. The proposed approach achieves higher robustness as compared to [30] & [51] (refer to Table 10.5).

(c) **Stronger Tamper Tolerance Ability:** Another crucial security aspect of an ideal watermarking approach is strong tamper tolerance ability i.e. partial removal should not impact the proof of ownership of an IP core. As the proposed approach distributes the watermarking constraints throughout the design in three different phases of HLS, therefore, ownership can be proven even after signature tampering. Additionally, as mentioned earlier the 1st and the 3rd phase of proposed watermark is independent of each other. The proposed approach achieves stronger tamper tolerant ability compared to [30] & [51] (refer to Table 10.6).

(d) **Minimal Watermark Implanting & Detection Time:** A robust signature may incur complex signature decoding and high embedding time, therefore the total runtime to generate a watermarked IP core may increase massively. In an ideal watermarking approach, the watermarking implanting process and the watermark detection process (in case of conflict) should be prompt. As the proposed approach is automated through a HLS framework therefore,watermark implanting time is fast (refer to Table 10.9). Moreover, in case of conflict of ownership the watermark detection process is speedy and straightforward for the original IP owner, who has the complete knowledge of encoding rules whereas, it is extremely difficult and complex to penetrate for an adversary or fraud IP owner.

(e) **Preserve the Functionality:** Unlike watermarking of multimedia artifacts (i.e. image, audio, video etc.), SoC based IP core watermarking has zero tolerance towards modification of quality and functionality of the design. Therefore, an ideal IP core watermarking process must preserve the correct functionality. The proposed triple-phase watermarking process carefully implants the watermarking constraints without altering the IP core functionality keeping the functionality of a watermarked IP core intact.



(a) (b)

Figure 4.4: Corresponding to the DWT benchmark: (a) data flow graph, (b) scheduled data flow graph based on 3(+), 3(*)

Table 4.1: “Timing table for non-critical operations” (before embedding watermark)

Operation No.	2	3	4	5	7	8	9
Control Step	1		2			3	

Table 4.2: “Functional unit allocation” table (before embedding watermark)

Odd CS	**Operation No.**	1	2	3	8	9	10	12	14	16
	Allocated FU	M2	M1	M1	A1	A2	A1	A1	A1	A2
Even CS	**Operation No.**	4	5	6	7	11	13	15	17	–
	Allocated FU	M2	M1	A2	A1	M2	M2	M1	A1	–

4.3 Motivational Example

This section presents a demonstration of the proposed triple-phase watermarking approach through a standard benchmark.

A DFG of DWT benchmark shown in Fig.4.4(a) is taken as input. According to the figure, the primary inputs of the benchmark are shown in purple block, final output is shown in orange block, multipliers and adders are indicated by blue and green nodes respectively and the corresponding operation number of each node is indicated by an integer number. It is then scheduled based on user provided hardware resources i.e. 3 adders and 3 multipliers

Table 4.3: "Register allocation" table (before embedding watermark)

CS	(R)	(G)	(B)	(Y)	(P)
0	V0	V1	V2	V3	V4
1	V5	V8	V6	V3	V4
2	V7	V8	V9	V10	V11
3	V12	–	V9	V13	V15
4	V14	–	V9	V13	V15
5	V16	–	V9	V13	V15
6	V17	–	–	V13	V15
7	V18	–	–	V13	V15
8	V19	–	–	–	V15
9	V20	–	–	–	V15
10	V21	–	–	–	–

using list scheduling algorithm **(step A.1)**. After that, the functional unit allocation of each operation and the register allocation of each storage variable (V0-V21) are performed **(step A.2)**. The IP cores in this approach are obtained from two distinctive IP vendors. To illustrate that, out of 3 adders, two are obtained from vendor type 1 (A1) and one is obtained from vendor type 2 (A2). Similarly, out of the 3 multipliers, two are obtained from vendor type 1 (M1) and one is obtained from vendor type 2 (M2). Fig.4.4(b) shows the scheduled DFG after random functional unit allocation and register allocation (before watermarking). The register allocation (before watermark) is performed based on colored interval graph (shown in Fig.4.5(a)). The graph has 22 nodes indicating 22 storage variables i.e., V0 to V21 which are executed through 5 registers represented by the color of the node i.e. register red (R), register green (G), register blue (B), register yellow (Y) and register purple (P). The timing table for non-critical operations, functional unit allocation table and register allocation table of the non-watermarked design are generated as shown in Table4.1, Table4.2 and Table4.3 respectively **(step A.3)**. All the operations are sorted control step wise in ascending order according to **step A.4**. A 12-digits owner's signature is selected as: "$\gamma\gamma\alpha\beta\alpha\beta\beta iIIT$" **(step A.5)**.

The vendor's signature and its corresponding decoded meaning is shown in Table4.4 . The first 'γ' digit of the signature moves opn. 2 from CS 1 to CS 2. According to the rule mentioned in **B.1**, the second 'γ' digit moves opn. 9 from CS 3 to CS 4. This is because op-

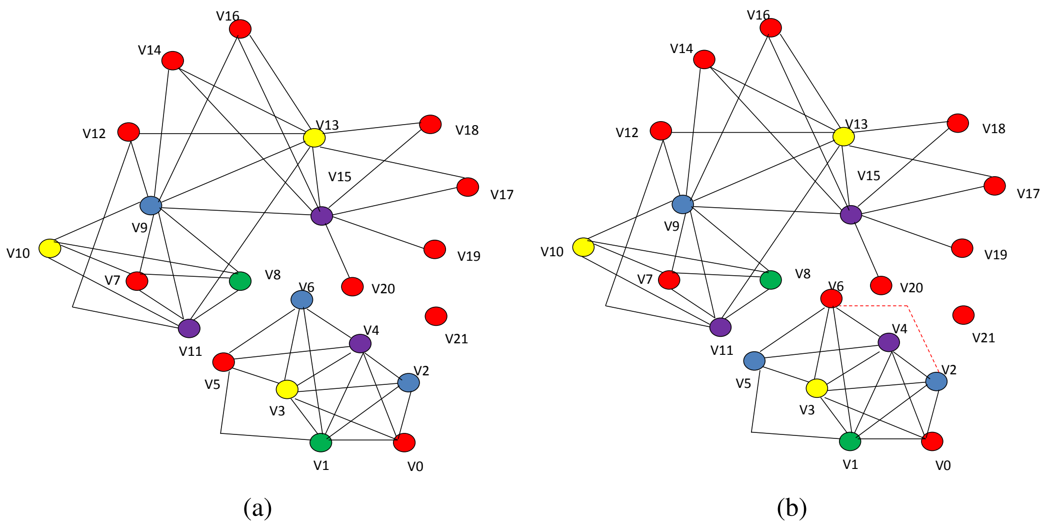


Figure 4.5: Corresponding to the color interval graph of the input application: (a) before implanting watermark, (b) after implanting watermark

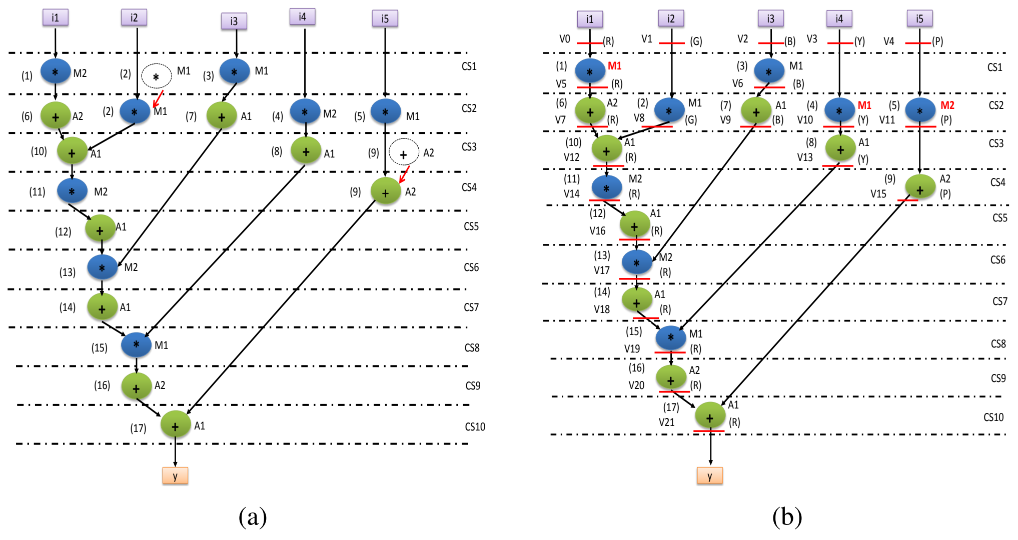


Figure 4.6: Corresponding to the proposed triple phase watermark: (a) after implanting 1st phase watermark, (b) after implanting 1st and 2nd phase watermark

erations 3, 4 & 5 violate rule mentioned in B.1(i), opn. 7 violates rule mentioned in B.1(ii) and opn. 8 violates rule mentioned in B.1(iii). Fig.4.6(a) shows the corresponding scheduled DFG and Table4.5 represents the modified timing table for “non-critical operations” **(step B.2)**.

All the ‘α’ and ‘β’ digits must be implanted on top of the 1st phase watermarked design **(step C.1)**. The third signature digit modifies the hardware assignment of opn.1 from M2

Table 4.4: Vendor signature and its decoded meaning

Desired signature (12-digits)	Corresponding operation to shift (Phase 1)	Allocated FU type (Phase 2)	Additional edges to insert between nodes in the colored interval graph (Phase 3)	Observations
γ	Opn. 2 from CS 1 to CS 2	–	–	**CS shift to be done**
γ	Opn. 9 from CS 3 to CS 4	–	–	**CS shift to be done**
α	–	Opn. 1 with vendor 1	–	**FU reallocation to be done**
β	–	Opn. 2 with vendor 1	–	No change
α	–	Opn. 3 with vendor 1	–	No change
β	–	Opn. 4 with vendor 1	–	**FU reallocation to be done**
β	–	Opn. 5 with vendor 2	–	**FU reallocation to be done**
i	–	–	(V2, V3)	No change
I	–	–	(V2, V4)	No change
I	–	–	(V2, V6)	**New edge to be added**
T	–	–	(V1, V2)	No change
!	–	–	(V0, V1)	No change

Table 4.5: "Timing table for non-critical operations" (after embedding 1st phase watermark)

Operation No.	3	**2**	4	5	7	8	**9**
Control Step	1		2			3	4

Table 4.6: "Functional unit allocation" table (after embedding 2nd phase watermark)

Odd CS	**Operation No.**	**1**	3	8	10	12	14	16	–	–	–
	Allocated FU	**M1**	M1	A1	A1	A1	A1	A2	–	–	–
Even CS	**Operation No.**	**4**	2	**5**	6	7	9	11	13	15	17
	Allocated FU	**M1**	M1	**M2**	A2	A1	A2	M2	M2	M1	A1

(multiplier obtained from vendor type 2) to M1 (multiplier obtained from vendor type 1). Similarly, the sixth signature digit modifies the hardware assignment of opn.4 from M2 to M1 (Note: the fourth and fifth digit of the signature are already satisfied in the design). Similarly, rest of the 'α' and 'β' digits will be implanted **(step C.2)**. Fig.4.6(b) shows the corresponding scheduled DFG and Table4.6 represents the modified "functional unit allocation" table **(step C.3)**.

All the 'i', 'I', 'T' and '!' digits must be implanted on the top of 1st and 2nd phase watermarked design **(step D.1)**. The color interval graph constructed earlier is shown in Fig.4.5(a) **(step D.2)**. According to the decoded signature shown in Table4.4 , the list of additional edges are between nodes (V2 and V3), (V2 and V4), (V2 and V6), (V1 and V2),

Table 4.7: "Register allocation" table (after embedding 3rd phase watermark)

CS	(R)	(G)	(B)	(Y)	(P)
0	V0	V1	V2	V3	V4
1	**V6**	V1	**V5**	V3	V4
2	V7	V8	V9	V10	V11
3	V12	–	V9	V13	V11
4	V14	–	V9	V13	V15
5	V16	–	V9	V13	V15
6	V17	–	–	V13	V15
7	V18	–	–	V13	V15
8	V19	–	–	–	V15
9	V20	–	–	–	V15
10	V21	–	–	–	–

and (V0 and V1) **(step D.3)**. Among these five edges only the edge between nodes (V2 and V6) is not present in the graph in advance **(step D.4)**. The aforementioned edge forces storage variable V2 and V6 to be executed through distinct register, therefore node color of V2 and V6 must be different which is same now (see Fig.4.5(a)). Hence to incorporate the watermarking constraint, the allocated register i.e. (B) for storage variable V6 needs to be swapped with another storage variable in same control step. Swapping V6 with V5 does not violate any constraints, therefore, storage variable V6 is now allocated to (R) (previously allocated to (B)) and storage variable V5 is now allocated to (B) (previously allocated to (R)) **(step D.5)**. The final modified graph is shown in Fig. 4.5(b). Fig. 4.7 shows the corresponding scheduled DFG and Table4.7 represents the modified "register allocation" table **(step D.6)**.

4.4 Threat Scenarios of Fraudulent Ownership

This section presents the possible threat scenarios of the fraudulent IP core ownership and its resolution through the proposed triple-phase watermarking process. Let's assume 'A' is an IP vendor, designed a watermarked IP core ('D_w') with her signature and sold it to 'B'. In that case, 'B' can claim his ownership through three possible scenarios [30]:

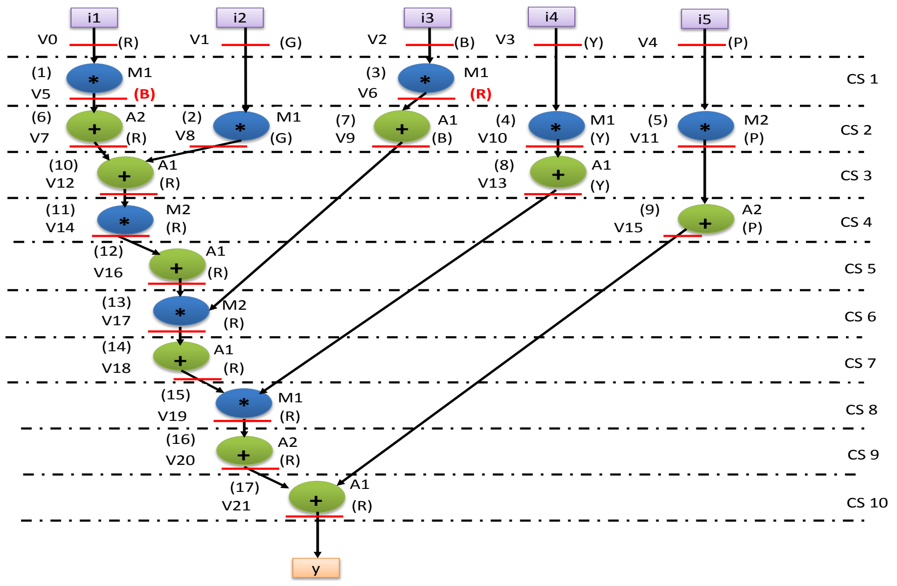


Figure 4.7: Final watermarked design after implanting 1st, 2nd and 3rd phase watermark

4.4.1 Extracting unintended signature

After purchasing 'D_w', 'B' may extract the complete information of it through reverse engineering. He may then claim any random/arbitrary design information as his watermarking constraints, thus may claim his ownership for 'D_w'. In such scenario, the entity having more meaningful and consistent watermark will be awarded as the original owner of 'D_w'. 'B's claim may valid for a single design, but will be invalided for other watermarked design. Therefore, 'B's claim will not be strong enough to prove his ownership. On the contrary, 'A's watermarking constraints will be more meaningful and consistent for all watermarked design. For example, according to Fig.4.6(b), 'B' may claim "all operations of CS 1 must be allocated to Vendor 1" as his signature, which may work for this particular design, but will be invalid for other watermarked designs. As in the proposed approach each signature variable has an encoding meaning, therefore, it is capable to resolve this scenario.

4.4.2 Inserting unauthorized signature

After purchasing 'D_w', 'B' may embed his own signature on it and then may claim his ownership for 'D_w'. In such case, 'B' applies his signature on the top of the 'A's signature. Therefore, the design will contains both 'A's and 'B's signature. However, 'A's design only contains her signature, whereas, Bs design contains signature of both ‘A’ and ‘B’. As in the proposed triple-phase watermarking approach has strong tamper tolerance ability, therefore, it is capable to resolve this threat scenario.

4.4.3 Tampering original signature in the design

After purchasing ‘D_w’, ‘B’ may remove all the signature to make ‘D_w’ a non-watermarked design. In such case, the watermarking constraints must be independent of each others and distributed throughout the design. Thus, makes complete watermark removal process extremely difficult. Additionally, sometimes tampering of implanted watermarking constraints may force to perform all the pre-synthesis steps from the beginning, which is a costly and time-consuming process. As in the proposed triple-phase watermarking approach implants the signature during in-systhesis steps of HLS with strong tamper tolerance ability, therefore, it approach is capable to resolve this threat scenario.

Note: The corresponding experimental results of the proposed methodology is explained in Chapter 10 Section 10.2.

4.5 Summary

This chapter presents a novel robust IP core watermarking methodology for CE systems. The work proposes a 7-variable signature encoding scheme to implant the IP owner’s watermark in the design. The watermarking constraints are implanted in three different phases of HLS for the first time in the literature. Additionally, as the watermarking constraints are implanted at the higher design abstraction level, therefore it will automatically propagate to the lower abstraction level. Thus, provides a highly robust IP core protection not at the higher level but also at the lower level of the design. The implanted watermark satisfies all the desirable properties such as covertness, robustness, strong tamper tolerance ability, low signature implanting & detection complexity and correct functionality. This proposal has several unique features: (a) the proposed algorithm encode the owner’s signature through a novel 7-variable signature encoding scheme(b) watermarking constraints are implanted covertly during scheduling, hardware allocation and register allocation phase of HLS without changing the actual functionality (c) ensures higher robustness while incurring low design overhead. Finally, experimental results over the standard applications indicate an average reduction in the design cost of 7% and 6% compared to two recent approaches. Thus provides an significant advancement in IP core protection for CE systems.

Chapter 5

Symmetrical IP Core Protection of CE Systems

The last two chapters discussed the watermarking approaches to protect an IP core seller from various security threats. However, in the design supply chain of an IP core, along with the IP seller another entity involved is: IP core buyer. An IP seller also known as IP vendor is the manufacturer of an IP core whereas an IP buyer also known as IP user is the purchaser of that IP core. To design a secured IP core, protection of both the entities against threats is extremely essential.

The protection of an IP core should ensure the buyer to exercise exclusive user right, i.e. to not allow the IP seller to resell/redistribute the same IP copy to other users. This happens when an IP buyer procures an IP core with his/her own custom specifications. Therefore, an exclusive one-to-one mapping between both the parties exists. Implanting buyer's signature (known as fingerprint) into an IP core design, facilitates detection of illegal redistributed/resold copies of an IP core by a deceitful IP seller [20]. Similarly, an IP seller must protect his/her design from piracy and false claim of ownership before selling it to an IP buyer. Implanting seller's signature (known as watermark) into an IP core design protects an IP core from ownership abuse. This necessitates a robust symmetrical protection of reusable IP core from both buyer's and seller's perspective.

This chapter presents a novel symmetrical IP core protection methodology for DSP core of CE systems to protect the rights of both the entities (shown in Fig. 5.1). It implants

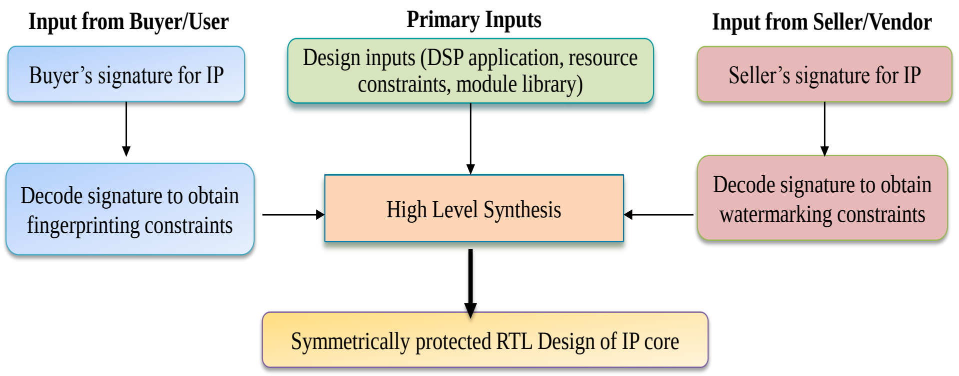

Input from Buyer/User
Primary Inputs
Input from Seller/Vendor
Buyer's signature for IP
Design inputs (DSP application, resource constraints, module library)
Seller's signature for IP
Decode signature to obtain fingerprinting constraints
High Level Synthesis
Decode signature to obtain watermarking constraints
Symmetrically protected RTL Design of IP core

Figure 5.1: Proposed Symmetrical IP core Protection during high level synthesis.

signature of both the IP buyer and the IP seller into the design to safeguard their rights. As shown in Fig. 5.1, firstly, the fingerprinting constraints are implanted during scheduling and register allocation phase of HLS, thereafter, the watermarking constraints are implanted during register allocation phase of HLS. Thus the IP core design has both fingerprinted and watermarked constraints in addition to its normal design constraints (i.e. module library, resource configurations etc.). It also ensures strong resistance against complete removal or malicious modification of signature. In case of ownership conflict, both the implanted signature can be detected by inspecting the datapath and controller HDL file of the IP core. This is the first work in IP core protection of CE systems that provides symmetrical protection to secure both IP buyer and IP seller right.

The chapter is organized as follows: Section 5.1 discusses the problem formulation, threat model and the target platform of the proposed approach; Section 5.2 explains the proposed methodology; Section 5.3 demonstrates the proposed symmetrical IP core protection approach through a standard application; finally, the summary of this chapter is presented in Section 5.4. (*NOTE: All the abbreviations and taxonomy are listed in 'Acronyms' and 'Nomenclature' section respectively.*)

5.1 Proposed Approach: Problem Formulation, Threat Model and Target platform

This section presents the problem formulation, threat model and target platform of the proposed approach.

5.1.1 Problem formulation

For a DSP core given in the form of a DFG, design an symmetrically protected IP core (after implanting fingerprint and watermark) using optimal solution, within buyer's provided resource constraints, $(X_i) = N(R_1), N(R_2), ..., N(R_D)$. The problem can be formulated as follows:

Minimize: Design cost (A_T, L_T) of symmetrically protected (through robust watermarking and fingerprinting) IP core for design solution (X_i),

Subject to: $A_T \leq A_{cons}$ and $L_T \leq L_{cons}$.

5.1.2 Threat model

The objective of a IP core security is to protect both the IP buyer and the IP seller. Therefore, the threat model of the proposed symmetrical approach is as follows:

IP Buyer's perspective

Detecting illegally redistributed/resold IP core copies in the market by a deceitful IP seller [3].

IP Seller's perspective

Protection against (i) fraudulent ownership (ii) ownership conflict/abuse and (iii) IP infringement.

5.1.3 Target technology/platform

The proposed protection methodology can be easily integrated with any EDA tools. HDL or any high level language used for IP generation can easily merge with the proposed technique in the design tools.

5.2 Proposed Methodology

The proposed symmetrical IP core protection approach covertly inserts fingerprinting and watermarking constraints to protect the IP buyer as well as the IP seller. Based on the signature constraints provided by a buyer, an IP seller inserts buyer's fingerprint with his/her own watermark, safeguarding the privileges of both the entities. The multiplexing process for each resource is generated to identify the resource interconnectivity, multiplexer size and inputs/outputs of the watermarked and fingerprinted design. Using the generated multiplexing scheme of each resource, the complete datapath and timing description of the control unit of the DSP IP core is developed. Thus the final output of a symmetrically protected RTL design of an IP core is generated as datapath and controller represented as HDL files.

5.2.1 Evaluation models

This section discuss the proposed models used to evaluate the design cost of an symmetrically protected IP core in terms of hardware area and execution latency.

Proposed hardware area model

Total area of the watermark and fingerprint implanted IP core is expressed by the following model:

$$A_T = \sum_{i=1}^{m} \left(A(R_i) * R_i \right) + A(mux) * N(mux) + A(buffer) * N(buffer) \tag{5.1}$$

Proposed execution delay model

Total execution delay of the watermark and fingerprint implanted IP core is expressed by the following model:

$$L_T = \sum_{c.s=1}^{n} Max(D(op_i), ..D(op_r)) \tag{5.2}$$

Proposed design cost model

Based on the calculated value of the total hardware area and total execution delay the design cost of an symmetrically protected IP core can be evaluated by the following model:

$$C_f(X_i) = w_1(L_T)/L_{max} + w_2(A_T)/A_{max} \tag{5.3}$$

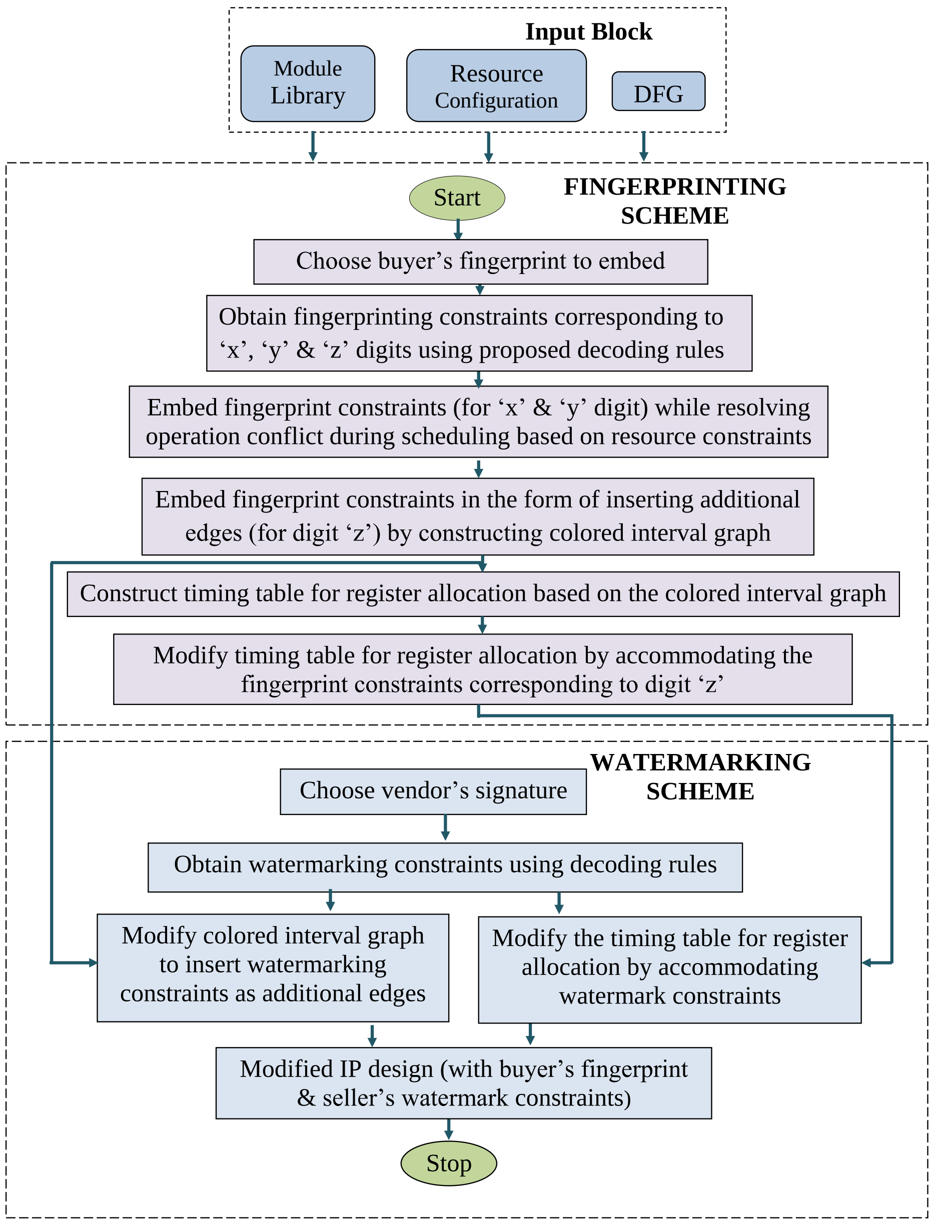
5.2.2 Proposed signature encoding

In the proposed approach, fingerprint constraints are implanted in both scheduling and register allocation phase, while watermarking constraints are implanted in the register allocation phase of HLS. Fingerprinting constraints in the scheduling phase are achieved by forcing specific operations in specific CS during the schedule conflict resolution process. In other words, instead of employing priority resolver functions or random break, a novel fingerprint encoding rule is proposed to select which operation is to be assigned to which CS. It is a covert way of inserting buyer fingerprint since during regular operation scheduling conflict the fingerprint is inserted with zero hardware and minimal latency overhead. Further, the fingerprint and watermarking constraints during register allocation phase of HLS are realized with the concept of CIG, where the nodes of the graph represent storage variables and the edges represent overlapping lifetime between corresponding storage variables. Namely, if two storage variables exist in the same control step of a scheduler, their lifetime overlapping; there will be an edge between them. Adding additional edges as constraints will force storage variables of a CIG to execute through distinct registers. The flow diagram of the proposed symmetrical IP protection is shown in Fig. 5.2. The detailed encoding rules of the proposed symmetrical approach are presented below:

Fingerprint encoding

In the proposed symmetrical IP core protection approach, the fingerprint encoding is as follows:

- x = Force even operation in odd control step while resolving scheduling conflict in scheduling phase.



Input Block
Module Library
Resource Configuration
DFG
Start
FINGERPRINTING SCHEME
Choose buyer's fingerprint to embed
Obtain fingerprinting constraints corresponding to 'x', 'y' & 'z' digits using proposed decoding rules
Embed fingerprint constraints (for 'x' & 'y' digit) while resolving operation conflict during scheduling based on resource constraints
Embed fingerprint constraints in the form of inserting additional edges (for digit 'z') by constructing colored interval graph
Construct timing table for register allocation based on the colored interval graph
Modify timing table for register allocation by accommodating the fingerprint constraints corresponding to digit 'z'
WATERMARKING SCHEME
Choose vendor's signature
Obtain watermarking constraints using decoding rules
Modify colored interval graph to insert watermarking constraints as additional edges
Modify the timing table for register allocation by accommodating watermark constraints
Modified IP design (with buyer's fingerprint & seller's watermark constraints)
Stop

Figure 5.2: Design flow of the proposed symmetrical IP core protection during

- y = Force odd operation in even control step while resolving scheduling conflict scheduling phase.

- z = Add additional edge between node pair number (odd, odd) in the colored interval graph.

In order to apply the fingerprint encoding for ’x’ and ’y’, operations are kept in a sorted list. Similarly to apply fingerprint encoding for ’z’, the storage variables in the scheduling must be kept in a sorted order. The strength of buyer’s fingerprint increases with its signature size. The proposed approach is flexible for accepting any large sized fingerprint. As evident, the proposed fingerprint signature scheme comprises three unique variables and each variable is mapped with a unique encoding that does not add much overhead to the design while preserving robustness.

It is recommended that while selecting buyer’s fingerprint the signature should have higher number of ‘x’ and ‘y’ digits than ‘z’ digits. The reason being that insertion of ‘x’ and ‘y’ digit imposes zero hardware area overhead and negligible execution latency overhead as both digits are inserted during operation scheduling conflict resolution. On the other hand, numerous insertions of ’z’ digits may result in hardware overhead in terms of the number of storage hardware. However, it is also crucial to add another protection layer by implanting fingerprint constraint in the register allocation stage.

Watermark encoding

In the proposed symmetrical approach, watermarking is applied using the methodology proposed in [50]. In [50], additional constraints for watermark are inserted in the register allocation step of HLS by adding additional edges between the nodes of a CIG. Adding these additional edges will force storage variables of a CIG to execute through distinct registers. This approach is flexible for adding any number of additional edges in the register allocation step. The strength of the watermark increases with the number of additional edges (i.e., the watermark size). Though robustness increases with increase in signature size, a large signature may incur significant hardware overhead, which makes it is necessary to employ a robust watermarking scheme that provides substantial protection with minimal overhead. The proposed symmetrical approach employs a multi-variable watermark creation mechanism from [50].

As proposed in [50], the mechanism of watermark creation consists of four different variables. Each variable carries a different encoding rule. The seller is free to choose any random combination of these four variables as his/her watermark. Each variable adds an additional edge between two specific nodes encoded as the rule of that variable. In other words, selection of node pair from CIG to add an additional edge is defined in the variable as encoding mechanism. The watermark encoding is as follows:

- i = encoded value of edge with node pair as (prime, prime)
- I = encoded value of edge with node pair as (even, even)
- T = encoded value of edge with node pair as (odd, even)
- ! = encoded value of edge with node pair as (0, any integer)

Although 'z' digit of fingerprint and 'i', I', T' and '!' digits of seller watermark both embed constraints in register allocation phase, however, their encoding rules are different resulting into completely different constraints.

5.2.3 Proposed signature implanting process

The broad and detailed steps for generating symmetrical IP is described here. It is achieved by implanting fingerprint and watermark.

Fingerprint implanting process

Broad steps:

1. Select the desired buyer signature.
2. Decode the buyer signature to its equivalent fingerprinting constraints.
3. Use the decoded constraints to perform scheduling during operation conflict.

4. Assign the storage variables to registers from the scheduling using the concept of colored interval graph.
5. Insert additional edges in the colored interval graph based on decoded constraints and perform re-assignment of register allocation.

Detailed steps:

1. Accept a preferred signature for fingerprint as any random combination of: x, y, & z from the buyer.
2. Create a sorted list of operations of the DFG as per their operation number in increasing order.
3. Convert fingerprint signature corresponding to digits 'x' and 'y' into equivalent constraints that help in resolving schedule operation conflict.
4. Initiate DFG scheduling based on the given resource configuration.
5. In case of operation conflict during scheduling, use obtained fingerprint constraints (in step 3) to resolve and generate a scheduling (Note: When there are no fingerprinting constraints available, conflict is broken based on sorted ordering).
6. Assign storage variables in the scheduling and create a colored interval graph to find the minimum number of registers required for allocation.
7. Sort the assigned storage variables as per their number in increasing order.
8. Construct a timing table for register allocation based on the generated colored interval graph.
9. Generate a list of extra edge pairs conforming to z digit(s) by traversing the sorted storage variables in step 7.
10. Insert the additional edges in the colored interval graph generated in step 6 based on obtained fingerprint constraints.

After performing the above steps, a buyer fingerprint implanted IP design is generated. This IP design is used as an input for further implanting the seller watermark and finally generating the symmetrical IP core.

Watermark implanting process

Broad steps:

1. Select desired seller signature.
2. Decode the seller signature into its equivalent watermarking constraints.
3. Construct a colored interval graph to represent registers required for storage variables from the fingerprint implanted scheduling.
4. Perform re-allocation of register if necessary while inserting extra edges in the colored interval graph based on watermarking constraints.

Detailed steps:

1. Accept the fingerprint embedded IP design (schedule and register allocation) as an input.
2. Sort the assigned storage variables as per their number in increasing order.
3. Accept the watermark of the seller in the form of i, I, T, and !, where these characters hold the encoded meaning of additional edges to be inserted.
4. Create a list of additional edge pairs corresponding to its encoded values by traversing the sorted nodes.
5. Insert the additional edges in the colored interval graph of fingerprint embedded design as watermarking constraints.
6. Modify the timing table of the register allocation based on the generated watermark.

Post-execution of the above steps, on top of the buyer fingerprint a seller watermark implanted IP design is generated. Consequently, the IP design has signatures of both the entities. This ensures a reusable IP core with symmetrical protection.

5.2.4 Signature detection process

Signature detection is a compulsory process for any signature-based IP core protection mechanism. This process extracts and validates the implanted signature from the IP core design to protect the rights of the corresponding entity. The signature detection for the proposed approach is a two-step process:

- *Design inspection:* The objective of this step is to extract the appropriate design information in terms of structural property, specification, etc., from the received IP core.

- *Signature verification:* The objective of this step is to verify the decoded constraints of the signatures in the extracted information from the received IP design signifying the signature and the encoding meaning of each digit needs to be known. Consequently, the list of additional constraints corresponding to the signature can be generated. Lastly, the presence of each constraint must be verified in the extracted controller and datapath information of the received IP core.

5.2.5 Properties of generated signatures

In the proposed symmetrical IP core protection approach, the embedded signatures have the following properties:

- *Low embedding cost:* An ideal signature must incur minimal hardware area and execution latency overhead. Therefore, any symmetrically protected IP core design solution should impose overhead as low as possible. In the proposed approach, a low-cost watermarked-fingerprinted design solution has been explored.

- *Resiliency:* A signature should be so strong and robust that it is difficult to remove or tamper. The proposed watermark and fingerprint are both equipped with strong protection which makes the watermark and fingerprint extremely hard to remove or tamper without complete knowledge of both the encoding rules. Additionally, it is also difficult to identify in which HLS step the signatures has been implanted.

Besides, the additional constraints are distributed all over the IP design which renders it more resistant against removing or tampering.

- *Fault tolerance:* An ideal signature must be distributed enough throughout the design so that partial removal of the signature constraints does not prevent the proof of ownership. As the proposed signature scheme distributes the additional constraints throughout the design, ownership remains preserved even after partial removal or tampering of signature.

- *Adaptability to any CAD Tool:* An ideal signature must be easily adaptable for any modern CAD tool. The proposed symmetrical methodology is compatible with any new generation CAD tool and can be integrated with any modern CAD tool flow seamlessly.

- *Signature creation and detection time:* An ideal signature creation process should not be too complex to generate a signature. In the proposed approach, the signature creation time for both watermark and fingerprint is significantly less. Further, the signature detection process is simple and straightforward for a genuine entity with complete knowledge of encoding rules.

5.3 Motivational Example

This section gives a demonstration of the proposed symmetrical IP core protection mechanism through a standard benchmark. The first subsection illustrates the proposed fingerprint creation and insertion process while the second subsection illustrates the proposed watermark creation and insertion process. As mentioned earlier, fingerprint digits ‘x’ and ‘y’ are implanted in the scheduling step of HLS by assigning specific operations to specific CS during schedule conflict resolution process while variable ’z’ adds additional edges in register allocation phase by forcing specific storage variables to execute in distinct registers. On the other hand, seller watermarking constraints in the form of ’i’, I’, T’ and ’!’ are embedded only in the register allocation step of HLS by forcing specific storage variables to execute in distinct registers.

Table 5.1: Fingerprint and its meaning

Desired fingerprint (7-digit)	**Corresponding operation assigned in control step based on fingerprint**	**Corresponding additional edges to add between nodes in the colored interval graph**
x	assign operation 2 in CS1	-
x	assign operation 4 in CS1	-
y	assign operation 5 in CS2	-
x	assign operation 6 in CS3	-
y	-	-
z	-	(V1, V3)
z	-	(V1, V5)

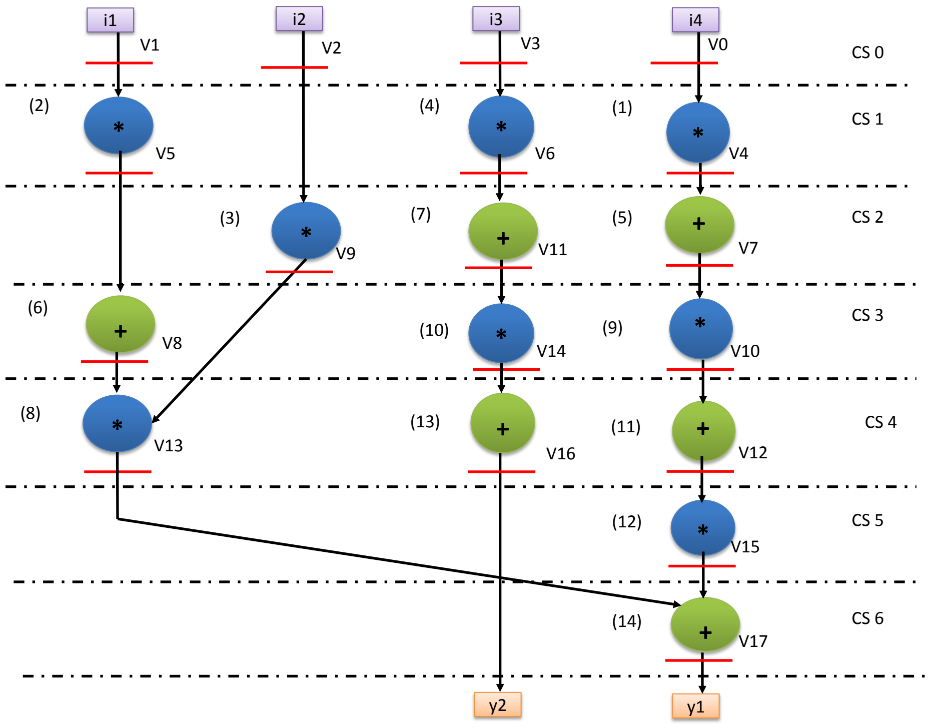


Figure 5.3: Scheduling of MESA DFG with 2 adders and 3 multipliers (after implanting proposed fingerprint)

5.3.1 Example of fingerprint implanting

Assuming MESA data flow graph and resource configuration (2 adders and 3 multipliers) provided as inputs. The desired buyer signature is chosen (using an arbitrary combination of 'x', 'y' and 'z') as discussed in section 5.2.3. For the purpose of demonstration, a 7-digit

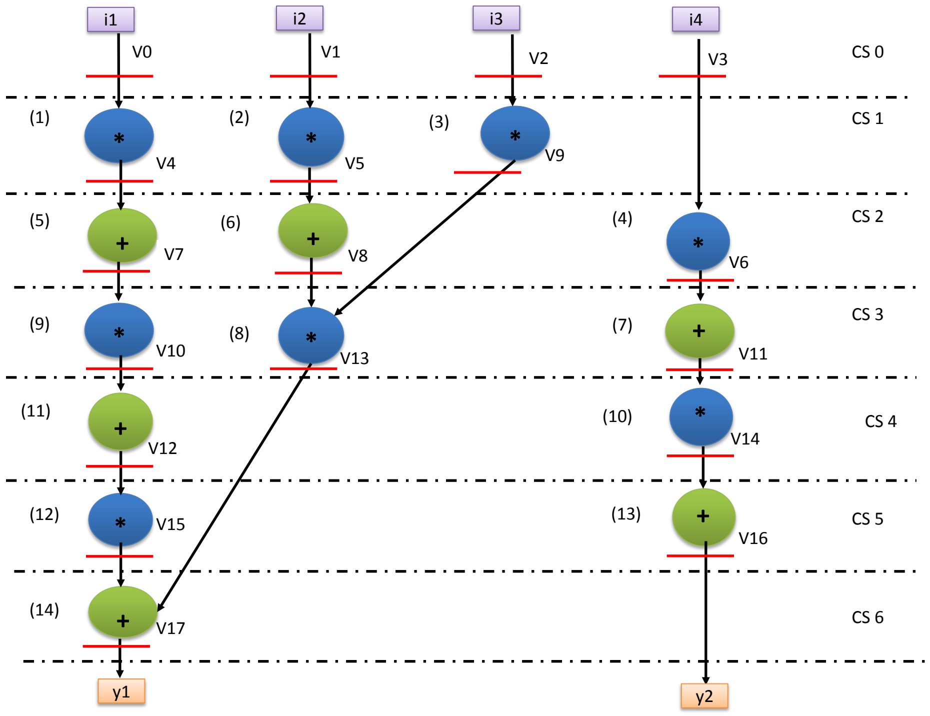


Figure 5.4: Scheduling of MESA DFG with 2 adders and 3 multipliers (with no buyer signature implanted)

Table 5.2: Timing table for register allocation after implanting ‘x’ & ‘y’ but before implanting ‘z’ digits as fingerprint constraints

Control Step	**Red (R)**	**Green (G)**	**Blue (B)**	**Yellow (Y)**
0	V0	V1	V2	V3
1	V4	V5	V2	V6
2	V7	V5	V9	V11
3	V10	V8	V9	V14
4	V12	V13	-	V16
5	V15	V13	-	-
6	V17	-	-	-

buyer fingerprint signature is selected: ‘x x y x y z z’. The fingerprint signature is then decoded. The corresponding decoded meaning is shown in Table 5.1 . The decoded constraints are obtained using the proposed encoding meaning of each fingerprint digit. Next, the MESA data flow graph needs to be scheduled using the resource constraint specified. In the first odd CS, there are four ready multiplication operations viz. opn 1, opn 2, opn

Table 5.3: Timing table for register allocation after embedding additional edges as fingerprint constraints

Control Step	Red (R)	Green (G)	Blue (B)	Yellow (Y)
0	V0	**V1**	V2	V3
1	**V5**	V4	V2	V6
2	V5	V7	V9	V11
3	V10	V8	V9	V14
4	V12	V13	-	V16
5	V15	V13	-	-
6	V17	-	-	-

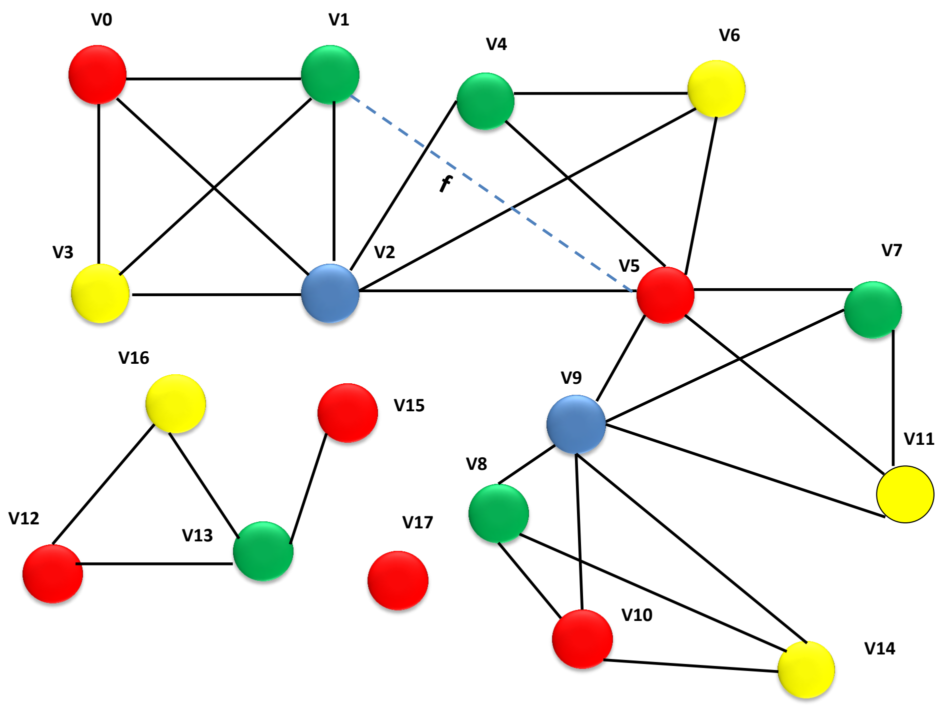


Figure 5.5: Colored interval graph with additional edges as fingerprinting constraints indicated through blue dotted line

3 and opn 4, but only 3 multiplier resources are available. This indicates the existence of a schedule operation conflict scenario. The decoded fingerprint constraints will now be inserted in the schedule while resolving the operation conflict. For example, first digit 'x' forces even multiplication operation (i.e., opn 2) into odd control step (i.e., CS 1) during schedule conflict resolution. Similarly the next digit 'x' forces multiplication opn 4 in odd

CS 1 during schedule conflict resolution. The third multiplication is scheduled in CS 1 by simply choosing the first operation in the sorted list. Thus opn 1 is selected and scheduled in CS 1. Similarly, odd opn 5 has been scheduled in even CS 2 after resolving operation conflict through fingerprint encoding of digit 'y'. Further even opn 6 has been scheduled in odd CS 3 after resolving operation conflict through fingerprint encoding of digit 'x'.This process of embedding fingerprint constraints in scheduling continues until all operation conflicts are resolved. Since no further schedule operation conflict exists, hence the last three fingerprint digits could not be embedded. However, scheduling is normally continued until all operations of DFG are scheduled based on resource constraint. The final schedule with buyer fingerprint embedded is shown in Fig. 5.3. Its schedule counterpart where no fingerprint embedding is performed during conflict resolution is shown in Fig. 5.4. As evident with absolutely zero hardware area and zero latency overhead, the buyer fingerprint is successfully embedded in the design without distorting the functionality.

Next, the storage variables (V1 - V17) are assigned in the obtained schedule and its corresponding CIG is created. Two additional edges corresponding to two 'z' fingerprint digits are to be added as constraints in the CIG (representing register allocation phase). Although two additional edges have to be added, but coincidently an edge between (V1, V3) already exists by default. Thus, only new edge between (V1, V5) needs to be added (marked by a blue dotted line in CIG of Fig. 5.5). The information of this new edge added in the CIG needs to be accommodated in the timing table of register allocation (shown in Table 5.2). This is accommodated by executing storage variables V1 and v5 in distinct registers (Red and Green). The modified controller with fingerprinting constraints is shown in Table 5.3.

5.3.2 Example of watermark implanting

After embedding buyer's fingerprint, it now acts as an input for embedding seller watermark. For demonstration, a 7-digit watermark signature is selected: 'i i i I i T !'. Its corresponding decoded constraints as additional edges are shown in Table 5.4. As evident from the CIG in Fig. 5.6, only two additional edges ((V2, V7), (V2, V9)) as constraints

Table 5.4: Watermark and its meaning

Desired watermark (7-digit)	**Corresponding additional edges to insert between nodes of coloured interval graph**
i	(V2,V3)
i	(V2,V5)
i	(V2,V7)
I	(V2,V4)
i	(V2,V9)
T	(V1,V2)
!	(V0,V1)

Table 5.5: Final timing table for register allocation after implanting fingerprint and watermark

Control Step	**Red (R)**	**Green (G)**	**Blue (B)**	**Yellow (Y)**
0	V0	V1	**V2**	V3
1	V5	V4	**V2**	V6
2	V5	**V7**	V11	**V9**
3	V10	V8	V14	V9
4	V12	V13	-	V16
5	V15	V13	-	-
6	V17	-	-	-

need to be inserted in the CIG. This is because the remaining five edges are already present in the graph by default. Therefore, due to an extra edge being inserted between V2 and V9, both are forced to execute through distinct registers. V2 and V7 are anyway executed through distinct registers as evident in Table 5.3. The final modified timing table for register allocation is shown in Table 5.5. It is clear that zero hardware area and latency overhead have been incurred after embedding seller's watermark as well.

Thus for MESA benchmark, embedding both buyer fingerprint and seller watermark for symmetrical IP protection incur absolutely zero area and latency design.

Note: The corresponding experimental results of the proposed methodology is explained in Chapter 10 Section 10.3.

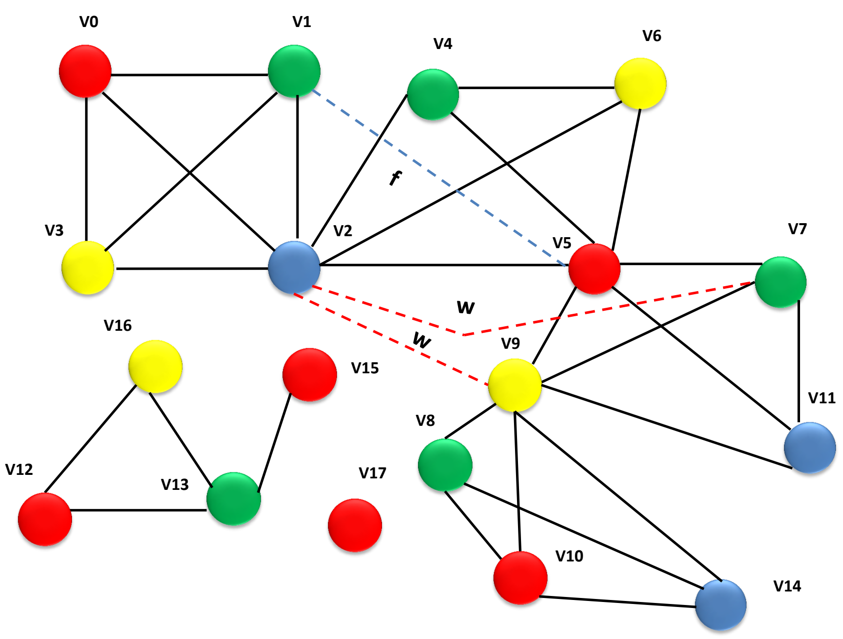


Figure 5.6: Final colored interval graph incorporating buyer fingerprint (additional edges in blue dots) and seller watermark (additional edges in red dotted line)

5.4 Summary

The chapter presents a novel solution to the long pending concern of IP cores protection for both the IP buyer and the IP seller for CE systems. The work proposes a symmetrical IP core protection methodology that implants buyer's fingerprint during scheduling and register allocation phase and seller's watermark during register allocation phase of HLS. Additionally, as the additional constraints are implanted at the higher design abstraction level, therefore it will automatically propagate to the lower abstraction level and protect both the design level. This is the first approach in the literature that provides symmetrical protection for IP cores used in consumer electronics designs. Several novelties integrate the framework of the proposed algorithm: (a) multi-variable encoding scheme for fingerprint (b) multi-variable encoding scheme for watermark (c) novel encoding rule for each encoded variable (d) automation of the complete symmetrical protection process during in-synthesis steps of HLS (no pre or post-synthesis steps are required). Finally, experimental results over the standard applications indicate an average 1% design cost overhead compared to baseline (unprotected) design and <1% design cost (includes 0% area overhead and 1.1% latency overhead) overhead compared to a non-symmetrical approach.

Chapter 6

Multi-Stage Structural Obfuscation to Secure IP Core used in CE Systems

Signature-based IP core protection approaches such as watermarking, fingerprinting, etc. (discussed in chapter 4, 5) are useful for passive attacks such as ownership abuse, IP piracy, IP infringement. However, these defense mechanisms are inapplicable to active attacks such as reverse engineering. RE of an IP core is a process of identifying its design, structure and functionality [45]. Using RE one can identify the device technology (Intel transistors, 2012), extract the gate-level netlist [61], and infer the IP functionality [23]. Therefore, using RE an attacker can identify the functionality of an IP core, he/she can launch IP piracy/attacks, insert malicious Trojan logic into an IP core, etc. Therefore, it is extremely crucial to secure an IP core from RE attack.

Structural obfuscation obscures the functionality of a DSP IP core by transforming its architecture into a non-obvious one while preserving the correct functionality. Thus, enhancing the RE complexity for an adversary. The primary challenge in structural obfuscation is maintaining the exact functionality of the design while transforming it. In summary, structural obfuscation is a process of converting a design D into its equivalent form O(D), such that O(D) is unintelligible to an adversary targeting to recover D from O(D).

This chapter presents a novel low-cost, multi-stage structural obfuscation methodology to enhance the RE complexity for an attacker (shown in Fig.6.1). In the proposed approach, a DSP kernel is obscured by its design through several successive high-level transformation

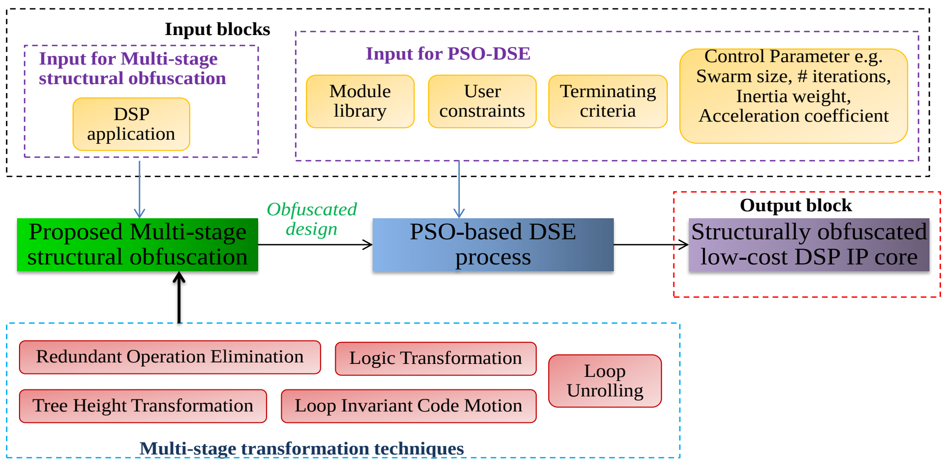

Input blocks
Input for Multi-stage structural obfuscation
DSP application
Input for PSO-DSE
Module library
User constraints
Terminating criteria
Control Parameter e.g. Swarm size, # iterations, Inertia weight, Acceleration coefficient
Proposed Multi-stage structural obfuscation
Obfuscated design
PSO-based DSE process
Output block
Structurally obfuscated low-cost DSP IP core
Redundant Operation Elimination
Logic Transformation
Loop Unrolling
Tree Height Transformation
Loop Invariant Code Motion
Multi-stage transformation techniques

Figure 6.1: Proposed Symmetrical IP core Protection during high level synthesis.

techniques. These transformations increase the complexity for an adversary to discover its original functionality. In the proposed approach, total five high-level transformation approaches are used to obfuscate both loop and non-loop based DSP kernel. They are Redundant Operation Elimination (ROE), Logic Transformation (LT), Tree Height Transformation (THT), Loop Unrolling (LU) and Loop Invariant Code Motion (LICM). Additionally, this approach performs optimization to select a low-cost design architecture that meets design objectives like execution latency, hardware area and all-inclusive design cost. Thus, a low-cost, multi-stage transformation based structural obfuscation with stronger robustness is proposed here. This is the first work in IP core protection of CE systems that provides low-cost, multi-stage structural obfuscation to secure an IP core against RE attack.

The chapter is organized as follows: Section 6.1 discusses the problem formulation, threat model and the target platform of the proposed approach; Section 6.2 explains the proposed methodology; Section 6.3 explains the PSO-DSE framework to explore the low-cost design solution for the obfuscated design; Section 6.4 demonstrates the proposed multi-stage structural obfuscation approach through an example; finally, the summary of this chapter is presented in Section 6.5. (*NOTE: All the abbreviations and taxonomy are listed in 'Acronyms' and 'Nomenclature' section respectively.*)

6.1 Formulation, Threat Model and Evaluation Models

This section presents the problem formulation, threat model and evaluation models of the proposed approach.

6.1.1 Problem formulation

For a given loop-based DSP core in the form of a CDFG, explore the design space to determine an optimal structurally obfuscated design solution. The generated solution should minimize the overall design cost while satisfying conflicting user-given constraints. The problem can be formulated as follows:

Minimize: Obfuscated design cost (A_T^{OBF}, L_T^{OBF}) for optimal $(X_i) = \{N(R_1), N(R_2), ..., N(R_D), UF\}$.

Subject to: $A_T^{OBF} \leq A_{cons}$ and $L_T^{OBF} \leq L_{cons}$ and provide resiliency against RE attack.

6.1.2 Threat model

The proposed work enhances the RE complexity for an adversary during RTL synthesis by obscuring the original structure of an IP core design. Thus, provides defense against threats resulting due to RE such as IP piracy, IP infringement (counterfeit) and Trojan insertion.

6.1.3 Evaluation models

Area model

Total area A_T^{OBF} consumed by a structurally obfuscated design can be expressed as:

$$A_T^{OBF} = \sum_{i=1}^{m} \left(A(R_i) * N(R_i)\right) + A(mux) * N(mux) + A(buffers) * N(buffers) \quad (6.1)$$

Total design area is evaluated by adding total area of hardware resource units (such as adder, subtractor, multiplier etc.), interconnecting units (such as multiplexer and de-multiplexer) and storage units (such as register, delay elements). The total design area is assessed with respect to the module library of 15 nm NanGate-based technology [37].

Latency model

Total execution latency L_T^{OBF} of the obfuscated design can be expressed as follows:

$$L_T^{OBF} = \underbrace{\left(T_{body}^{OBF} * \left[\frac{I}{UF}\right]^{floor}\right)}_{\text{for unrolled loop}} + \underbrace{\left(I \, mod \, UF\right) * T_{first}^{OBF}}_{\text{for sequential loop}}. \tag{6.2}$$

Fitness function

The fitness of each obfuscated design solution is calculated (considering hardware area consumption and execution latency) based on the following function:

$$C_f(X_i) = \phi_1 \frac{A_T^{OBF} - A_{cons}}{A_{max}^{OBF}} + \phi_2 \frac{L_T^{OBF} - L_{cons}}{L_{max}^{OBF}}, \tag{6.3}$$

(*NOTE: During the experiment, both the weightage factor ϕ_1 and ϕ_2 are kept 0.5 to provide equal preference to both the design parameters. However, based on the user's priority the weightage on silicon area and latency can be modified by tuning the value of weightage factors. The value of each weightage factor lie between 0 to 1.*)

6.2 Proposed Methodology

As mentioned earlier, the proposed approach leverages five high-level transformation techniques i.e. ROE, LT, THT, LU, and LICM to obfuscate the input DSP application during algorithmic synthesis. It takes the input of an application in the form of a loop-based CDFG or non-loop based DFG and applies each of the aforementioned HLTs to obfuscate it. The flow diagram of the proposed approach is shown in Fig. 6.2. To obtain higher robustness during obfuscation, all the aforementioned HLT techniques have performed in consecutive stages as mentioned in Fig. 6.2. An example of non-obfuscated original CDFG shown in Fig. 6.3 is used for explanation. The design is scheduled based on 3 adders and 4 multi-

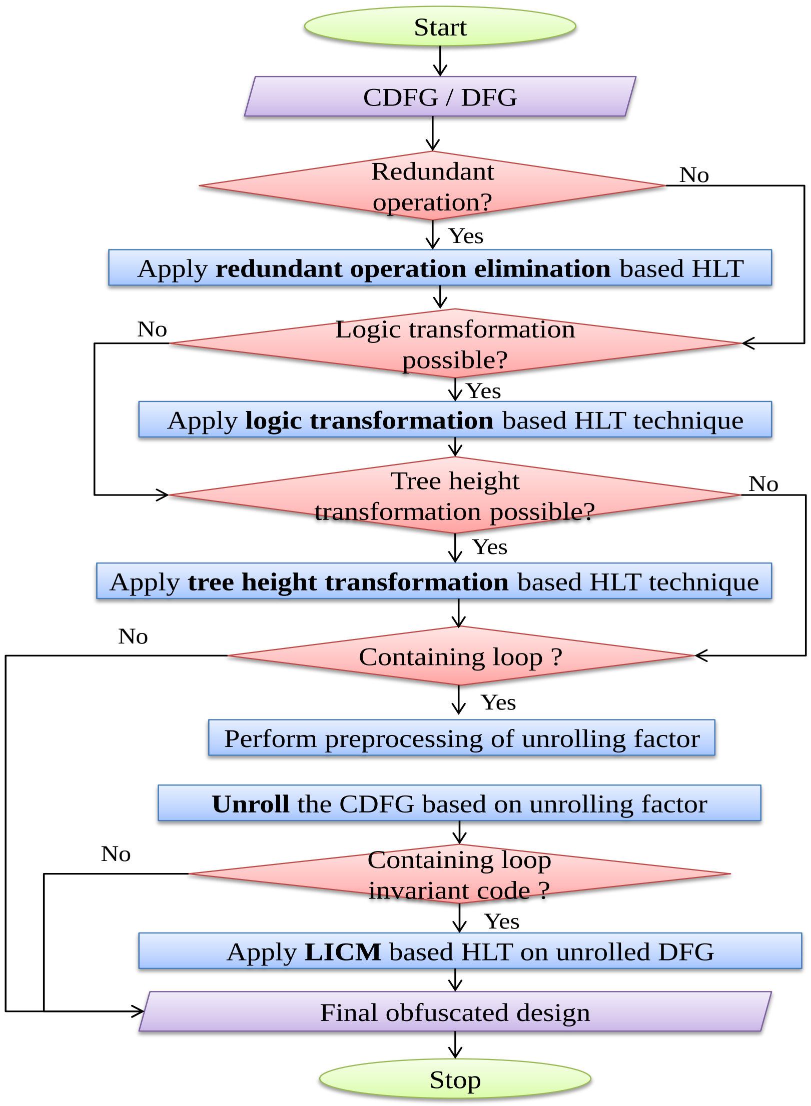

Start
CDFG / DFG
Redundant operation?
No
Yes
Apply **redundant operation elimination** based HLT
Logic transformation possible?
No
Yes
Apply **logic transformation** based HLT technique
Tree height transformation possible?
No
Yes
Apply **tree height transformation** based HLT technique
Containing loop ?
No
Yes
Perform preprocessing of unrolling factor
Unroll the CDFG based on unrolling factor
Containing loop invariant code ?
No
Yes
Apply **LICM** based HLT on unrolled DFG
Final obfuscated design
Stop

Figure 6.2: Flowchart of the proposed multi-stage structural obfuscation approach

pliers which is provided as user input. The scheduled CDFG of the same non-obfuscated design is shown in Fig. 6.4. The corresponding equivalent DSP circuit is shown in Fig. 6.5. As evidence from the design, it employs 6 4:1 switches, 4 2:1 switches, 10 input registers, 1 output register and 8 delay elements. The detailed process of each HLT technique is explained with an example in the following subsections.

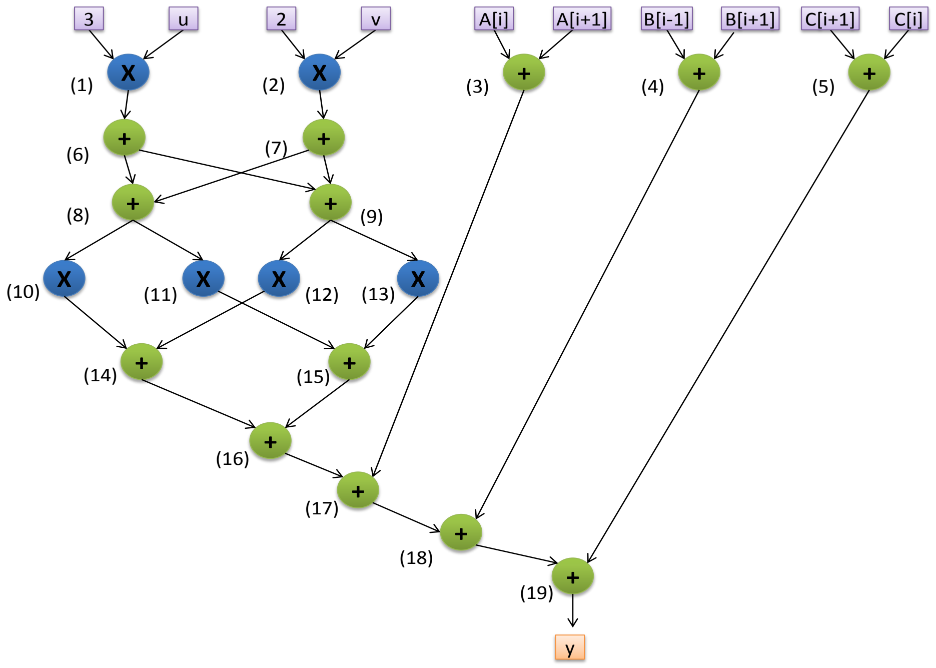


Figure 6.3: Original non-obfuscated loop-based CDFG

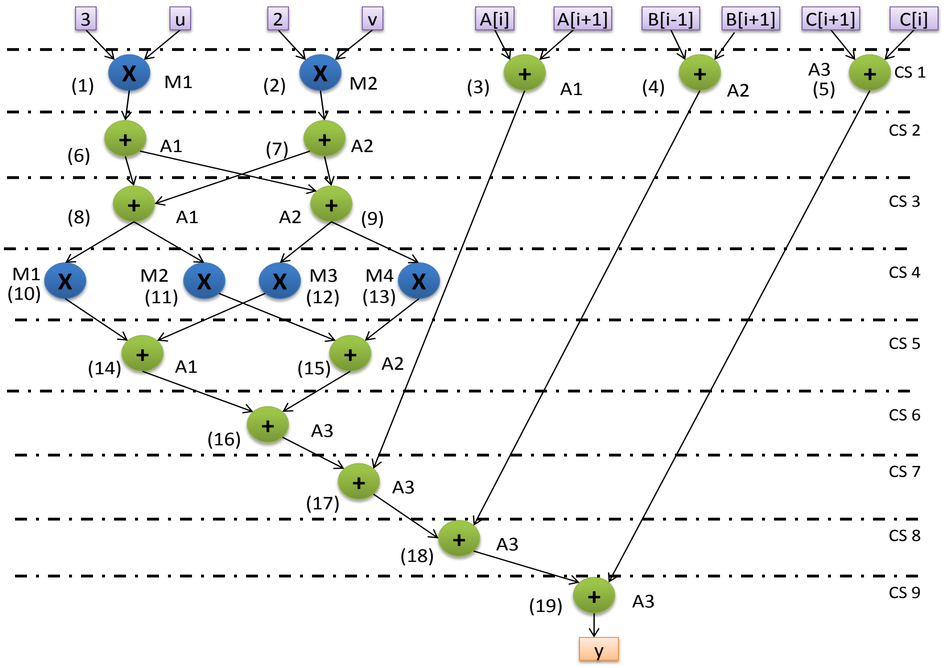


Figure 6.4: Original non-obfuscated CDFG scheduled based on 3(+), and 4(*)

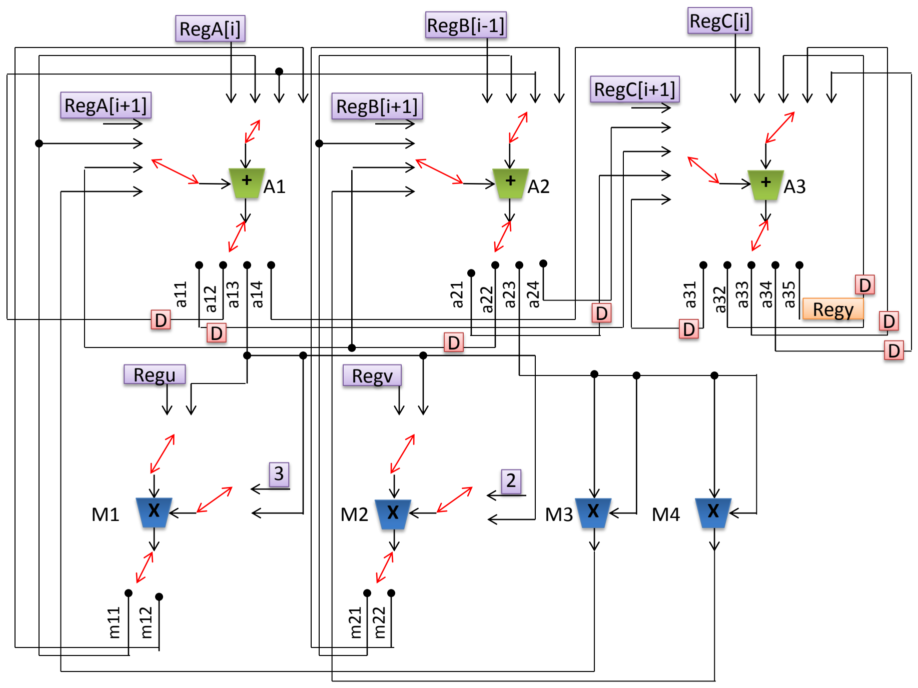


Figure 6.5: Equivalent circuit of non-obfuscated scheduled CDFG

6.2.1 Redundant operation elimination process

It is an HLT technique which is applied to obfuscate a DSP core by removing duplicate nodes from its equivalent graph to form a new functionally equivalent RTL netlist. A node in the input graph is identified as a redundant node if there exists another node which has exactly same parents/inputs and same operation type. If a pair of such nodes is found then the nodes with higher node number are removed from the graph. To preserve the correct functionality necessary adjustments are performed in the graph. The algorithm of the proposed ROE is shown in Fig. 6.6. Finally, ROE based structurally obfuscated graph is produced as output which is fed to the next obfuscation stage. For example, in the non-obfuscated design shown in Fig. 6.3, the redundant operations are operation no. 9, 11, 12, 13, 15 which are removed as shown in Fig. 6.7 to structurally obscure the design. However, to preserve the correctness of the application, the inputs of operation no. 14 and 16 are taken from operation no. 10 and 14 respectively (Fig. 6.7). The eliminated nodes and dependencies are marked with dotted circle and line respectively. However, as both the design is functionally equivalent thereby correctness of the application is preserved.

```
Input – DFG of the application 'A[]' (list of nodes)
Output – ROE based obfuscated DFG 'A[]'
1.      Begin
        // Identifying redundant nodes//
2.      int i , j
3.      For i = 0 to sizeof(A[]) Do
3.1         For j = i+1 to sizeof(A[]) Do
3.1.1           IF(A[i]==A[j]) Then
3.1.2           Assign child of A[j] to child of A[i]
3.1.3           Remove A[j]
3.1.4           End IF
3.2         End For
4.      End For
5.      Perform equivalence checking
6.      End
```

Figure 6.6: Algorithm of the proposed ROE based obfuscation

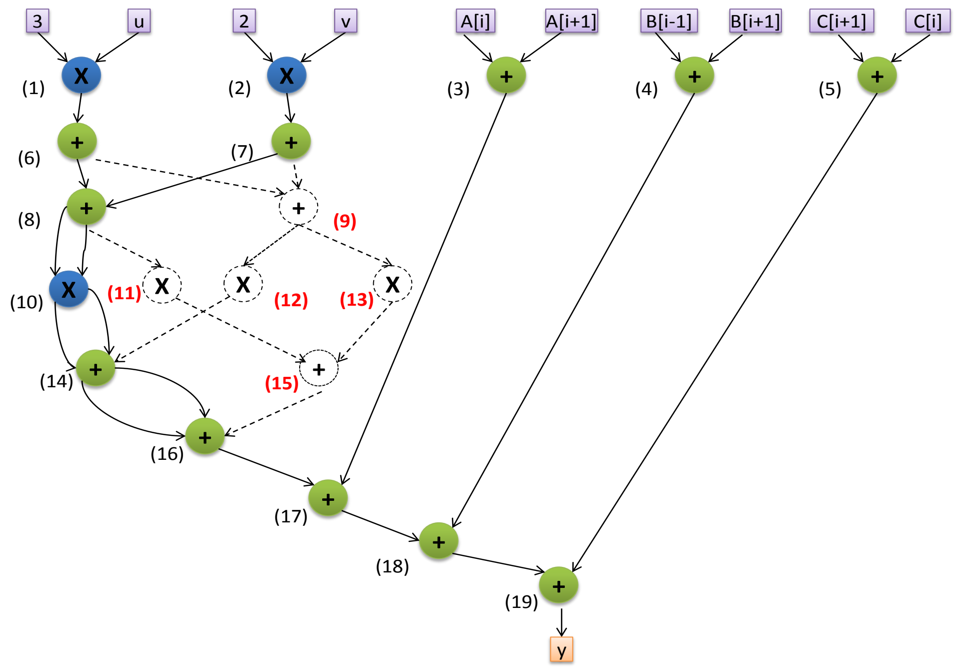


Figure 6.7: Redundant operation elimination based obfuscated CDFG

6.2.2 Logic transformation process

It is another HLT technique which is applied to obfuscate the DSP core by generating a different logically equivalent function to form a new functionally equivalent RTL netlist. In

```
Input – DFG of the application 'A[]' (list of nodes)
Output – LT based obfuscated DFG 'A[]'
1.    Begin
      // Identifying equivalent sub-function//
2.    int a , b, i
3.    For i = 0 to sizeof(A[]) Do
3.1       a = parent1 of node A[i]
3.2       b = parent2 of node A[i]
3.3       Replace f(a,b) with f'(a',b') such that both are functionally
          equivalent
4.    End For
5.    Perform equivalence checking
6.    End
```

Figure 6.8: Algorithm of the proposed LE based obfuscation

the context of DFG/CDFG of an application, LT is applied to obscure the input DFG/CDFG by modifying the graph with different logically equivalent function. It alters the nodes of the input graph such that the graph looks different than the original still satisfies the functionality correctly. The algorithm of the proposed LE is shown in Fig. 6.8. Finally, LT based structurally obfuscated graph is produced as output which is fed to the next obfuscation stage. For example, Fig. 6.9 represents the LT-based obfuscated form of the input graph (Fig. 6.7); newly added/modified nodes are marked with green colored node number and the modified dependencies are marked with the green dotted line. However, as both the design is functionally equivalent thereby correctness of the application is preserved.

6.2.3 Tree height transformation process

It is another HLT technique which is applied to obfuscate the DSP core by modifying the height of its equivalent graph by introducing parallelism in sub-computations to form a new functionally equivalent RTL netlist. It divides the critical path dependency into temporary sub-computations and evaluates in parallel, thereby generates structurally dissimilar yet functionally equivalent graph. The algorithm of the proposed THT is shown in Fig. 6.10.

Finally, THT based structurally obfuscated graph is produced as output which is fed to the next obfuscation stage. For example, Fig. 6.11 represents the THT-based obfuscated form of the input graph (Fig. 6.9). It reduces the height of the obfuscated graph from 10 to 8 compared to the original design. The computation of node 17 and 18 in the obfuscated

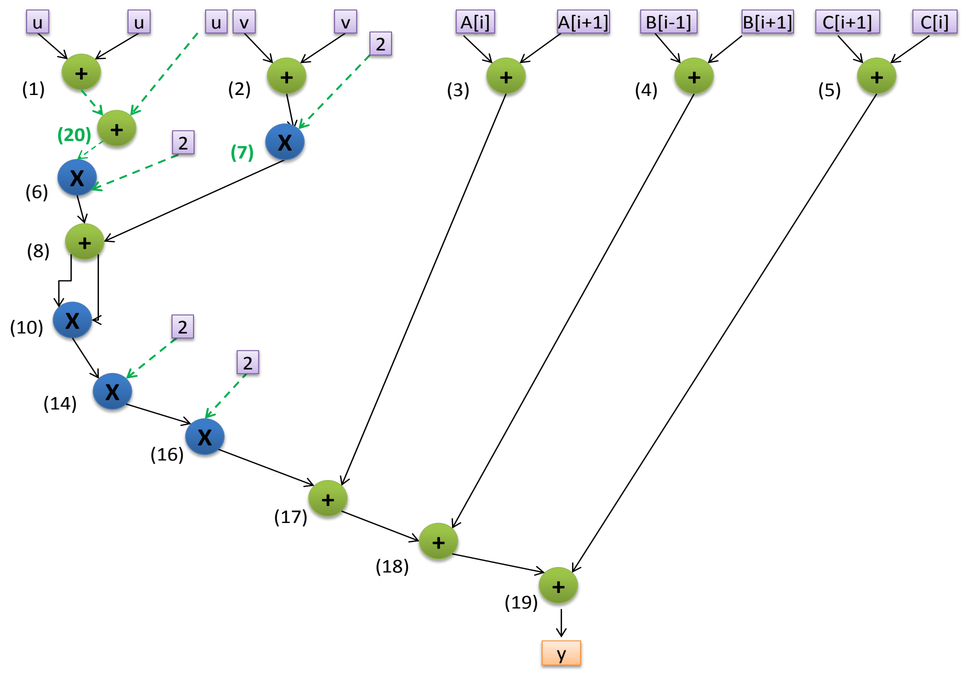


Figure 6.9: Logic transformation based obfuscated CDFG

```
Input – DFG of the application 'A[]' (list of nodes)
Output – THT based obfuscated DFG 'A[]'
1.      Begin
        // Identifying path dependency//
2.      List Path [] = Calculate the list of path of A[]
3.      For int i = 0 to sizeof(Path[]) Do
3.1             IF(Path[i] have sequentially executed associative
                operation) Then
3.2              Divide the path dependency into sub-computations and
                evaluate in parallel
3.3              End IF
4.      End For
5.      Perform equivalence checking
6.      End
```

Figure 6.10: Algorithm of the proposed THT based obfuscation

design is executed earlier compare to the input design. The dependencies of the obfuscated graph are adjusted to maintain the correct functionality. The modified dependencies are marked with red lines. However, as both the design is functionally equivalent thereby correctness of the application is preserved.

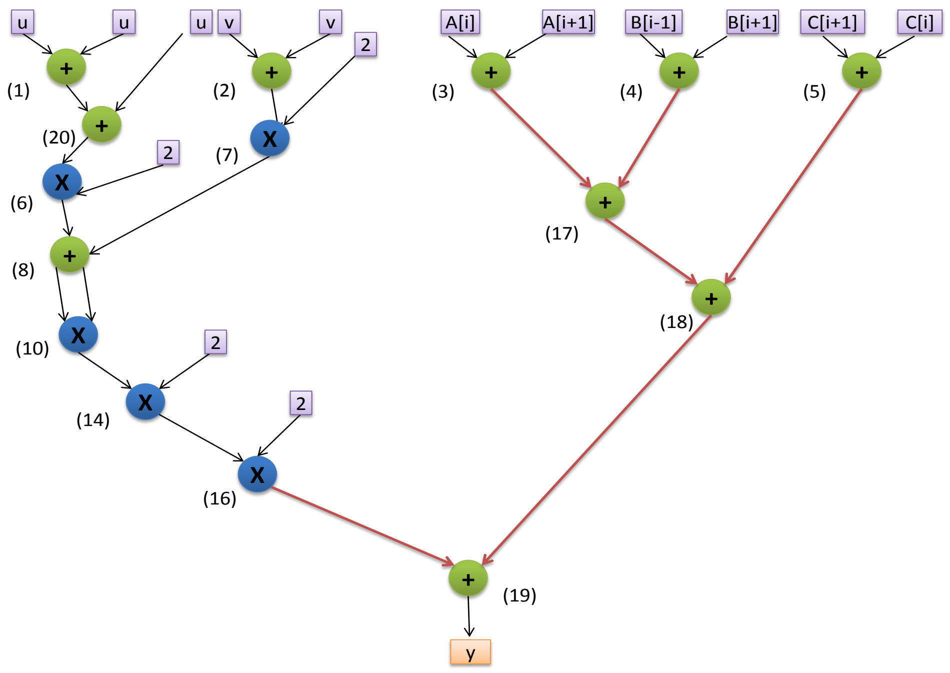


Figure 6.11: Tree height transformation based obfuscated CDFG

```
Input – CDFG of the application 'A[]' (list of nodes) and UF
Output – LU based obfuscated DFG 'A[]'
1.      Begin
        // Identifying loop unrolling factor//
2.      For int i = 0 to UF Do
2.1         Repeat the loop body of A[i]
2.2         Assign the corresponding inputs of the repeat operations
3.      End For
4.      Perform equivalence checking
5.      End
```

Figure 6.12: Algorithm of the proposed LU based obfuscation

6.2.4 Loop unrolling process

It is another HLT technique which is applied to obfuscate the loop-based DSP application by unwinding the loop from its equivalent graph and executes the same calculation multiple times to form a new functionally equivalent RTL netlist. Thus generates structurally dissimilar yet functionally equivalent graph. The algorithm of the proposed LU is shown in Fig. 6.12.

Finally, LU based structurally obfuscated graph is produced as output which is fed to the next obfuscation stage. For example, Fig. 6.13 represents the LU-based obfuscated

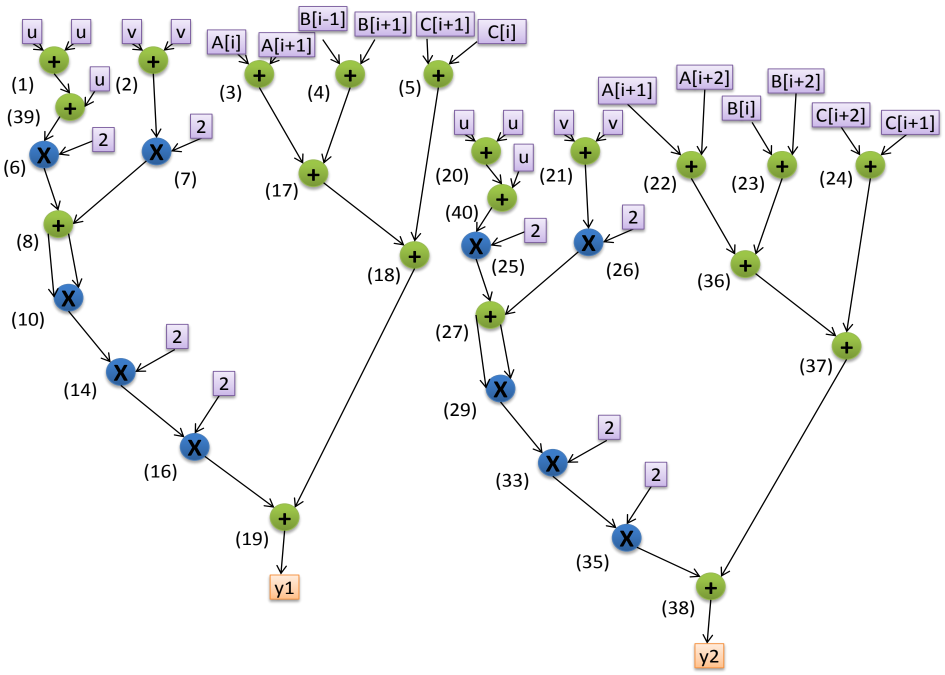
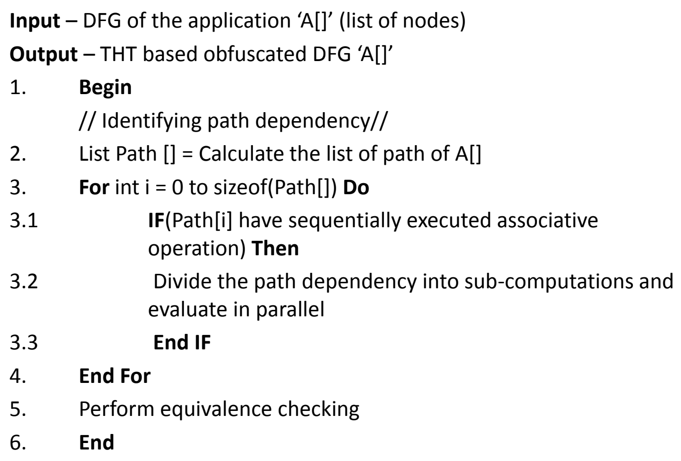


Figure 6.13: Loop unrolling based obfuscated CDFG



Input – DFG of the application 'A[]' (list of nodes)

Output – THT based obfuscated DFG 'A[]'

```
1.    Begin
      // Identifying path dependency//
2.    List Path [] = Calculate the list of path of A[]
3.    For int i = 0 to sizeof(Path[]) Do
3.1       IF(Path[i] have sequentially executed associative
          operation) Then
3.2       Divide the path dependency into sub-computations and
          evaluate in parallel
3.3       End IF
4.    End For
5.    Perform equivalence checking
6.    End
```

Figure 6.14: Algorithm of the proposed LICM based obfuscation

form of the input graph (Fig. 6.11) based on UF = 2. As the execution latency of a design is directly dependent on the UF, therefore, in the proposed approach the optimal UF is explored through PSO driven DSE process. It minimizes the execution time as well as concurrently structurally obscure the IP design. However, as both the design is functionally equivalent thereby correctness of the application is preserved.

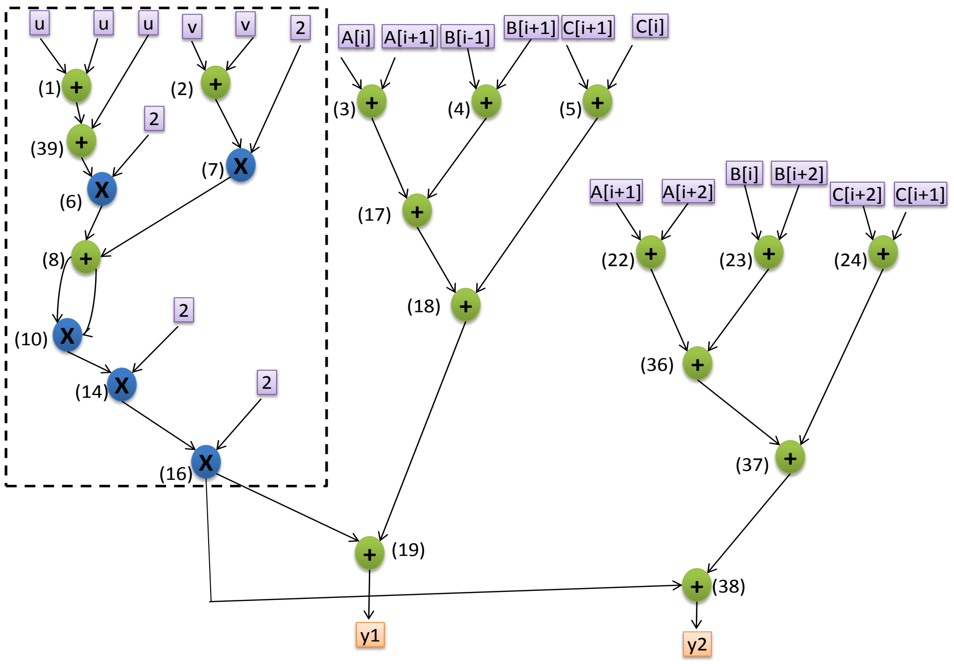


Figure 6.15: Loop invariant code motion based obfuscated CDFG

6.2.5 Loop invariant code motion process

It is another HLT technique which is applied to obfuscate the loop-based DSP application by moving the loop independent nodes out of the loop body from its equivalent graph to form a new functionally equivalent RTL netlist. More specifically, it moves out the nodes of the loop which would not make any differences if it executes inside the loop iteratively or outside the loop once. Thereby, it speeds up the execution process while maintaining the correct functionality of the graph. Thus, generates structurally dissimilar yet functionally equivalent graph. The algorithm of the proposed LICM is shown in Fig. 6.14.

Finally, LICM based structurally obfuscated graph is produced as output. For example, Fig. 6.15 represents the LICM-based obfuscated form of the input graph (Fig. 6.13) based on UF = 2. The dotted box shows the nodes which do not depend on the loop. As evidence from the Fig. 6.15, for UF = 2 loop invariant operations are executed once while loop depended operations are executed two times. However, as both the design is functionally equivalent thereby correctness of the application is preserved.

(NOTE: The failure of the last step of each algorithm i.e. Perform Equivalence checking indicates the corresponding transformation algorithm is unable to preserve the functional-

ity of the input DSP IP core. Therefore, the transformations due to this algorithm will be undone. In other words, the design of the DSP IP core will be roll backed to its previous version (before applying the algorithm).)

6.3 Exploring PSO-driven Low-Cost Structural Obfuscation

In the proposed approach low-cost obfuscated IP design is achieved through particle swarm optimization driven design space exploration. It accepts the obfuscated design (explained in Section 6.2) of the application in the form of DFG/CDFG, module library, terminating criteria of PSO, control parameters (like inertia weight, social factor, cognitive factor etc.) and user given area-delay constraints as inputs and generates a low-cost optimized obfuscated IP design as output. The detailed PSO-DSE process shown in Fig. 6.16, is explained in the following subsections.

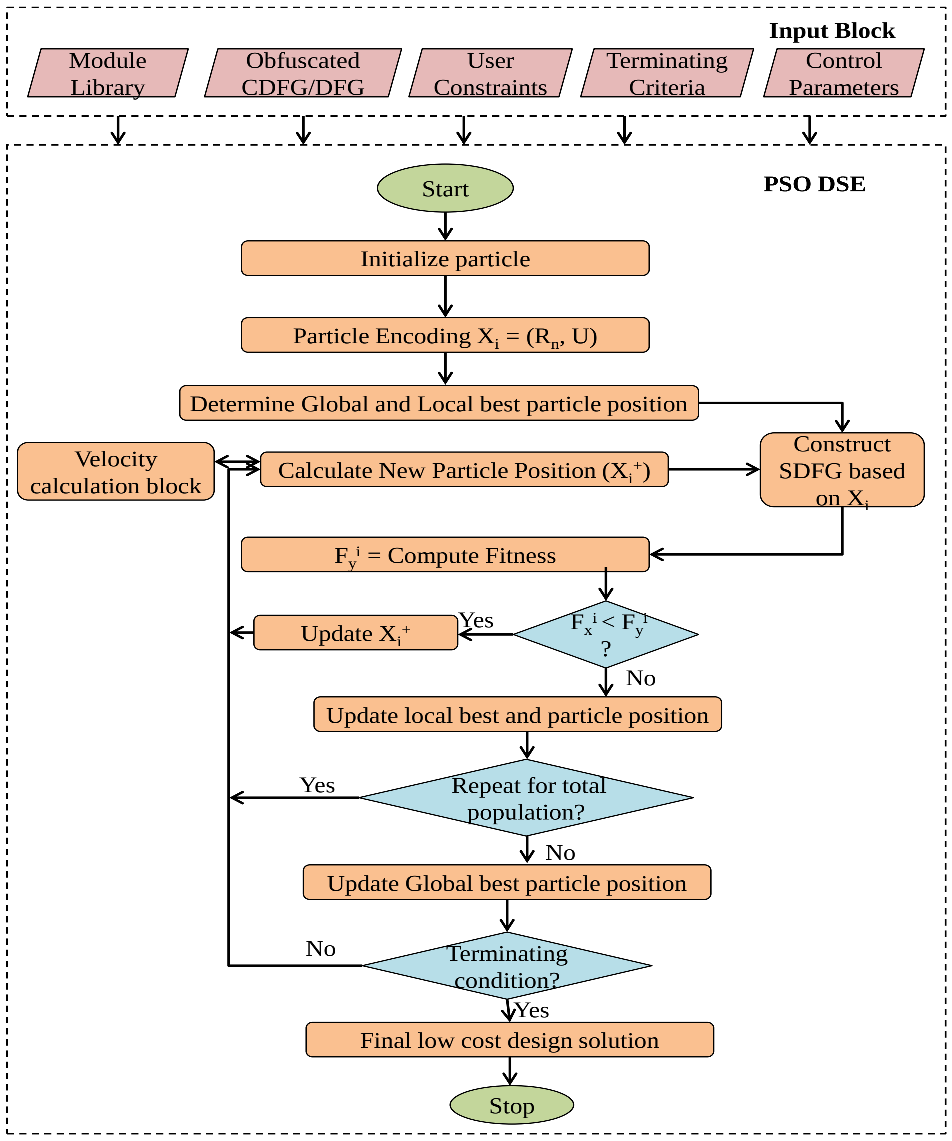
6.3.1 Overview of PSO-DSE

PSO is a population-based stochastic optimization methodology where every single solution is known as a particle. The fitness of each particle is evaluated based on the fitness function to be optimized. The velocity of each particle directs the movement of the particle. The particles move through the search space by following the current global best $gbest$ and its own best location $lbest$. After finding a better $gbest$ or $lbest$ the i^{th} particle updates its velocity and position.

6.3.2 Initialization of particle

In the PSO-DSE process, the particle position (X_i) be expressed as follows:

$$X_i = \{N(R_1), N(R_2), ..N(R_D), UF\} \tag{6.4}$$



Input Block
Module Library
Obfuscated CDFG/DFG
User Constraints
Terminating Criteria
Control Parameters
PSO DSE
Start
Initialize particle
Particle Encoding $X_i = (R_n, U)$
Determine Global and Local best particle position
Construct SDFG based on X_i
Velocity calculation block
Calculate New Particle Position (X_i^+)
F_y^i = Compute Fitness
$F_x^i < F_y^i$?
Yes
Update X_i^+
No
Update local best and particle position
Repeat for total population?
Yes
No
Update Global best particle position
Terminating condition?
No
Yes
Final low cost design solution
Stop

Figure 6.16: PSO driven DSE process for low-cost obfuscated design

The particles are initialized based on uniform distribution over the search space and can be represented as follows [53]:

$$X_1 = \{min(R_1), min(R_2), ..min(R_D), min(UF)\} \tag{6.5}$$

$$X_2 = \{max(R_1), max(R_2), ..max(R_D), max(UF)\} \tag{6.6}$$

$$X_3 = \left\{\frac{(min(R_1)+max(R_1))}{2}, .. \frac{(min(R_D)+max(R_D))}{2}, \right.$$
$$\left. \frac{(min(UF)+max(UF))}{2}\right\} \quad (6.7)$$

Where, the first three particles are initialized as mentioned above and the rest of the particles are initialized using the following equation:

$$X_n = \left\{\frac{(min(R_1)+max(R_1))}{2} \pm \alpha, .. \right.$$
$$\frac{(min(R_D)+max(R_D))}{2} \pm \alpha,$$
$$\left. \frac{(min(UF)+max(UF))}{2} \pm \alpha\right\} \quad (6.8)$$

Where, $min(R_1)$ denotes minimum number of resource of resource type R_1. Similarly, $max(R_1)$ denotes maximum number of resource of resource type R_1, α denotes a random integer between minimum and maximum number of resource in d^{th} dimension of the design space.

6.3.3 Movement of particle using velocity

In the PSO-DSE process, each dimension of a particle velocity is updated based on the following equation:

$$V_{d_i}^{+} = \omega V_{d_i} + b_1 r_1 [R_{d_{lb_i}} - R_{d_i}] + b_2 r_2 [R_{d_{gb}} - R_{d_i}] \quad (6.9)$$

Where, V_{d_i} is the velocity of i^{th} particle in the d^{th} dimension, $V_{d_i}^{+}$ is the updated velocity. ω is the inertia weight, b_1, b_2 are acceleration coefficients. r_1 , r_2 are random numbers. $R_{d_{lb_i}}$ denotes local best of i^{th} particle in the d^{th} dimension. Similarly, $R_{d_{gb}}$ denotes global best of the d^{th} dimension. R_{d_i} is the number of resources of i^{th} particle in d^{th} dimension. The updated velocity is subsequently utilized to calculate new position of the particle swarm in the design space using the following equation:

$$R_{d_i}^{+} = R_{d_i} + V_{d_i}^{+} \quad (6.10)$$

Where, R_{d_i} and $R_{d_i}^{+}$ represents the current and updated particle positions.

6.3.4 Terminating criteria of PSO

The PSO based DSE process will terminate if any of two condition arises, a) reached the maximum number of iteration (I_{max}), b) no improvement is observed in global best for δ number of iteration. In the proposed approach the value of I_{max} and δ is taken as 100 and 10 respectively.

6.4 Motivational Example

This section gives a demonstration of the proposed multi-stage structural obfuscation mechanism through an example. For the ease of explanation of the same non-obfuscated CDFG shown in Fig. 6.3 is taken as input. In the shown graph adders and multipliers are denoted through green and blue nodes respectively. The integer value on the side of each node designates the respective operation number. As shown in Fig. 6.3, all the primary inputs are shown in the purple boxes, where, three arrays i.e. A, B and C with two constant values i.e., u, and v are available. The arrays are loop dependent variables and the constant values are loop independent variables. The final output is stored in register 'y', shown in an orange box. The total number of node in the graph is 19 before applying any transformation technique, it consists of total 8 iterations and the height of the graph is 9. After applying multiple successive transformation techniques on the input CDFG, the final multi-stage structurally obfuscated CDFG is shown in Fig. 6.17.

Due to ROE, node number 9, 11, 12, 13, 15 is removed; due to LT, node number 6, 7, 14, 16, and 39 is altered; due to THT, the height of the graph is decreased from 9 to 8; due to LU for UF = 2, the CDFG is unrolled two times; and finally due to LICM, the loop invariant nodes are kept outside the loop. Therefore, the total number of nodes present in the graph is increased from 19 to 21 in the final obfuscated CDFG.

This multi-stage structurally obfuscated CDFG is further fed as input with the module library and PSO control parameters to explore a low-cost design solution through PSO-DSE framework (refer to Fig. 6.16). For the particular application, it is observed that the optimal

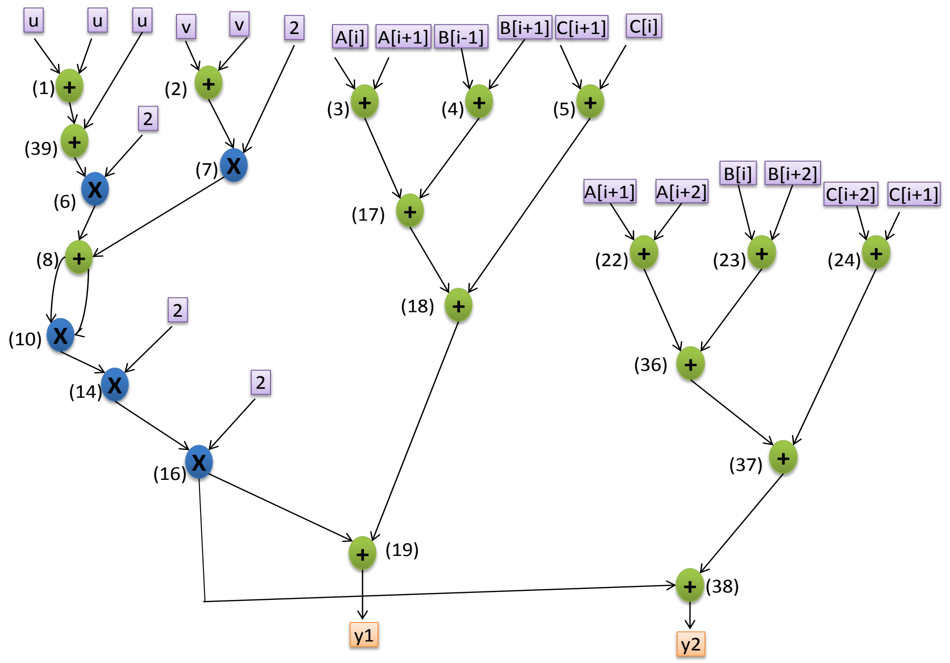


Figure 6.17: Obfuscated CDFG after employing multi-stage HLTs

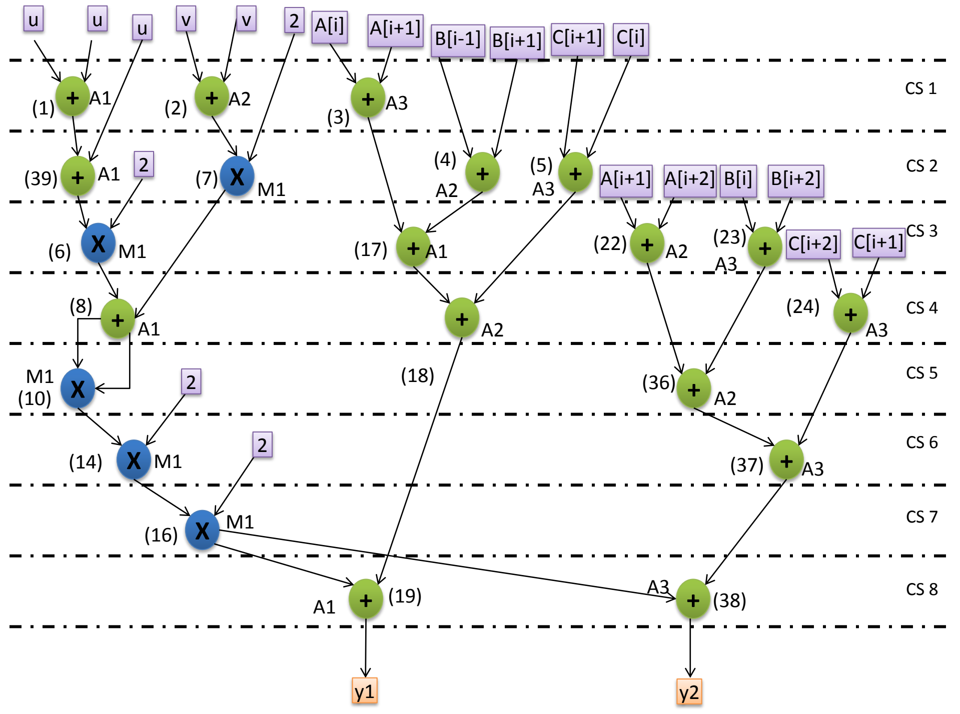


Figure 6.18: Low-cost obfuscated IP design scheduled with 3(+) and 1(*)

design solution is three adders and one multiplier. Therefore, the multi-stage structurally obfuscated CDFG is scheduled based on aforementioned resources. Fig. 6.18 represents

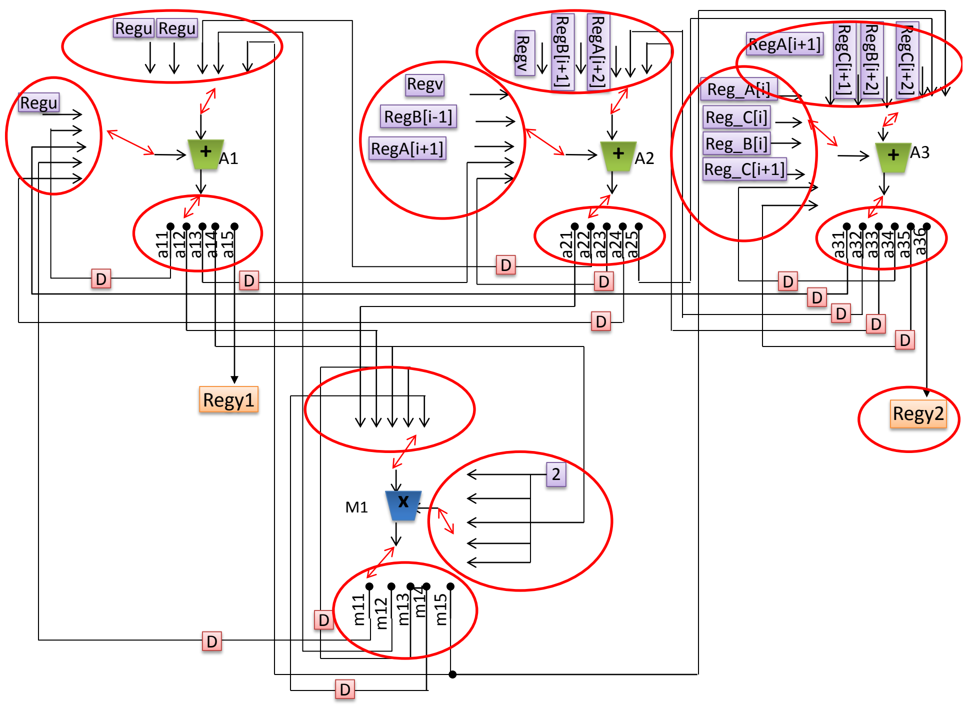


Figure 6.19: Equivalent circuit of multi-stage structurally obfuscated IP design

the final low-cost, structurally obscured scheduled CDFG of the input application. The equivalent obfuscated circuit of the application is shown in Fig. 6.19. It employs 8 8:1 muxes, 18 input registers, 2 output register and 13 delay element. The changes due to proposed multi-stage structural obfuscation for this application (marked as red circle in Fig. 6.19) at register-transfer/module level and gate level are reported in Table 10.18.

Note: The corresponding experimental results of the proposed methodology is explained in Chapter 10 Section 10.4.

6.5 Summary

The chapter presented a novel solution to the hardware security of RE on DSP based IP cores. The work presented a solution for protecting a DSP IP core against RE attacks using multi-stage structural obfuscation. The work proposes several DSP circuit obfuscation techniques by leveraging high-level transformations. This is the first approach in the literature that provides low-cost, multi-stage structural obfuscation to enhance the complexity of

RE attack for IP cores used in consumer electronics designs. Several novelties integrate the framework of the proposed approach: (a) multiple HLT-based obfuscation methodology applied successively to hinder RE (b) achieves low-cost obfuscated design through PSO-DSE framework (c) capable of handling both loop and non-loop based applications (d) automate the complete symmetrical protection process during in-synthesis steps of HLS (no pre or post-synthesis steps are required). Finally, experimental results of the proposed approach yielded an enhancement on obfuscation of 22 % and reduction in obfuscated design cost of 55 % compared to a single-stage structural obfuscation approach.

Chapter 7

SAT and Removal Attack Resilient Functional Obfuscation to Secure IP Core of CE Systems

In the previous chapter, a novel structural obfuscation approach is proposed to thwart RE attack for DSP IP cores used in CE systems. Hardware obfuscation can also be achieved through functional obfuscation. Functional obfuscation (also known as logic encryption) inserts additional logic components known as key-components into the design to lock its functionality and implementation. These additional logic components accept key bits as input and based on these key value it produces the output. Therefore, applying the correct key combination will produce the correct result while applying the wrong keys led to exhibit an incorrect functionality of the design. Thus functional obfuscation thwarts RE attack. However, functional obfuscation is prone to other attacks like sensitization attack, removal attack, SAT attack, IP piracy and Trojan attack.

The functional obfuscation approach for DSP IP core available in the literature is unable to handle all the aforementioned attacks [25]. More specifically, this approach is susceptible to removal and SAT attack. This chapter presents a novel removal and SAT attack resilient block, which is integrated with [25] to enhance the robustness of the existing functional obfuscation approach (shown in Fig.7.1). As shown in the figure, the green block represents the functionally obfuscated circuit which accepts three inputs viz. primary

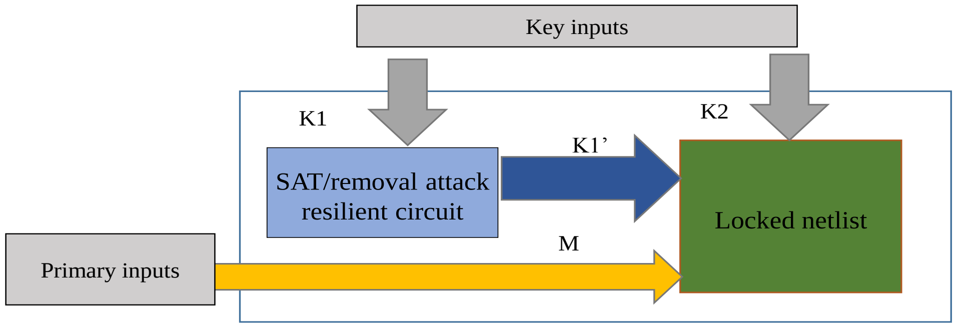


Figure 7.1: Overview of the proposed approach

inputs (M), encrypted key (K1') and direct non-encrypted key (K2). The blue block represents the SAT/removal attack resilient circuit which accepts K1 as input and generates K1' as encrypted output.

The chapter is organized as follows: Section 7.1 discusses the possible attacks on functional obfuscation and the threat model of the proposed approach; Section 7.2 explains the proposed approach; and the summary of this chapter is presented in Section 7.3. (*NOTE: All the abbreviations and their corresponding full form are listed in 'Acronyms' section.*)

7.1 Possible Attacks and Threat Model

This section discusses the functional obfuscation approach for DSP IP core available in the literature [25], the attacks which can be as well as cannot be mitigated through this approach and thus derives the threat model of the proposed approach.

7.1.1 Possible attacks

Functional obfuscation generates a locked netlist by inserting key components or key-gates in the circuit. However, incorrect placement of key components may invite several attacks. This sub-section discusses possible attacks scenarios of a locked netlist.

Isolated key-gates based key-sensitization attack

A key-gate in a locked netlist is considered as isolated if there is no path between that key-gate and any other key-gates. An attacker can easily sensitize the key value of an

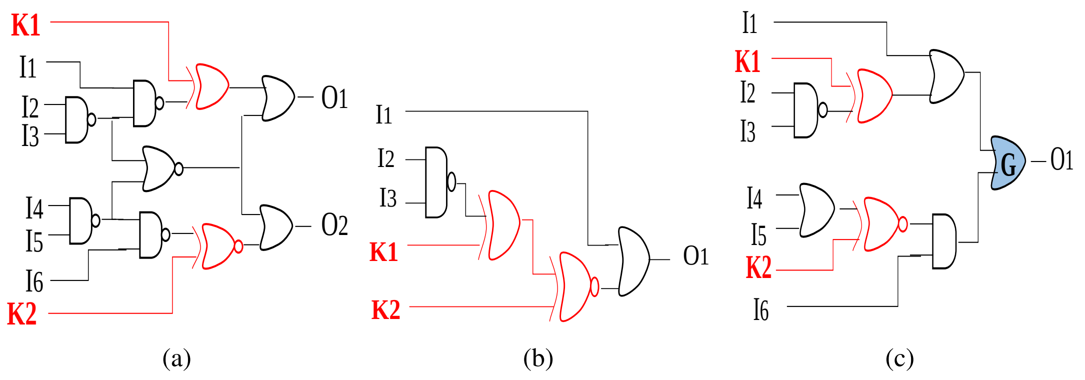


Figure 7.2: Corresponding to the sensitization attack: (a) Isolated key gates K1 and K2, (b) Run of key gates K1 and K2, (c) Concurrently mutable key gates K1 and K2

isolated key-gate. For example in Fig. 7.2(a), there is no path between key-gate K1 and K2. Therefore, K1 and K2 are isolated key-gates. An attacker can sensitize the value of K1 as 0 to the output 1 (O1) by applying '100XXX' i/p pattern.

Run of key-gates based key-sensitization attack

A set of key-gates in a locked netlist is considered as run of key-gates if they are connected in a back-to-back fashion. Few run of key-gate structures are vulnerable for locked netlist as it reduces the effort of an attacker to identify the correct key by increasing the possible correct key combinations. The reason is, some run of key-gate structure may be replaceable with a single key-gate, thus, diminishing security strength. For example in Fig. 7.2(b), K1 and K2 are connected back-to-back, thus forming a run of key-gates. In this can both '01' and '10' are the correct key value. Moreover, K1 and K2 can be replaced by a single key-gate (XOR) with '1' as the correct key value.

Mutable key-gates based key-sensitization attack

If two or more key-gates converges but have no common path between them, sensitization may still occur. Therefore, if key-gates K1 and K2 converge to another gate so that a specific key-bit can be obtained by muting the other, then both key-gates are called concurrently mutable key-gates. In this type of scenario, an attacker can easily trace the key value K1 by muting K2. For example in Fig. 7.2(c), K1 and K2 converge at the gate G. The value of K1 can be sensitized to the O1 by muting the effect of K2. Applying I6=0 in the input pattern will mute K2. Similarly, The value of K2 can be sensitized to the O1 by muting the effect of K1. Applying I1=1 in the input pattern will mute K1.

IP piracy and Trojan attack

If somehow the correct key combination of a locked netlist is identified by an attacker, he/she can easily discover the correct functionality of that netlist. This may enhance the possibility of IP piracy/counterfeit and Trojan insertion attacks. An attacker can either resell/redistribute the IP core illegally to a make profit or can insert a Trojan logic at safe places to introduce malfunctioning of an IP core.

SAT attack

SAT attack model is as follows:

Objective: Identify the correct key value of the locked netlist.

Requirements: (i) The locked netlist, (ii) an activated functional IC bought from the open market.

Process: The process of launching SAT attack is as follows:

1. The SAT attack algorithm formulates a CNF of the circuit.
2. The SAT solver generates distinguishing input patterns (DIP).
3. Feed the DIP to the activated functional IC and observed the correct output.
4. This distinguishing input/output pair is used on the locked netlist to eliminate the wrong keys.
5. Continue this process until no wrong key is found.

Removal attack

Removal attack model is as follows:

Objective: Remove all the key component from the locked netlist.

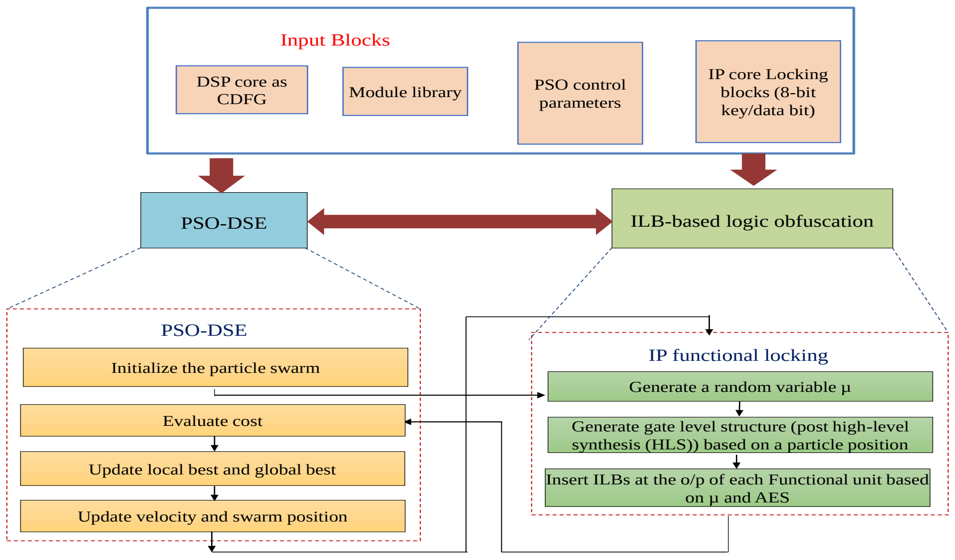

Input Blocks
DSP core as CDFG
Module library
PSO control parameters
IP core Locking blocks (8-bit key/data bit)
PSO-DSE
ILB-based logic obfuscation
PSO-DSE
Initialize the particle swarm
Evaluate cost
Update local best and global best
Update velocity and swarm position
IP functional locking
Generate a random variable µ
Generate gate level structure (post high-level synthesis (HLS)) based on a particle position
Insert ILBs at the o/p of each Functional unit based on µ and AES

Figure 7.3: ILB based functional obfuscation approach

Requirements: (i) Template of the ILBs.

Process: The process of launching SAT attack is as follows:

1. Based on the available template search the ILBs.
2. Remove the ILBs from the locked netlist.
3. Generate an unlocked circuit.

7.1.2 ILB-based functional obfuscation

This section explains the process of generating the locked netlist (shown in Fig. 7.1) through function obfuscation. Authors in [25] presented a low-cost functional obfuscation methodology for DSP core, which consists of two major phases, i.e. (i) designing functionally locked DSP core, (ii) exploring low-cost design configuration through the PSO-DSE process (shown in Fig. 7.3). Both these phases perform iteratively until a globally optimal design configuration is explored.

Based on this design configuration, a gate-level datapath structure (post-processing through standard HLS framework) of the input CDFG application is generated, which is then locked using ILB. This generates a functionally obfuscated design of the input DSP

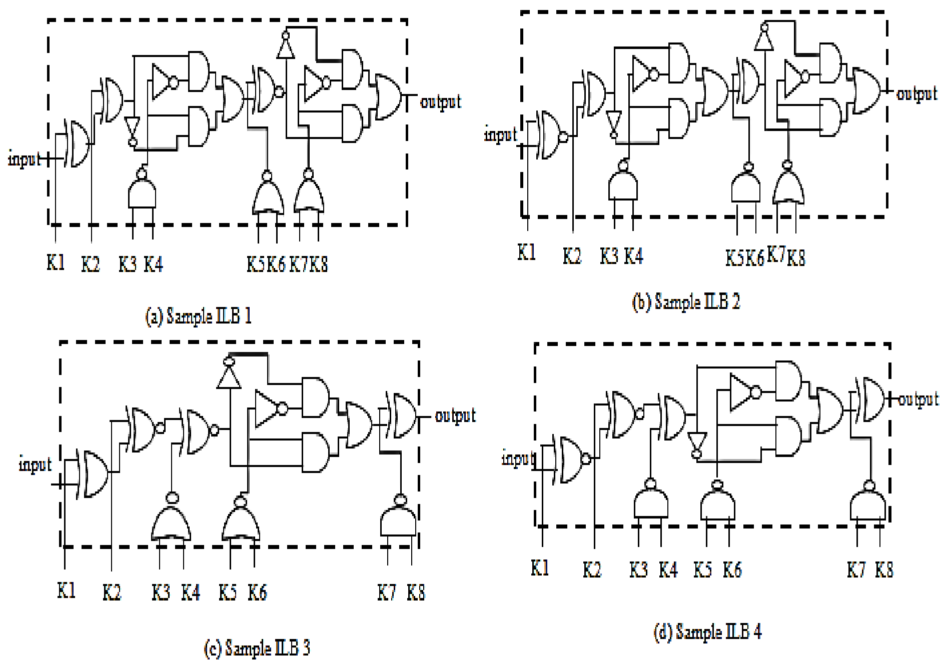


Figure 7.4: Sample ILB structures

core. Then the total power and delay of the locked design are calculated to evaluate the design cost through a fitness function. Subsequently, particle positions are updated to explore a new fitter particle, and the complete process is repeated until the termination condition is met. Thus the final particle position after meeting the terminating condition is used to generate a low-cost obfuscated design of a DSP core. Therefore, the final design architecture not only optimizes the total power, delay and implementation cost but also minimizes the total gate count of the complete locked netlist, which includes the total number of functional units, registers/latches, muxes/demuxes, and ILBs.

Overview of ILBs

ILBs are key components inserted into a non-obfuscated design to generate a functionally obfuscated design. Each ILB consist of 8-bit key value inserted into each bit of output data. ILBs are designed using the different combination of AND, NAND, NOT, XOR and XNOR gates. However, structures of ILB depend on the key values. Innumerable different structures of ILBs with the same area can be possible. Each of these ILBs generates correct output if a correct key value is provided. Different ILB structures are shown in Fig 7.4.

Features of ILBs

The ILBs are inserted in each output data bit of the gate structure of DSP IP core to generate a locked netlist. These ILB structures are designed mainly to avoid key-sensitization attacks, IP piracy and Trojan attack. Therefore, multiple features are considered while designing the ILBs to avoid any possible scenario of the aforementioned attacks.

Multi-pairwise security: In [25], all the ILBs have 8 key-bits and all are multi-pairwise secure. In other words, to sensitize any key-bit of an ILB to the output, an attacker has to know or control the correct value of other 7 key-bits. Therefore, to identify the correct value of other 7 key-bits, an attacker has to apply brute-force attack.

Prohibiting key-gate isolation: In [25], different ILBs structures (shown in Fig. 7.4) consists of 8 intertwined dependent on each other key-gates. This ensures no isolated key-gate is present in the ILBs. Thus, prohibiting isolated key-gate from the locked netlist.

Protection against run of key-gates: In [25], the key-gates of ILBs are interlaced among each other, therefore, replacing all key-gates (8 input key-gates) with one single key-gate is highly difficult.

Non-mutable convergent key-gates: In [25], all the ILBs consist of 8 non-mutable key-bits, therefore, to sensitize one key-bit to the output remaining 7 key-bits need to be muted, which is a complicated process.

Example: generating a locked netlist through ILBs

A gate level structure of FIR benchmark, designed using 1 adder and 1 multiplier is functionally obfuscated through ILBs. The complete gate structure of the functionally obfuscated design is shown in Fig. 7.5 . While inserting the ILBs, ILB1 (shown in Fig. 7.2(a)) is inserted for the first two output data bit of adder, then ILB2 (shown in Fig. 7.2(b)) is inserted for the next two output data bit and so on until all the output data bit of adder and multiplier are locked through ILBs. The ILB based functional obfuscation approach [25] is vulnerable to removal and SAT attack discussed earlier. The proposed approach presents a novel removal and SAT attack resilient block, which is integrated with [25] to enhance its robustness. The threat model of the proposed approach is presented below.

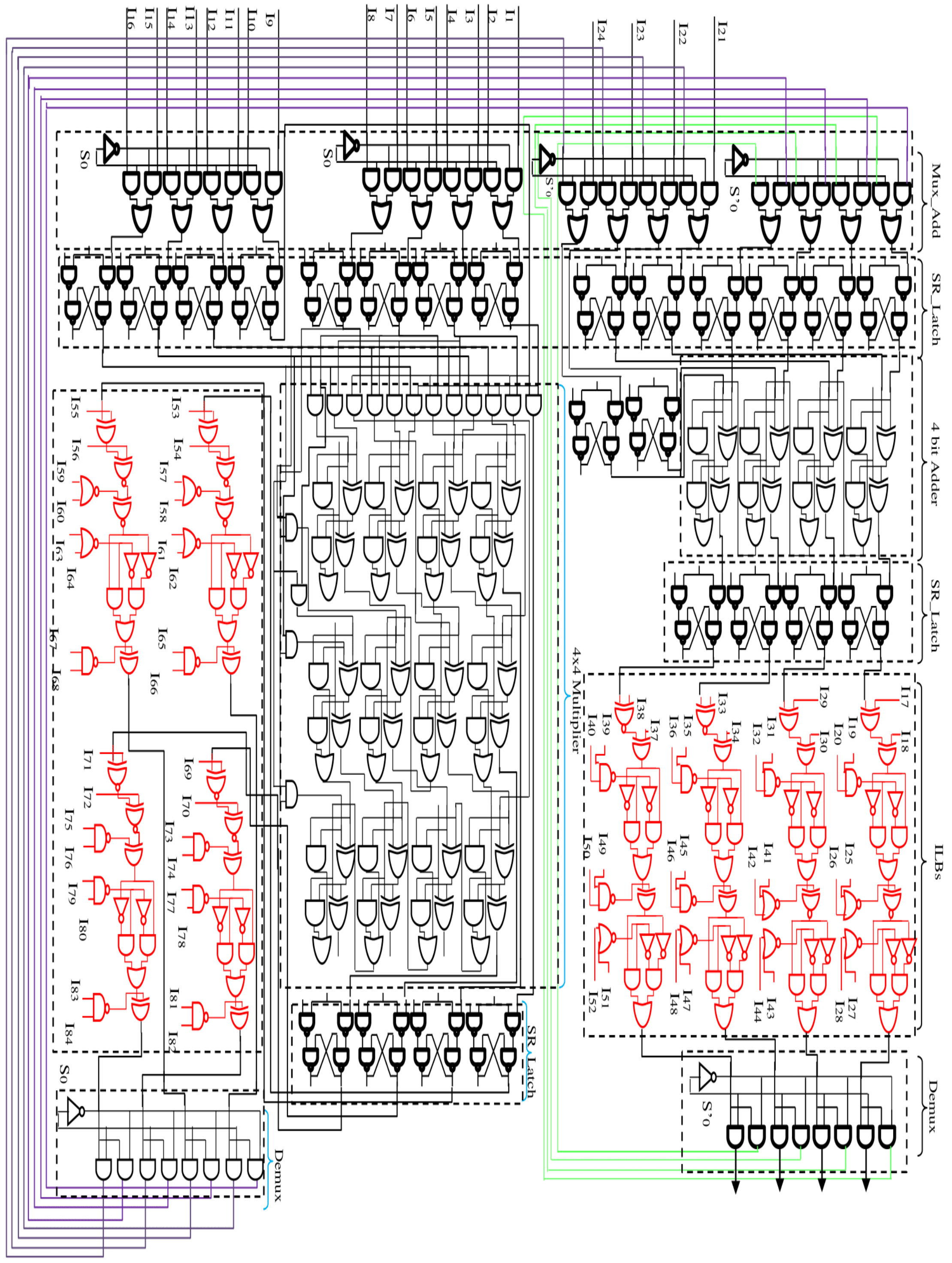


Figure 7.5: ILB based functionally obfuscated FIR benchmark

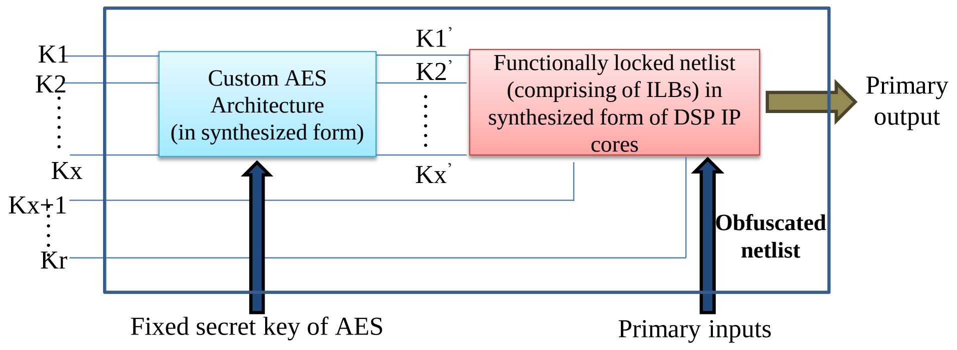

K1
K2
Kx
Kx+1
Kr
Custom AES Architecture (in synthesized form)
K1'
K2'
Kx'
Functionally locked netlist (comprising of ILBs) in synthesized form of DSP IP cores
Primary output
Obfuscated netlist
Fixed secret key of AES
Primary inputs

Figure 7.6: Countermeasure of SAT-attack based on custom AES module

7.1.3 Threat model

The proposed approach aims to enhance the resiliency of ILB-based functional obfuscation approach by mitigating SAT and removal attack. Thus, enhances the complexity of reverse engineering of an IP netlist structure, for an attacker present anywhere in the design flow. In this threat model, it is assumed that an adversary may present in an untrusted foundry.

7.2 Proposed Methodology

The proposed approach integrates a custom light-weight AES IP module with the ILB-based locked netlist of a DSP IP core to thwart SAT and removal attack (shown in Fig. 7.6). To achieve this a custom lightweight AES module with a fixed secret key is designed, which is not publicly available. This module encrypt a subset (K1,..Kx) of the key-bits and generates encrypted key-bits (K1',..Kx'). The encrypted key-bits along with the rest of the key-bits (Kx+1,..Kr) and primary inputs are fed into the functionally obfuscated DSP IP core. This section discusses the custom made lightweight AES module and how it is used to thwart SAT and removal attack.

7.2.1 Designing a light-weight AES module

AES algorithm has three different variations of key sizes i.e., 128 bits or 192 bits or 256 bits. The input data size is 128 bits and the encrypted output data is also 128 bits. The input data is called the plaintext and the encrypted output data is called ciphertext. The

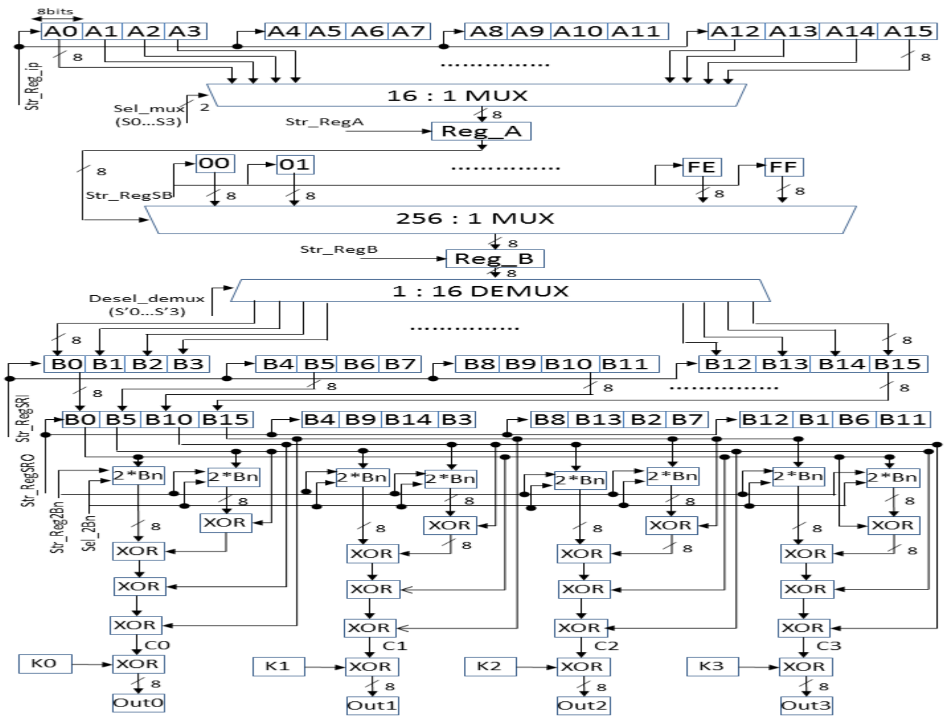


Figure 7.7: Block diagram of the designed lightweight custom AES module

designed AES module has a key size of 128 bits. The core structure of the AES module comprises of the following components: (i) byte substitution layer (ii) diffusion layer (iii) key addition layer (iv) key generator. Fig. 7.7 shows the designed AES module, where, A0 to A15 indicates 128 bit input data, B0 to B15 indicates 128 bit data after performing substitution, C0 to C15 indicates data after performing diffusion, K0 to K15 indicates 128 bits secret key of round 0 and OUT0 to OUT15 indicates 128 bits output after 1st round. Fig. 7.8 shows the design structure of the key generator, where, $K^1 0$ to $K^1 15$ indicates 128 bits key of round 1. This custom AES module is implemented using Altera Quartus 13.0 version in Cyclone II FPGA [14]. It is observed that it is comprising of 429 logic elements only which is $<$1% of the available resources of that device.

7.2.2 Mitigating SAT attack

The SAT solver available in the literature [58] is not scalable for multiplier circuit. Because, for a small size 3-bit multiplier, the equivalent CNF has 20 clauses which is difficult to solve through an SAT solver. Generally, a DSP core comprises several large bit size multipliers

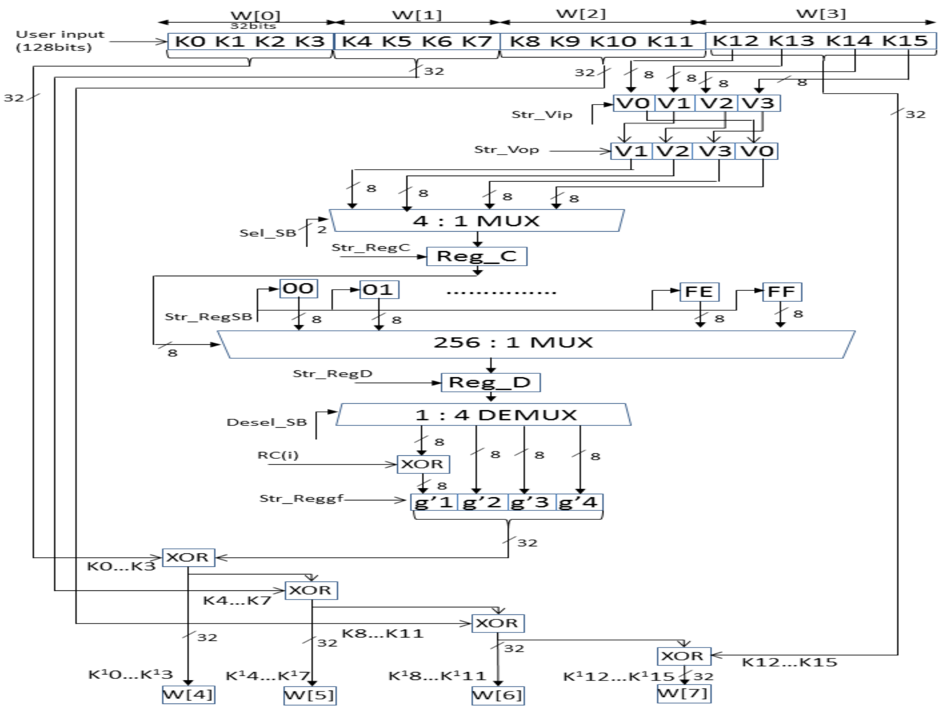


Figure 7.8: Block diagram of the key generator of the AES module

as an integral part. Therefore, DSP IP core designs are SAT attack resistant by default. However, the proposed approach provides a proactive countermeasure (assuming efficient SAT attacks may be formulated for DSP cores in future). In the proposed approach the subset of the key-bits are encrypted through the designed custom lightweight AES module with a fixed secret key. Based on the encrypted key-bits, subset of ILBs are designed and inserted in the obfuscated netlist. This prevents an attacker from determining the correct key value as he/she do not have the key of AES block (shown in Fig. 7.6). This AES module utilizes $<$1% of final design logic elements while implementing using Altera Quartus 13.0 version in Cyclone II FPGA [14] after integration with the obfuscated netlist of the 4-bit FIR. The implemented gate structure of the custom AES module integrated obfuscated 4-bit FIR in Altera Quartus is shown in Fig. 7.9. Post-synthesis result for FIR DSP core after integrating the custom AES block is shown in Table 7.1. It is computationally infeasible to determine the inputs of AES from its output when the key is unknown. To prevent SAT attack, the following steps are performed:

1. Designer synthesizes an AES block with a fixed secret key.

Table 7.1: Resource usage of AES module integrated FIR IP core

Resource type	Total used	Total available	Used %
Logic elements	548	68416	<1%
Combinational function	548	68416	<1%
Dedicated logic registers	32	68416	<1%

2. Subsets of keys of functional obfuscation are encrypted through the AES block.

3. The outputs of the AES block (encrypted keys) are connected to few ILBs in the locked netlist depending on the number of keys are encrypted through AES.

4. Based on the fixed secret key of the AES block and the input to AES, a designer can identify the output of AES and configures the ILBs accordingly by re-organizing the internal gate structure of ILB.

Therefore, the total area of ILB and the design cost remain almost similar. For example, based on an encrypted output of key bits (first 8-bits viz. 11001101) through the AES module, a reconfigured ILB can be designed as shown in Fig. 7.10. As it is a custom AES block, therefore its structure is unknown to an outsider. Therefore, distinguishing and removing AES block from the post-synthesized functionally locked gate design is not feasible. However, it is important to note that more the number of key inputs of functional obfuscation fed into the AES block, more is the number of available exhaustive options for reconfigurable ILBs. Further, the attack time grows exponentially thus enhances the difficulty for an attacker in determining the correct key value.

7.2.3 Mitigating removal attack attack

Removal of ILBs and AES from the locked netlist is difficult because:

1. The ILBs along with AES module inserted into the design for locking are re-synthesized. This synthesis makes each component of the AES circuit along with the ILB-based locked netlist indistinguishable, which is not feasible to remove by an attacker.

2. As the subset of ILB structures inserted in the netlist are reconfigured by the designer based on the AES encrypted output corresponding to the secret key, therefore, no

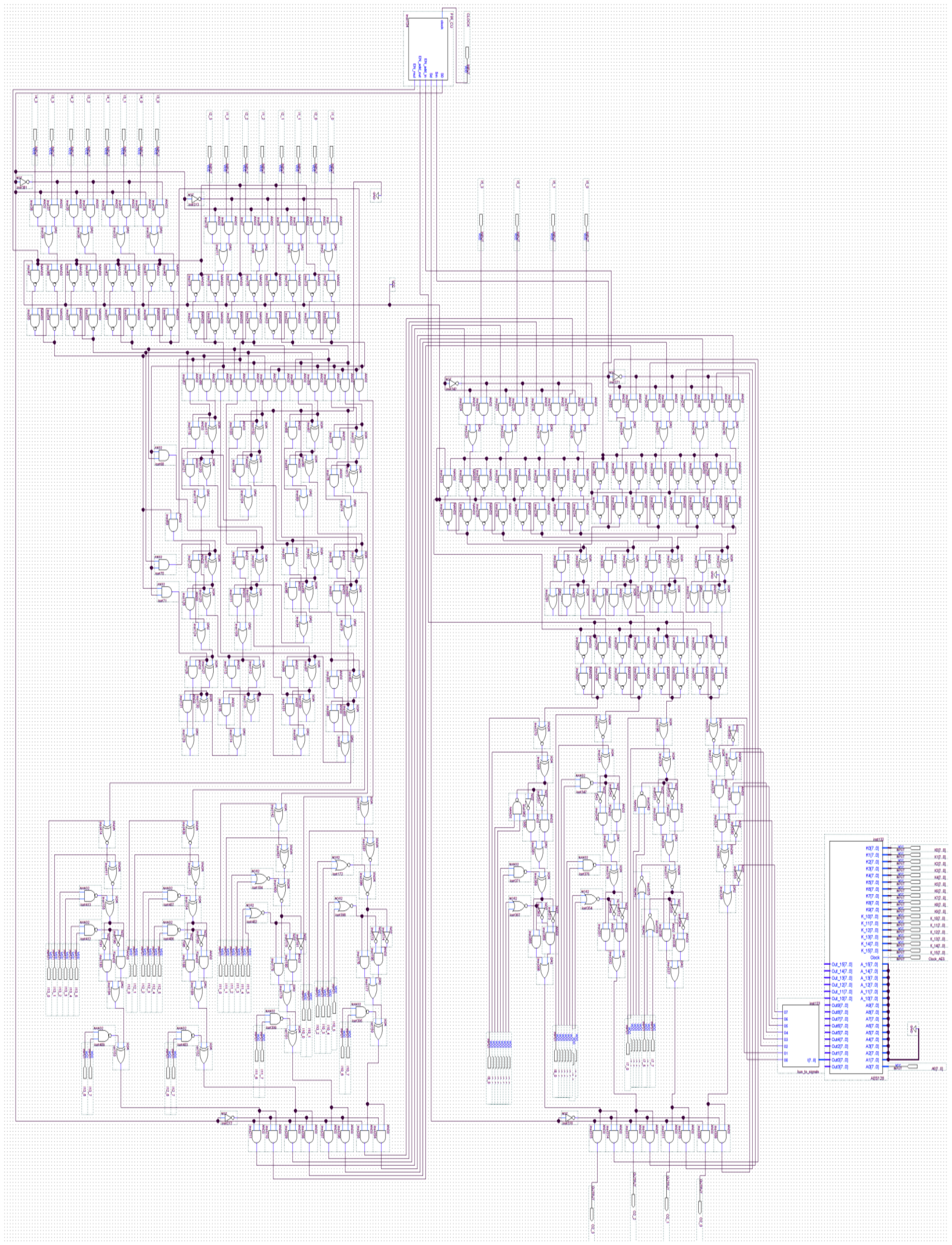


Figure 7.9: AES module integrated functionally locked FIR benchmark

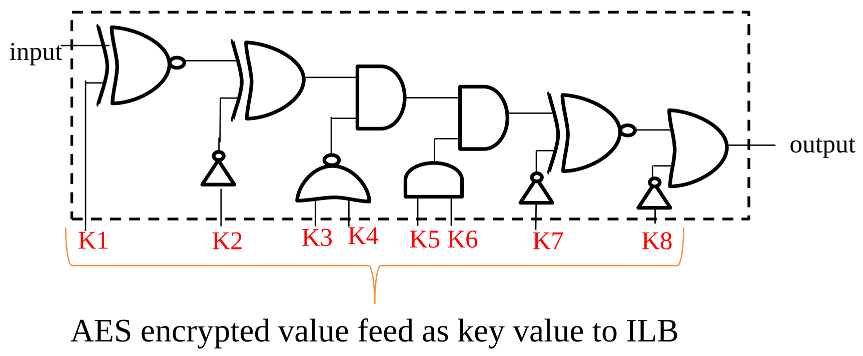


Figure 7.10: Example of a reconfigured ILB based on a sample encrypted output (11001101.) from the AES circuit

fixed template of ILBs are available. The configuration of ILB depends on AES output which in turn depends on the secret key and input of the AES, resulting in several possible structures. Moreover, the number of possibilities also increases with the number of key-inputs fed to ILBs through AES. Therefore, knowing the structures of all the possible ILBs for an attacker is not practical in this scenario.

3. Since a custom lightweight AES architecture is designed and it is not publicly available thus, the components of AES structure after post-synthesis become indistinguishable from the locked netlist of DSP core.

7.3 Summary

The chapter presented a novel solution to the SAT attack and removal attack to enhance the resiliency of functional obfuscation approach. Though SAT for DSP IP core is not feasible till now, however, the proposed approach provides a proactive countermeasure. This is the first approach in the literature that provides resiliency against SAT and removal attack during functional obfuscation for DSP IP cores used in consumer electronics designs. Several novelties integrate the framework of the proposed approach: (a) a custom lightweight AES module is proposed and designed in a standard CAD tool (b) the custom light-weight AES module is integrated with a locked circuit of a standard DSP core to mitigate SAT and removal attack.

Chapter 8

Obfuscation to Secure Multimedia Processor IP Core of CE Systems

Nowadays, most of the CE systems handle image compression and decompression process either through general purpose processor or through a dedicated IP core. The reason is, modern CE devices such as smartphone, tablets, scanner, laptop, smartwatch etc. use high-quality camera lenses, recording and displaying components to capture and display a digital image in high-resolution. Therefore, the image size increases, which becomes a critical issue from data storing and transmission perspective. Reducing the size of an image while storing and/or transferring it, is one of the popular and commercially successful techniques to address this issue.

DCT-based JPEG image CODEC is one of the effectively used technique to perform image compression and decompression. DCT segregates an image into multiple 8x8 blocks based on the visual quality of the image and then convert each block to the frequency domain from the spatial domain. It discards small high-frequency components; therefore, it is a lossy image compression in nature. As JPEG CODEC is hugely data-intensive/ power intensive computational process, therefore, many CE products use it as a dedicated IP core in the SoC. Use of JPEG CODEC in the form of an IP core not only balances the trade-off between hardware area and processing speed but also enhances the design productivity. However, a successful RE attack on this JPEG CODEC may result into several threats like IP piracy, IP infringement, malicious Trojan insertion etc. which necessitate its protection.

Standard JPEG CODEC design process does not aim to secure it from RE attack. Successful RE attack enables an adversary (in the foundry) to identify the functionality of the design to counterfeit the netlist of the design and make several copies of it without the knowledge of the IP vendor/owner. Further, if an adversary is able to identify the functionality of the design from its structure, he/she can make a malicious modification (i.e. insert hardware Trojan) into the design and introduce malfunctioning not only in the JPEG IP core but also the complete SoC design.

This chapter presents a novel low-cost, structurally obfuscated dedicated JPEG CODEC IP core design the thwart RE attack. The proposed approach can enhance the reverse engineering complexity for an adversary by obscuring the original structure of the JPEG CODEC. Thus, thwarting the aforementioned threats. In other words, a JPEG CODEC is considered secure, if its architecture is not obvious i.e. its functionality is not easily discoverable by inspecting its structure. This is the first work for CE systems that provides low-cost, secured JPEG CODEC IP core to thwart RE attack.

The chapter is organized as follows: Section 8.1 discusses the overview of the proposed approach; Section 8.2 explains the proposed methodology; Section 8.3 explains the design process of proposed obfuscated JPEG CODEC IP core; finally, the summary of this chapter is present in Section 8.4. (*NOTE: All the abbreviations and their corresponding full form are listed in 'Acronyms' section.*)

8.1 Overview of the Proposed Approach

This section discusses the threat model and problem formulation of the proposed work, an overview of the proposed methodology for generation low-cost, obfuscated JPEG CODEC IP core and its steps and the DSE framework used to obtain low-cost design solution.

8.1.1 Threat model and problem formulation

This section presents the threat model and the problem formulation of the proposed approach.

Threat model

The proposed work obscures the structure of a standard JPEG CODEC through structural obfuscation. It transforms the architecture in such a way that the functionality becomes unknown to an adversary. Thus hinders to launch reverse engineer to discover the actual functionality by an adversary.

Problem formulation

For a given non-obfuscated JPEG kernel, design a structurally obfuscated, low-cost IP core to hinder RE attack. The design cost of the obfuscated JPEG IP core is evaluated through two crucial orthogonal design parameters, i.e., hardware area and execution latency. It can be expressed as follows:

$$C_f(X_i) = w_1 \frac{A_T^{JPEG}}{A_{max}^{JPEG}} + w_2 \frac{L_T^{JPEG}}{L_{max}^{JPEG}} \tag{8.1}$$

(*NOTE: During experiment both w_1 and w_2 are kept 0.5 to provide equal preference to both the design parameters.*)

8.1.2 Proposed obfuscation and its steps

The proposed approach structurally obfuscates architecture of a JPEG CODEC IP core design to obscure it's functionality from an adversary. The proposed approach obfuscates the standard JPEG IP core design containing micro IPs as well as the overall macro IP through THT. THT is a compiler driven optimization that is useful for obfuscating an original DSP core design by transforming the height of the equivalent graph of the application. It divides the critical path dependency into temporary sub-computations and evaluates in parallel, thereby generating functionally equivalent yet structurally dissimilar graph elements. The steps to generate proposed obfuscated JPEG CODEC IP core is as follows:

1. Represent the DCT-based JPEG process through a mathematical function

2. Perform expansion of the formulated mathematical function to represent as a mathematical expression.

3. Construct the DFG/CDFG corresponding to the mathematical expression.
4. Represent each sub-expression using micro IP and the whole expression as a macro IP.
5. Apply THT based structural obfuscation on each micro IP as well as the macro IP of the corresponding DFG/CDFG.
6. Explore the low-cost design configuration for the obfuscated DFG/CDFG.
7. Feed the low-cost obfuscated DFG/CDFG into an HLS engine.
8. Finally, obtain a low-cost structurally obfuscated JPEG CODEC IP core as output.

8.1.3 Proposed DSE framework for low-cost obfuscated JPEG CODEC IP core

The proposed approach integrates the obfuscation methodology with a PSO driven DSE optimization framework to obtain low-cost design solution.

PSO process

PSO is a population-based heuristic optimization that searches for an optimal solution iteratively. Each solution of the search-space is encoded as a particle and the fitness of each particle is evaluated based on the fitness function. The velocity of each particle directs the movement of the particle. The particles move through the search-space by following the current global best $gbest$ and its own best location $lbest$. After finding a better $gbest$ or $lbest$ the i^{th} particle updates its velocity and position thus move towards the best solutions. More details are available in Chapter 6.

Benefits of PSO

1. Ability to escape local minima and converge on global optima in most case.
2. Ability to introduce stochasticity into the exploration process.

3. Preserves exploration-exploitation balance during searching low-cost solution.

For obtaining low-cost obfuscated IP core, obfuscated JPEG DFG, module library, PSO control parameters and PSO terminating criteria are provided as inputs to the PSO-DSE block.

8.2 Proposed Methodology of Obfuscated JPEG CODEC IP Core

This section discusses the overview of standard JPEG process, overview of the proposed method for JPEG compression and decompression, the process of generating non-obfuscated DFG of JPEG compression and finally explain the process of constructing JPEG CODEC IP core.

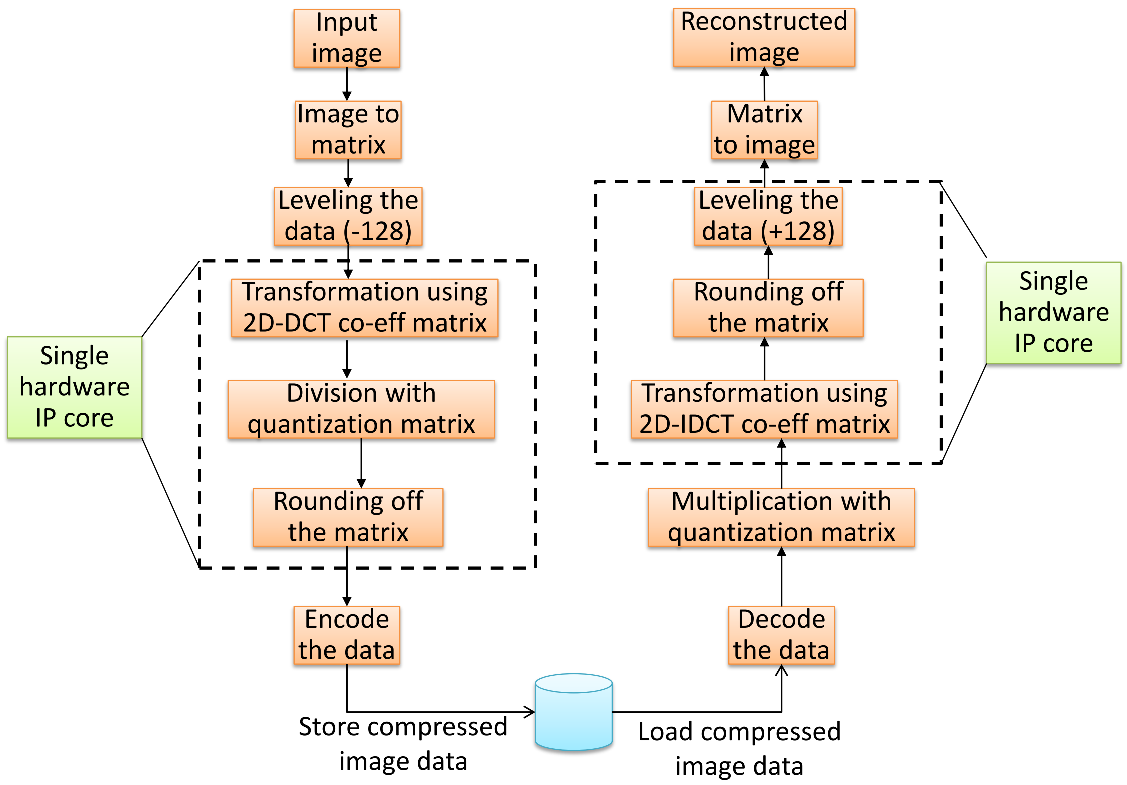


Figure 8.1: Generic overview of JPEG process

8.2.1 Overview of JPEG process

The overview of the JPEG process is shown in Fig. 8.1. In grayscale JPEG image compression/decompression process, a pre-processed NxN image is taken as input and convert it into an NxN matrix. Each integer value of the matrix represents the pixel intensity of a particular pair of co-ordinate (x,y) of the image. For 8-bit depth grayscale image, the maximum pixel intensity is 255 while the minimum is 0, where 0 indicates pure black and 255 indicates pure white. The leveling of the input matrix is performed by subtracting 128 from each pixel value (as DCT can handle pixel value within the range of -128 to 127). In the next step, the input matrix is subdivided into multiple non-overlapping 8x8 blocks because 2D-DCT can process an 8x8 block at a time. The generic 2D-DCT coefficient matrix T is shown below:

$$T = \begin{bmatrix} c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 \\ c_1 & c_3 & c_5 & c_7 & -c_7 & -c_5 & -c_3 & -c_1 \\ c_2 & c_6 & -c_6 & -c_2 & -c_2 & -c_6 & -c_6 & c_2 \\ c_3 & -c_7 & -c_1 & -c_5 & c_5 & c_1 & c_7 & -c_3 \\ c_4 & -c_4 & -c_4 & c_4 & c_4 & -c_4 & -c_4 & c_4 \\ c_5 & -c_1 & c_7 & c_3 & -c_3 & -c_7 & c_1 & -c_5 \\ c_6 & -c_2 & c_2 & -c_6 & -c_6 & c_2 & -c_2 & c_6 \\ c_7 & -c_5 & c_3 & -c_1 & c_1 & -c_3 & c_5 & -c_7 \end{bmatrix}$$

As ‘T’ is an orthogonal matrix, therefore, the inverse of ‘T’ (i.e. T^{-1}) is equal to transpose of ‘T’ (i.e. T^{trans}).

A generic pixel intensity of an 8x8 input image matrix ’M’ is shown below in the form if m_{ij}, where ’i’ and ’j’ represent the row and column number respectively of the pixel intensity of the 8x8 input image block.

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

The DCT on the ‘M’ block can be performed through following matrix multiplications:

$$X = D * T^{trans} \tag{8.2}$$

where, ‘D’ is calculated as follows:

$$D = T * M \tag{8.3}$$

The ‘D’ matrix ($d_{11}, d_{12}, ..d_{88}$) transforms the rows of the ‘M’ block. Further, ‘X’ matrix ($X_{11}, X_{12}, ...X_{88}$) transforms the columns of the ‘D’ matrix. Thus, transforms both rows and columns of the ‘M’ block. Therefore, ‘X’ is the DCT block of ‘M’ block. Each transformed pixel value is divided with the corresponding element of the standard quantization matrix fed as input to generate quantized pixel intensity. Zigzag scanning is performed on this output data to convert it into a 1-dimension array and then run-length encoding is applied to generate the bitstream data of the compressed image for finally storing it in a storage device.

To decompress the image pixel intensities from the stored data, the stored bit stream representing compressed pixel data is first decoded through run-length decoding and then through inverse zigzag scanning, its equivalent 2D image pixel intensity matrix is reconstructed. To perform JPEG image decompression, inverse quantization is applied on the

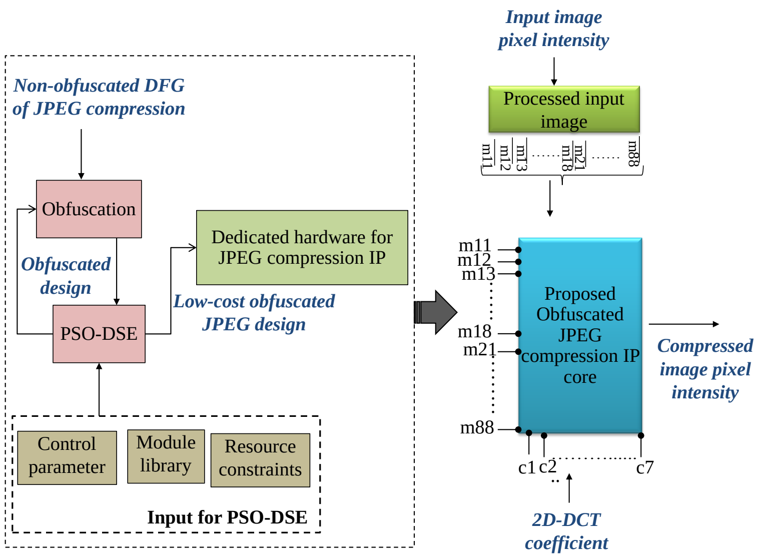

Input image pixel intensity
Processed input image
m11 m12 m13 m18 m21 m88
Non-obfuscated DFG of JPEG compression
Obfuscation
Obfuscated design
PSO-DSE
Dedicated hardware for JPEG compression IP
Low-cost obfuscated JPEG design
Control parameter
Module library
Resource constraints
Input for PSO-DSE
m11
m12
m13
m18
m21
m88
Proposed Obfuscated JPEG compression IP core
Compressed image pixel intensity
c1 c2 c7
2D-DCT coefficient

Figure 8.2: Obfuscated JPEG compression IP core

compressed image pixel block by multiplying each element of the block with the corresponding element of the quantization matrix (Q) to obtain de-quantized image pixel intensities. Next inverse DCT is applied on the de-quantized compressed image block for decompression. Similar to applying DCT on the input block in the JPEG compression process, inverse DCT is applied on the decompressed block (X”). The IDCT on the X” block can be performed through following matrix multiplications:

$$O = E * T \tag{8.4}$$

where, ‘O’ is calculated as follows:

$$E = T^t rans * X'' \tag{8.5}$$

The ‘E’ matrix $(e_{11}, e_{12}, ..e_{88})$ transforms the rows of the X” block. Further, ‘O’ matrix $(o_{11}, o_{12}, ...o_{88})$ transforms the columns of the ‘E’ matrix. Thus, transforms both rows and columns of the X” block. Therefore, ‘O’ is the IDCT block of X” block. After applying IDCT on the decompressed block, each element is rounded to the nearest integer and de-levelized by adding 128.

8.2.2 Overview of proposed methodology for compression

Using the proposed obfuscation steps discuss in section 8.1.2 and DSE engine process discuss in section 8.1.3, a low-cost, obfuscated JPEG compression IP core is designed. The design process includes multiple steps (as shown in Fig. 8.2). Initially, an unsecured (non-obfuscated) JPEG-DCT application in the form of a DFG is accepted as an input. Thereafter, resiliency against RE attack is provided in the form of structural obfuscation. This obfuscated DFG is processed through an optimization framework to obtain a low-cost hardware configuration (detail explained earlier). Thus, this low-cost hardware configuration is used to design an obfuscated dedicated hardware for JPEG compression IP core. The proposed obfuscated JPEG compression IP core uses levelized pixel intensity as input to generate the compressed image pixel intensity as output. Finally, to generate the compressed pixel intensities through the proposed IP core, 2D-DCT coefficients and standard quantization matrix are also fed as inputs.

8.2.3 Generating non-obfuscated DFG of JPEG compression

As mentioned in Section 8.2.1 Eqn. 8.2 and Eqn. 8.3 represent the function of applying 2D-DCT on the input image block M. To convert relationship into a hardware function for dedicated IP core design, this function is expanded and modeled as a mathematical expression to design an equivalent DFG. Further, this DFG is feed to an HLS engine to obtain a JPEG compression IP core. Based on Eqn. 8.2, the corresponding mathematical expression for the first pixel (X_{11}) of the transformed block is modeled as:

$$X_{11} = (c_4 * d_{11} + c_4 * d_{12} + c_4 * d_{13} + c_4 * d_{14} + c_4 * d_{15} + c_4 * d_{16} + c_4 * d_{17} + c_4 * d_{18}) \tag{8.6}$$

where, in Eqn. 8.6, $d_{11}, d_{12}, ..d_{18}$ are modeled as:

$$d_{11} = (c_4 * m_{11} + c_4 * m_{21} + c_4 * m_{31} + c_4 * m_{41} + c_4 * m_{51} + c_4 * m_{61} + c_4 * m_{71} + c_4 * m_{81}) \tag{8.7}$$

$$d_{12} = (c_4 * m_{12} + c_4 * m_{22} + c_4 * m_{32} + c_4 * m_{42} + c_4 * m_{52} + c_4 * m_{62} + c_4 * m_{72} + c_4 * m_{82}) \tag{8.8}$$

Similarly,

$$d_{18} = (c_4 * m_{18} + c_4 * m_{28} + c_4 * m_{38} + c_4 * m_{48} + c_4 * m_{58} + c_4 * m_{68} + c_4 * m_{78} + c_4 * m_{88}) \tag{8.9}$$

Similarly, other pixels of the block (M) are transformed where the input pixels remain the same but the 2D-DCT coefficients become different. Thus the structure of the equation remains the same, however, only the inputs will be different while computing different transformed image pixel intensities.

An equivalent DFG corresponding to Eqn. 8.6 denoting an unsecured (non-obfuscated) JPEG image compression is shown in Fig. 8.3. The complete DFG computes the $X_1 1$ is termed as macro IP, where, each d_{ij} (represented through Eqn. 8.7-Eqn. 8.9) is computed through a sub-DFG termed as micro IP. Each of the macro IP is designed using eight structurally equivalent micro IPs. The non-obfuscated macro IP with one zoom in micro IP is also shown in Fig. 8.3. Each micro IP operation, addition operation and multipli-

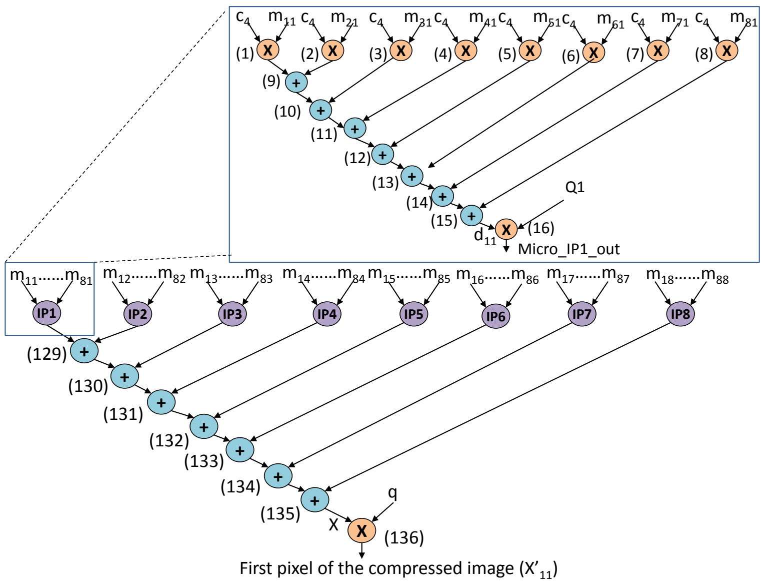


Figure 8.3: Non-obfuscated DFG of JPEG image compression for calculating first pixel of the compressed image (X'_{11})

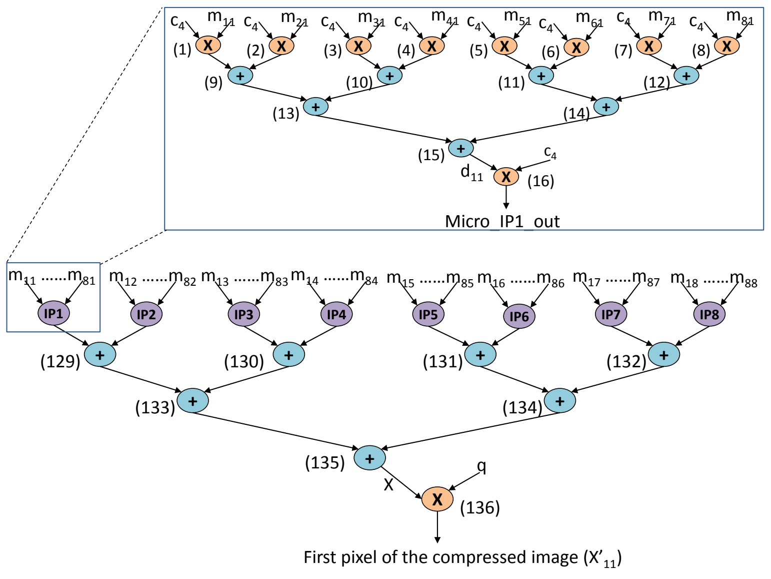


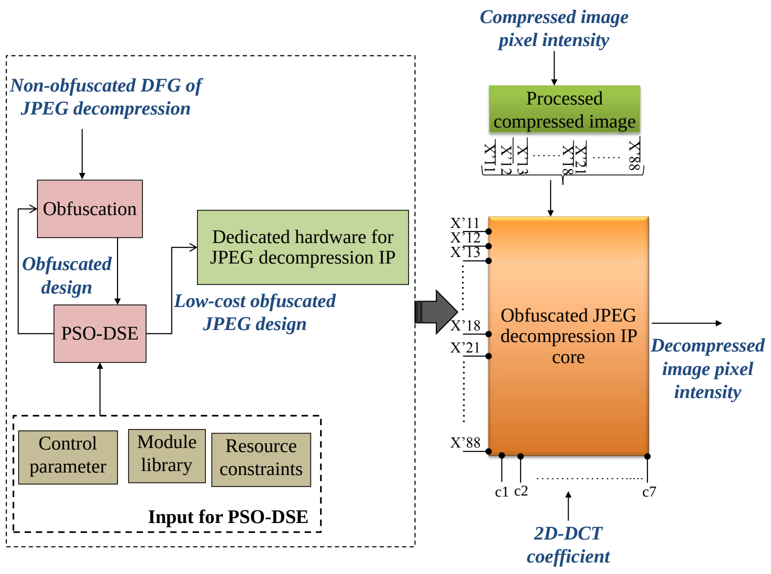
Figure 8.4: Obfuscated DFG of JPEG image compression for calculating first pixel of the compressed image (X'_{11})

cation operation are indicated by purple, blue and orange node respectively. As shown in Fig. 8.3 the output of operation 135 generates the pixel intensity of the transformed image (X). In this DFG, the output of node number 135 is multiplied with the inverse value of corresponding number (i.e. instead of division, multiplication is performed with the inverse of the number) of the quantization matrix (Q). This is achieved through operation 136 performing the quantization on the transformed image pixel intensities (X) based on the corresponding element 'q' of the quantization matrix. This produces the final output as quantized/compressed image pixel intensities (X'). This q value is feed as another input of the DFG based on the quality level of the compressed image.

8.2.4 Constructing obfuscated JPEG compression IP core

This sub-section, describe the process of constructing obfuscated JPEG compression DFG from the non-obfuscated DFG generated in the last sub-section. In the proposed approach, the complete JPEG compression DFG is structurally obfuscated through THT-based high-level transformation technique.

THT is a compiler-driven optimization technique which parallelizes the dependent critical path operations by dividing it into temporary sub-computations. However, this tech-



Compressed image pixel intensity
Non-obfuscated DFG of JPEG decompression
Processed compressed image
X'11
X'12
X'13
X'18
X'21
X'88
Obfuscation
Dedicated hardware for JPEG decompression IP
Obfuscated design
Low-cost obfuscated JPEG design
PSO-DSE
Obfuscated JPEG decompression IP core
Decompressed image pixel intensity
Control parameter
Module library
Resource constraints
Input for PSO-DSE
c1
c2
c7
2D-DCT coefficient

Figure 8.5: Obfuscated JPEG decompression IP core

nique has not been used to generate a structurally obfuscated DFG for JPEG. The complete transformed structure of macro IP with one zoom in micro IP is shown in Fig. 8.4. As shown in the figure, due to this transformation, each micro IP, as well as the macro IP, is changed structurally. For example, in micro IP1 total 12 nodes out of 16 nodes are structurally changed. They are node number 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14 and 15. Therefore, in the eight micro IPs, 96 nodes are affected, additionally, node no. 130, 131, 132, 133,134 and 135 are also modified. So total, 102 nodes of the complete micro IP is modified. For these aforesaid nodes, either the input/ output connectivity to the resources changes or the number of resources per control step changes, which resulting into different datapath architecture and controller logic once processed through HLS [53].

8.2.5 Overview of proposed methodology for decompression

The proposed low-cost, obfuscated JPEG decompression IP core is designed through multiple steps (as shown in Fig. 8.5). Similar to the proposed obfuscated JPEG compression IP core design process, the proposed low cost obfuscated JPEG decompression IP core design process also accepts an unsecured (non-obfuscated) JPEG decompression DFG as input. Similarly, as performed in compression of JPEG IP core, resiliency against RE in the form of structural obfuscation is provided to the unsecured DFG to obtain an obfuscated decompression DFG. This obfuscated DFG is processed through an optimization framework to obtain a low-cost hardware configuration (discussed in Sec 8.1.3). Thus, this low-cost hardware configuration is used to design an obfuscated dedicated hardware for JPEG decompression IP core. Finally, for generating the decompressed pixel intensities through the proposed IP core, 2D-DCT coefficients and standard quantization matrix are fed as inputs. The detailed is available in the next sub-section.

8.2.6 Constructing obfuscated JPEG decompression IP core

The compressed block X' is decompressed through inverse quantization. In this process, each element of X' is multiplied with the corresponding element of the quantization matrix of the same quality level, to obtain decompressed block X". As mentioned in Section 8.2.1 Eqn. 8.4 and Eqn. 8.5 represent the function of applying 2D-IDCT on the decompressed block X". To convert relationship into a hardware function for dedicated IP core design, this function is expanded and modeled as a mathematical expression to design an equivalent DFG. Further, this DFG is feed to an HLS engine to obtain a JPEG compression IP core. Based on Eqn. 8.4, the corresponding mathematical expression for the first pixel (o_{11}) of the inversely transformed block is modeled as:

$$o_{11} = (c_4 * e_{11} + c_1 * e_{12} + c_2 * e_{13} + c_3 * e_{14} + c_4 * e_{15} + c_5 * e_{16} + c_6 * e_{17} + c_7 * e_{18}) \tag{8.10}$$

where, in Eqn. 8.10, $e_{11}, e_{12}, ..e_{18}$ are modeled as:

$$e_{11} = (c_4 * X''_{11} + c_4 * X''_{21} + c_4 * X''_{31} + c_4 * X''_{41} + c_4 * X''_{51} + c_4 * X''_{61} + c_4 * X''_{71} + c_4 * X''_{81}) \tag{8.11}$$

$$e_{12} = (c_4 * X''_{12} + c_4 * X''_{22} + c_4 * X''_{32} + c_4 * X''_{42} + c_4 * X''_{52} + c_4 * X''_{62} + c_4 * X''_{72} + c_4 * X''_{82}) \tag{8.12}$$

Similarly,

$$e_{18} = (c_4 * X''_{18} + c_4 * X''_{28} + c_4 * X''_{38} + c_4 * X''_{48} + c_4 * X''_{58} + c_4 * X''_{68} + c_4 * X''_{78} + c_4 * X''_{88}) \quad (8.13)$$

Similarly, other pixels of the de-quantized image block (X") are transformed through inverse DCT matrix where the input pixels remain same but the 2D-IDCT coefficients become different. It is to be noted that the structure and pattern of forward DCT 8.6 and inverse DCT 8.10 are same, but only the inputs are different during the computation of decompressed image pixel intensity.

An equivalent DFG corresponding to Eqn. 8.10 denoting an unsecured (non-obfuscated) JPEG image decompression can be obtained. As performed for JPEG compression DFG, obfuscation through THT can be performed on the non-obfuscated JPEG decompression DFG. Each micro IPs, as well as the complete macro IP, is structurally obfuscated through THT. Similar to JPEG image compression DFG, an obfuscated JPEG decompression macro IP is designed using eight structurally equivalent micro IPs, where each micro IP executes a part of Eqn. 8.10. Thus it can be concluded that the DFG of JPEG DCT and IDCT are almost same except few things, such as: (a) in case of compression it is assumed that each element of the compressed block is multiplied with the corresponding element (q) of the quantization matrix, to perform quantization (through node number 136), whereas, quantization through node number 136 is not required in DFG of JPEG IDCT, (b) the last node (node 136) can be used to perform de-levelization by adding 128 to the pixel intensity of the decompressed block. Thus the total number of nodes in the DFG for both JPEG compression and decompression remains the same. Therefore, THT-based structurally obfuscated DFG can be similarly generated for the JPEG decompression also.

8.3 Proposed Design of JPEG CODEC IP Core

This section, explains the design implementation of the proposed JPEG CODEC IP core and demonstrate the process with a grayscale image using the generated IP core.

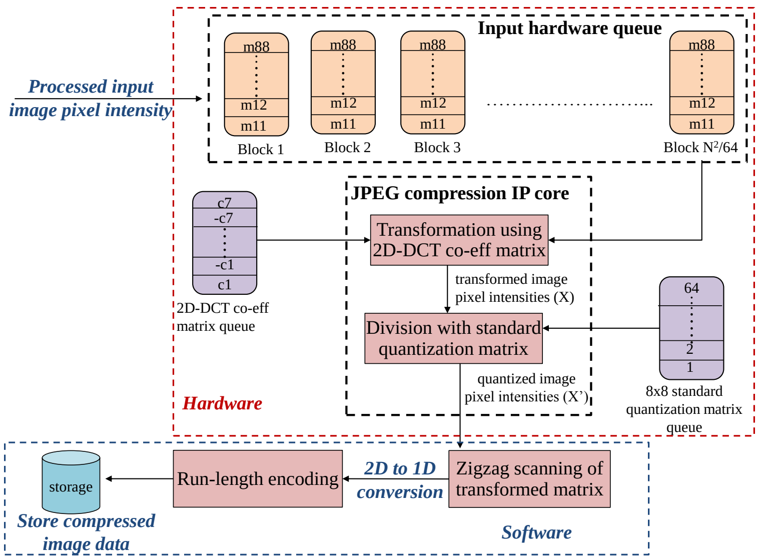

Processed input image pixel intensity
Input hardware queue
m88
m12
m11
Block 1
Block 2
Block 3
Block N²/64
JPEG compression IP core
c7
-c7
-c1
c1
2D-DCT co-eff matrix queue
Transformation using 2D-DCT co-eff matrix
transformed image pixel intensities (X)
Division with standard quantization matrix
64
2
1
8x8 standard quantization matrix queue
quantized image pixel intensities (X')
Hardware
storage
Run-length encoding
2D to 1D conversion
Zigzag scanning of transformed matrix
Store compressed image data
Software

Figure 8.6: Proposed hardware and software design flow using JPEG compression IP

8.3.1 Designing of obfuscated JPEG compression IP core

The complete design flow of the JPEG compression process using devised low-cost obfuscated JPEG compression IP core is shown in Fig. 8.6. In this complete design flow, partial steps are performed through hardware and remaining are performed through software. More specifically, 2D-DCT transformation and quantization steps are performed through devised JPEG compression IP core and leveling of input image pixel intensity and data encoding is performed through software. The IP core used is capable to accept an 8x8 block of a processed gray-scale image pixel intensity stored in a hardware queue along with 2D-DCT coefficients and standard quantization matrix. It performs the transformation using the 2D-DCT coefficient, quantizes and round off the transformed result and finally generates the pixel intensities of the compressed image data (X). The JPEG compression IP core is designed using HLS framework, where the obfuscated DFG (shown in Fig. 8.4) is scheduled (using list scheduling) based on a low-cost resource configuration i.e., 3 adders (Adder_A1, Adder_A2 and Adder_A3) and 3 multipliers (Multiplier_M1, Multiplier_M2 and Multiplier_M3) explored through PSO-DSE approach. After scheduling, resource allocation and binding of resources are performed. Table 8.1 reports all the 136 operations, their control step and corresponding assigned hardware.

Table 8.1: Scheduling and binding of operations for low-cost obfuscated JPEG compression IP core

CS	Opn.No. assigned to M1	Opn.No. assigned to M2	Opn.No. assigned to M3	Opn.No. assigned to A1	Opn.No. assigned to A2	Opn.No. assigned to A3
1	1	2	3	-	-	-
2	4	5	6	9	-	-
3	7	8	17	10	11	-
4	18	19	20	12	13	-
5	21	22	23	25	26	14
6	24	33	34	27	29	15
7	35	36	37	28	41	-
8	38	39	40	42	30	-
9	49	50	51	43	44	45
10	52	53	54	57	46	31
11	55	56	65	58	59	47
12	66	67	68	60	61	-
13	69	70	71	73	74	62
14	72	81	82	75	77	63
15	83	84	85	76	89	-
16	86	87	88	90	78	-
17	97	98	99	91	92	93
18	100	101	102	105	94	79
19	103	104	113	106	107	95
20	114	115	116	108	109	-
21	117	118	119	121	122	110
22	120	16	32	123	125	111
23	48	64	80	124	129	-
24	96	112	-	126	130	-
25	-	-	-	127	131	133
26	-	-	128	-	-	-
27	-	-	-	132	-	-
28	-	-	-	-	134	-
29	-	-	-	-	-	135
30	-	-	136	-	-	-

It can be observed from the table, total 30 control steps are required to schedule all the operations. Both adders and multipliers are designed in IEEE 754 half precision floating-point format. Therefore, all the operators have bit-width of 16 bits. The implemented JPEG compression IP core consists of two units: data-path unit and controller unit. The data-path unit represents the required number of resources and their interconnections while the controller unit synchronizes those resources by changing their signals (e.g. enables, selectors, strobes etc.). In other words, controller unit is responsible for controlling the data flow through the data-path by activating and deactivating different components of data-path in different clock cycle, thus synchronizing each component to generate an error-free output. The generic block diagram representation of data-path unit of low-cost obfuscated JPEG IP is shown in Fig. 8.7. As shown in this figure, the data-path is consist of 3 adders and multipliers, 10 multiplexers of type 32:1, 2 multiplexers of type 16:1, 5 demultiplexers of type 1:32, 1 demultiplexer of type 1:16, 40 registers and 18 latches. The primary inputs are represented through A1,..A8,.. Q1,..Q8 and the dependent inputs are represented through i1, i2,.. i135, while i136 represents the output. Both the data-path unit and control unit of the proposed low-cost obfuscated JPEG CODEC is implemented using Quartus CAD tool in Intel Cyclone II FPGA.

8.3.2 Designing of obfuscated JPEG decompression IP core

The complete design flow of JPEG decompression process using devised low-cost obfuscated JPEG decompression IP core is shown in Fig. 8.8. Similar to the JPEG compression design flow, few steps are performed through software and rest is performed through hardware. More specifically, data decoding and zigzag scanning are performed through software and inverse quantization, 2D-IDCT transformation and leveling of output image pixel intensity are performed through devised JPEG decompression IP. Finally, each 8x8 block with leveled off decompressed pixel intensity is stored in an output hardware queue.

Similar to JPEG compression, same resource configuration is used while performing scheduling for JPEG decompression IP core. Total 3 adders and 3 multipliers of size 16-bits are used to implement the design. Additionally, 10 multiplexers of type 32:1, 2 multiplexers of type 16:1, 5 demultiplexers of type 1:32, 1 demultiplexer of type 1:16, 40 registers and 18 latches are used for the data-path unit. Both data-path unit and control unit of JPEG decompression IP core is implemented using Quartus CAD tool in Intel Cyclone II FPGA.

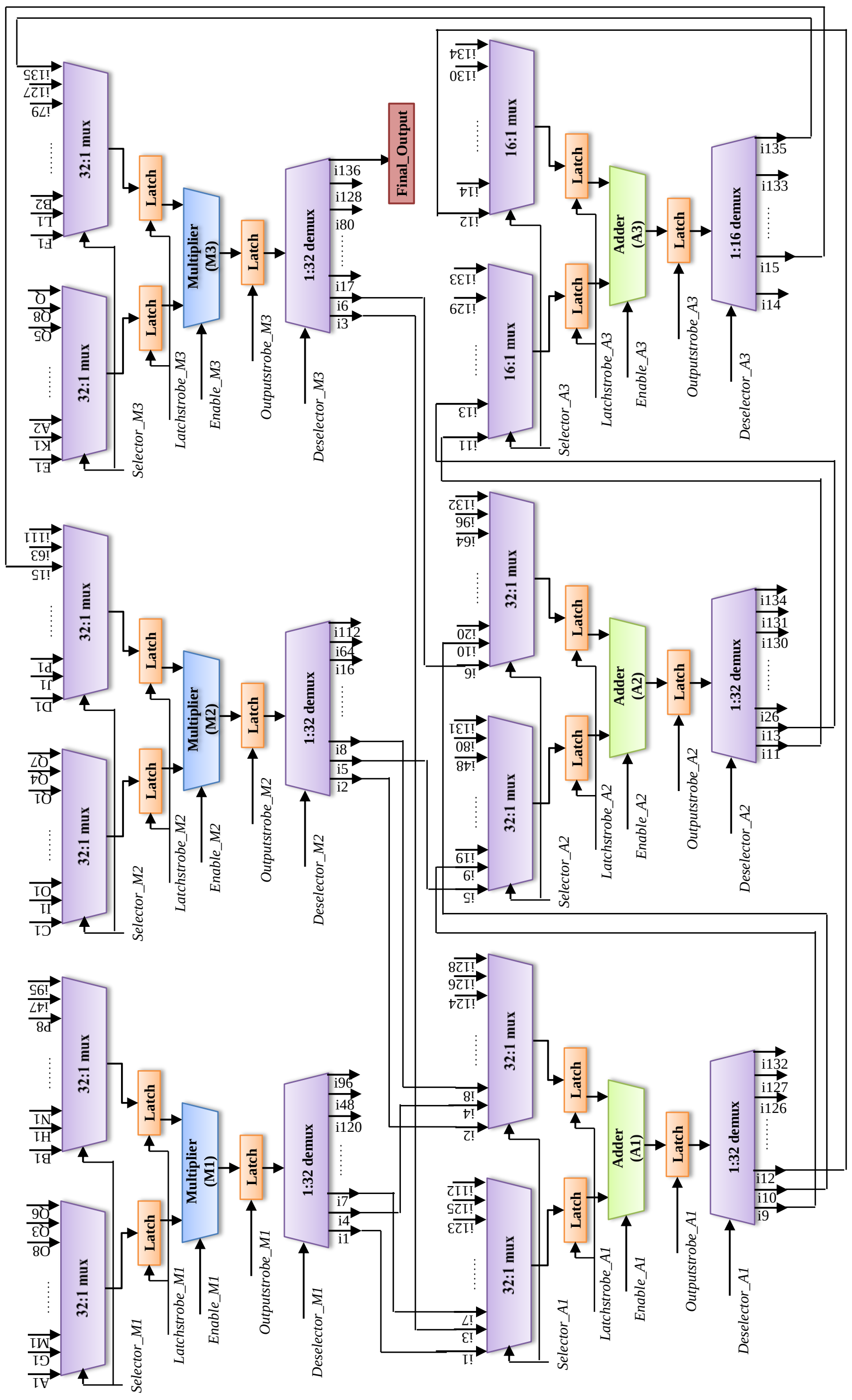


Figure 8.7: Block diagram representation of data-path unit of proposed low-cost obfuscated JPEG CODEC IP core

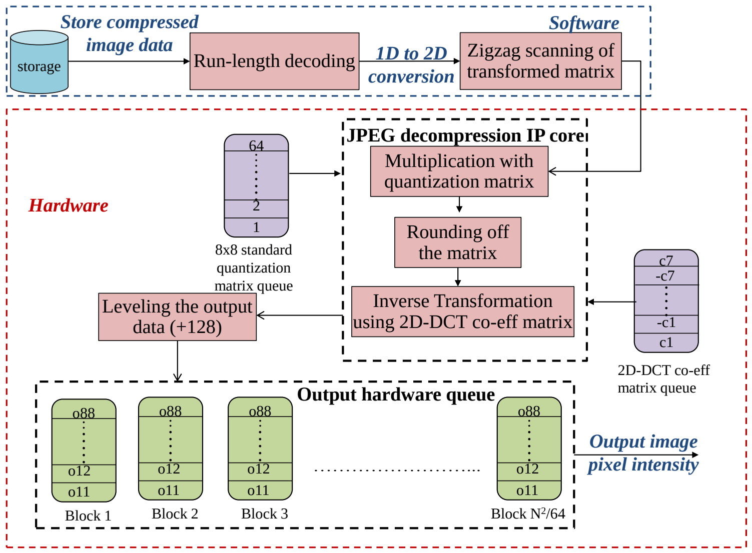

Store compressed image data
Software
storage
Run-length decoding
1D to 2D conversion
Zigzag scanning of transformed matrix
Hardware
JPEG decompression IP core
Multiplication with quantization matrix
Rounding off the matrix
Inverse Transformation using 2D-DCT co-eff matrix
8x8 standard quantization matrix queue
2D-DCT co-eff matrix queue
Leveling the output data (+128)
Output hardware queue
Block 1
Block 2
Block 3
Block N²/64
Output image pixel intensity

Figure 8.8: Proposed hardware and software design flow using JPEG decompression IP

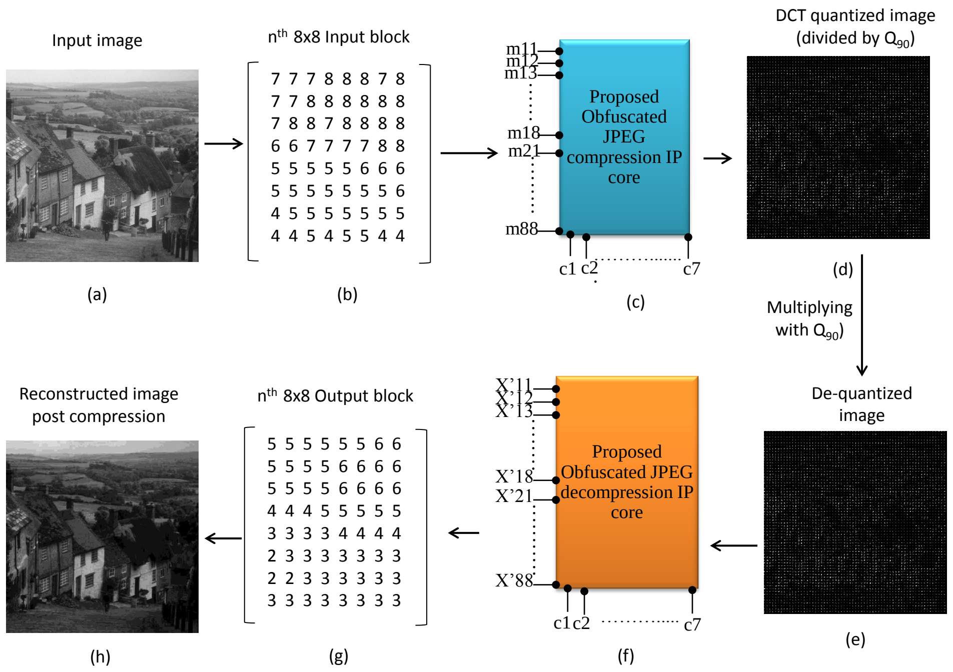

Input image
nth 8x8 Input block
DCT quantized image (divided by Q90)
Proposed Obfuscated JPEG compression IP core
Multiplying with Q90)
De-quantized image
Proposed Obfuscated JPEG decompression IP core
nth 8x8 Output block
Reconstructed image post compression
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)

Figure 8.9: Depiction of end to end process of JPEG image compression and decompression through the proposed JPEG hardware

8.3.3 End to End demonstration of JPEG CODEC through the designed IP core

For demonstration, a two-dimensional 512x512 grayscale image shown in Fig. 8.9 (a) is taken as the input in the form of a matrix. This input matrix is then sub-divided into multiple non-overlapping 8x8 blocks. Fig. 8.9 (b) represents the n^{th} 8x8 block of the input image which is compressed through the proposed compression IP core (shown in Fig. 8.9 (c)). The output of the proposed JPEG compression IP core produces DCT quantized image pixel intensity values. The DCT quantization image of the corresponding input image is shown in Fig. 8.9 (d), where quantization is performed based on quantization matrix Q_{90}. To reconstruct the de-quantized image, the DCT quantized image (after multiplying with Q_{90} quantization matrix) as shown in Fig. 8.9 (d) is again sub-divided into multiple non-overlapping 8x8 blocks (shown in Fig. 8.9 (e)). Each block of the de-quantized image is then decompressed through the proposed JPEG decompression IP core, as shown in Fig. 8.9 (f) to generate 8x8 output blocks. Fig. 8.9 (g) represents the n^{th} 8x8 block of the output image. After combing all 8x8 output blocks of the decompressed image, the output reconstructed image of size 512x512 is generated, as shown in Fig. 8.9 (h).

8.4 Summary

This chapter presented a novel solution to secure one of the most popularly used image CODEC IP core through obfuscation. The work presented a low-cost structurally obfuscated JPEG CODEC as a dedicated IP core to thwart RE attack. To obfuscate the standard JPEG CODEC architecture, THT technique is used. It transforms the standard JPEG CODEC architecture to a non-obvious one. Thus, obscure the original functionality. Further, a low-cost hardware configuration is explored through a PSO-DSE engine. The complete IP core is synthesized and simulated through Intel Quartus Cyclone II FPGA. This is the first approach in the literature that provides low-cost, structurally obfuscated JPEG CODEC to enhance the complexity of RE attack. Several novelties integrate the framework of the proposed approach: (a) THT technique is used to generate an obfuscated JPEG CODEC during HLS, (b) explores a low-cost obfuscated design through the PSO-DSE framework, (c) The proposed obfuscation achieves 76% PoO compared to a standard non-obfuscated JPEG CODEC design.

Chapter 9

Hardware Trojan Secured IP Core for CE Systems

Hardware Trojans are malicious logic or hardware components implanted by a rogue entity to induce malfunctioning of ICs. A hardware Trojan affected DSP IP core used in the SoC of a CE system can create malfunctioning of the whole system. High-level synthesis is the only solution to handle complex design algorithm of DSP IP cores. In high-level synthesis, design library may comprise of third-party IP cores as modules. The reason is to increase the design productivity. However, there are serious security concerns for design integrators due to the involvement of third-party IP vendors owing to globalization involved. These third-party IP vendors are untrustworthy, therefore the possibility of inserting Trojan logic in the IP cores is extremely high. This may result into erroneous computation of a digital design. Normally, these types of Trojans are inactive until triggered and can be activated either through external activation (such as antennas or sensors) or internal activation (such as internal states of finite state machine or counters), resulting in malfunctioning. The typical design cycle involving 3PIP vendors, in-house system design and the foundry is shown in Fig. 9.1 . Here, the in-house system design and the foundry is considered as trustworthy and the 3PIP vendors as untrustworthy.

Therefore, to have a Trojan secured design of a DSP IP core, it should be ensured that any possible infection of 3PIP is detectable through HLS framework. Thus, a paradigm shift in CE systems is needed to incorporate trust/security objective besides power, area,

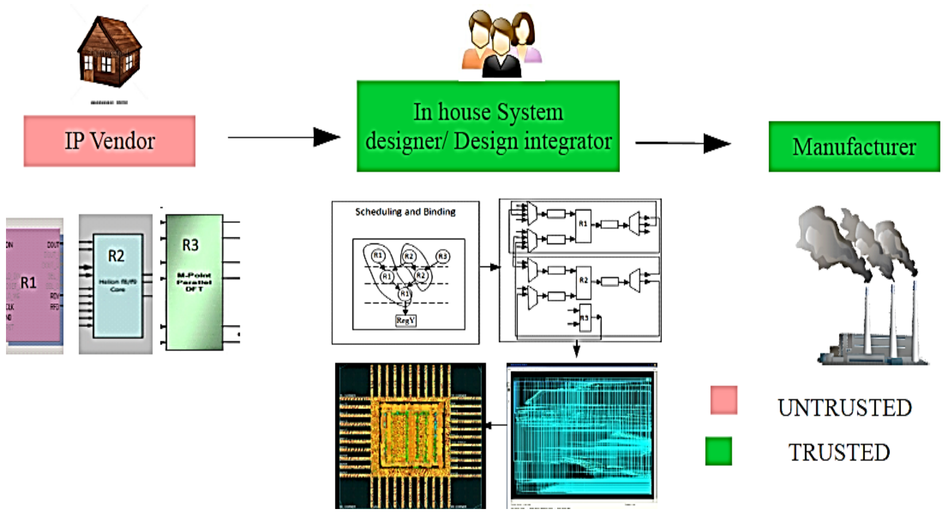


Figure 9.1: Design cycle of a SoC involving 3PIP core

energy etc. The designs generated should be equipped with low-cost detection capability to handle hardware Trojan that targets alteration of computational value. Typically, the hardware design with Trojan detection (security) capability generated through HLS methodology contains additional RTL blocks as detection logic resulting from imposed security constraints. Trojan security constraints in the form of dual modular redundancy include duplication of functional modules which incur additional RTL logic, resulting in both delay and area overhead. As providing Trojan detection capability to HLS designs will in most cases require additional digital circuitry hence exploring a low-cost solution is crucial besides providing Trojan security.

This chapter presents, a novel low-cost, Trojan security aware design methodology to handle single and nested loop DSP kernels while satisfying the user constraints. In the proposed approach, simultaneous exploration of low-cost Trojan security aware DMR schedule, optimal loop unrolling factor and vendor allocation is performed for a DSP application that abides by the user provided area-delay constraints. Additionally, this approach handles single and nested loop based DSP cores. The particle encoding scheme during PSO-DSE process comprises of candidate schedule resources, candidate loop unrolling factor and candidate vendor allocation information. Further, the proposed approach achieves an average reduction in the final cost of 12% compared to recent approach.

The chapter is organized as follows: Section 9.1 discusses the threat model and explain with an example; Section 9.2 discusses the problem formulation and evaluations models; Section 9.3 explains the proposed methodology; Section 9.4 explains the PSO-DSE methodology to obtain low-cost Trojan secured design; finally, the summary of this chapter is presented in Section 9.5. (*NOTE: All the abbreviations and taxonomy are listed in 'Acronyms' and 'Nomenclature' section respectively.*)

9.1 Threat Model and An Example

Normally, a Trojan in a circuit is in the inactive state as its trigger is not enabled, which indicates there is no payload effect. A Trojan gets triggered under certain rare specific criteria or events and then its payload introduces malfunctioning in the circuit. Trojan triggering process can be classified into three categories [66]: 1) rare value triggered, 2) time triggered and 3) both time and value triggered. Payload effect on a triggered Trojan depends on the application [64]. It may disable the system or leak vital information or change the output value of the circuit.

9.1.1 Threat model

There are various classes of hardware Trojan available for different circuits. However, in the context of DSP applications, Trojans that affects the functional output is most vulnerable. Therefore, the class of Trojan that only changes the computation output value of an IP core is addressed here.

These type of Trojan affected IP blocks or modules may be available in the library of an HLS tool. Insertion of Trojan into the IP block/module is possible through an untrustworthy 3PIP vendor from where the design team of the HLS tool imports the IP block/module. Therefore, in this threat model, it is considered that only the third-party IP vendor as untrustworthy whereas, the in-house design integrator team and foundry as trustworthy in the SoC design cycle (shown in Fig. 9.1).

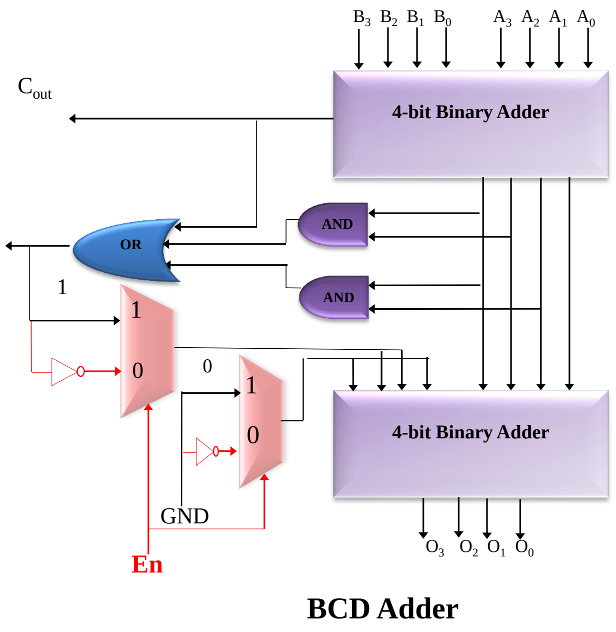


Figure 9.2: A Trojan infected BCD adder; Note: it will be triggered when 'En'=0

9.1.2 Example of a Trojan infected 3PIP module

This section discusses the aforementioned Trojan with an example. Let us assume there is a BCD adder IP available in the module library of an HLS tool as a black box (shown in Fig. 9.2). It has three inputs: (a) Enable ('En'), which switches on/off the execution of the adder (b) first primary input 'A' of bit width 4 ($A_3A_2A_1A_0$) (c) second primary input 'B' of bit width 4 ($B_3B_2B_1B_0$) and one output 'O' of bit width 4 ($O_3O_2O_1O_0$). During functional/simulation verification and test the 'En' signal is on ('En' = 1) to start the execution of the BCD adder IP i.e. the upper mux passes '1' and lower mux passes '0' to finally form bits '0110' (decimal of '6'). This value is added to correct any invalid BCD summation produced and yield a result between valid decimal values 0-9. After finishing the verification process the 'En' signal is off ('En' = 0) to stop the execution of the BCD adder IP. However, the Trojan is implanted in the BCD adder (shown in Fig. 9.2) in such a way that, whenever the 'En' signal remains 'off', it actually continues to work maliciously (due to Trojan logic activation) thereby producing the wrong output of a DSP IP core.

9.1.3 Why detection of such Trojan is difficult?

Detection of aforementioned Hardware Trojan is difficult during verification/tests because of the following reasons:

- A Trojan logic infected 3PIP produces functionally correct output in normal condition i.e. until the Trojan logic is triggered.
- Trojan logic is triggered under very rare and specific time/condition (discussed earlier), therefore identifying such condition or event is difficult during functional verification.
- Generally, during functional verification, only the functionality of the entire SoC is verified, not each and every IP/module used (imported) in the system (after system integration) is checked.
- Normally the Trojan logic is set in an IP in such a way that it remains dormant during functional verification and triggered when the complete SoC is deployed in the real-time situation.
- As there is no trustworthy golden IP (i.e. guarantee of 100% Trojan free IP) model available, the evasion of hardware Trojan in a 3PIP during HLS design for DSP core could be feasible.
- Physical inspection and reverse engineering which is a very costly and complex procedure in nature do not guarantee to detect such Trojan.

9.2 Formulation and Evaluation Models

This section presents the problem formulation and evaluation models of the proposed approach.

9.2.1 Problem formulation

To generate a low-cost, Trojan secured DMR schedule for a DSP application (given in the form of a C-code or control data flow graph (CDFG)) within the user-specified constraints, which accepts module library and PSO control parameters, as inputs can be formulated as follows:

Minimize: Trojan secured design cost (A_T^{DMR}, L_T^{DMR}) using optimal design solution (X_i).

Subject to: $A_T^{DMR} \leq A_{cons}$ and $L_T^{DMR} \leq L_{cons}$.

Based on the type of the DSP application, the design solution (X_i) can be represented in three different ways:

$(X_i) = \{N(R_1), N(R_2), ..., N(R_D), A_v\}$, (for non-loop DFG)

$(X_i) = \{N(R_1), N(R_2), ..., N(R_D), U, A_v\}$, (for single loop CDFG)

$(X_i) = \{N(R_1), N(R_2), ..., N(R_D), U_1, U_2, ..U_n, A_v\}$, (for n-times nested loop CDFG)

9.2.2 Evaluation models

In the proposed approach the Trojan secured design cost is evaluated based on the area and latency of the design where each design solution comprises of hardware resource configuration, loop unrolling factor and vendor allocation mode. This section discusses the evaluation models of total design area, execution latency and design cost.

Area model

Total area consumed (A_T^{DMR}) by a resource set is evaluated as:

$$A_T^{DMR} = \sum_{j=1}^{2} \sum_{i=1}^{m} \Big(A\big(R_i^{V_j}\big) \times N\big(R_i^{V_j}\big) \Big), \tag{9.1}$$

It is assumed there are 3PIP cores of minimum 2 distinct vendors that are available in the module library to achieve distinctness. The resource type R_i comprises of functional resources (such as adder, multiplier etc.), internal storage elements (latch, register), interconnecting units (multiplexer and demultiplexer), and comparator (for error detection).

The total design area is assessed with respect to 90 nm CMOS-based technology node.

Latency model

The execution latency is evaluated after derivation of the delay model using the following two cases:

Single loop CDFG: When the input DSP application have a single loop the total no. of CSs is derived as:

$$C_T^{DMR} = \underbrace{\left(C_{body}^{DMR} * \left[\frac{I}{U}\right]^{floor}\right)}_{\text{CSs for unrolled loop}} + \underbrace{\left((ImodU) * C_{first}^{DMR}\right)}_{\text{CSs for sequential loop}} \tag{9.2}$$

Hence the execution latency is calculated as:

$$L_T^{DMR} = \Delta * C_T^{DMR} \tag{9.3}$$

where, 'Δ' is the latency of one CS in nanoseconds. The total execution latency is assessed with respect to 90 nm CMOS-based technology node.

DFG and Nested Loop CDFG: When the input DSP application either have no loop or have nested loop the total no. of CSs is derived as:

$$L_T^{DMR} = \sum_{CS=1}^{CS=max} Max\Big(D\big(R_i^{V_j}\big), ..D\big(R_k^{V_j}\big), ..D\big(R_i^{\prime V_j}\big), ..D\big(R_k^{\prime V_j}\big)\Big)^{CS} \tag{9.4}$$

where, $1 \leq i \leq k$, resources in original unit and duplicate unit is labeled as R_i and R_i' respectively and V_j indicates vendor type of the resources.

Fitness model

The fitness function (considering execution time and area consumption of a solution) is formulated as:

$$C_f(X_i) = W_1\left(\frac{A_T^{DMR} - A_{cons}}{A_{max}^{DMR}}\right) + W_2\left(\frac{L_T^{DMR} - L_{cons}}{L_{max}^{DMR}}\right). \tag{9.5}$$

W_1 and W_2 are the user defined weights both kept at 0.5 during exploration to provide equal preference. The equation above yield a normalized value of cost (between 0 and 1). Since the objective of the proposed exploration approach is to satisfy the user constraints as well as minimize the hybrid cost, hence a higher negative value indicates a more desirable/fitter solution.

9.3 Proposed Methodology for Trojan Security Aware DSP IP Core

The proposed Trojan security methodology uses PSO driven DSE process for generating an optimal Trojan secured schedule based on area-delay constraint (shown in Fig. 9.3). A Trojan secured schedule is obtained as follows: a) duplicate all the operations of original CDFG to form a duplicate one, which together is called 'dual modular redundant system' b) Perform scheduling of operations of DMR system to obtained scheduled DMR c) Finally, perform hardware allocation of operations of DMR schedule based on proposed Trojan security allocation rules (i.e. assigning hardware from distinct/multiple vendors to similar operations of original and duplicate unit).

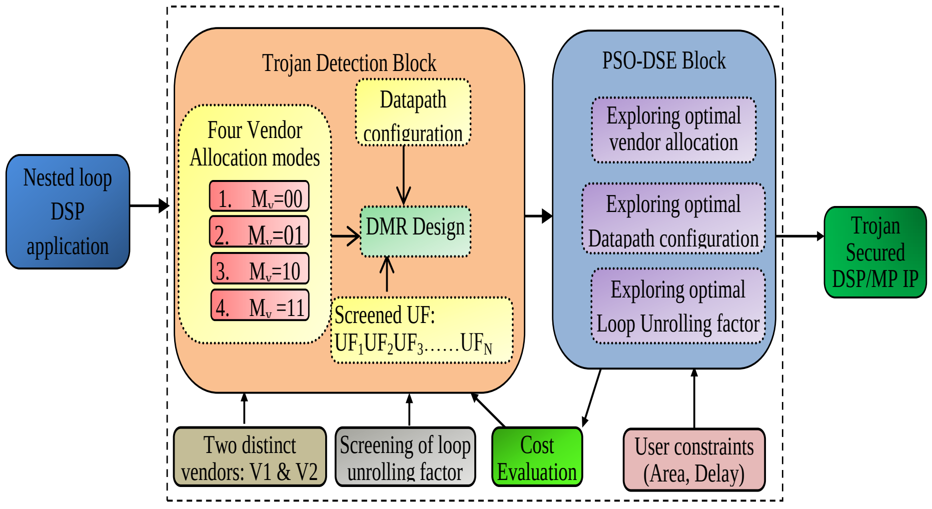

Nested loop DSP application
Trojan Detection Block
Four Vendor Allocation modes
1. M_v=00
2. M_v=01
3. M_v=10
4. M_v =11
Datapath configuration
DMR Design
Screened UF: $UF_1UF_2UF_3......UF_N$
PSO-DSE Block
Exploring optimal vendor allocation
Exploring optimal Datapath configuration
Exploring optimal Loop Unrolling factor
Trojan Secured DSP/MP IP
Two distinct vendors: V1 & V2
Screening of loop unrolling factor
Cost Evaluation
User constraints (Area, Delay)

Figure 9.3: Overview of proposed low-cost Trojan secured design

```
Input –value of ‘I’(Total no. of loop iteration)
Output –screened set of unrolling factor (U)
1.      Begin
        // Screening of U//
2.      int k = 0
3.      For U=2 to I Do
3.1             IF ((I mod U < (U/2) ) && (U <= I/2)) Then
                //Add Uinto the accepted U list//
3.2             Accepted U [k] = U
3.3             k++
3.4             End IF
3.5     End For
4.      End
```

Figure 9.4: Algorithm for preprocessing of unrolling factors

9.3.1 Particle encoding

The proposed methodology presents a novel encoding scheme for the particle which comprises of resource configuration ($\vec{R_n}$), unrolling factor (U) for single loop applications, iteration counts (U_1, U_2) for nested loop applications and vendor allocation procedure type (A_v). Therefore, a particle position (candidate design solution) is labeled as X_i for:

$X_i = (\vec{R_n}, U, A_v)$ (for single loop CDFG);

$X_i = (\vec{R_n}, U_1, U_2, ..U_n, A_v)$ (for nested loop CDFG).

9.3.2 Preprocessing of unrolling factor candidates

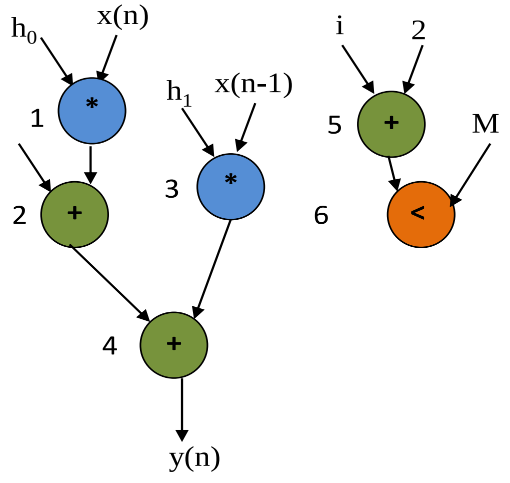
Preprocessing (screening) of unrolling factor candidates that do not form part of an optimal solution during exploration is extremely crucial. Some unrolling factors such as the ones which yield large trailer loops are potential sources for greater delay due to multiple sequential loops involved. Further, it is established in [55], that performance is not a monotonically increasing function of unrolling factor value, i.e., for large unrolling factor values; therefore, the performance improvement is found marginal. The corresponding algorithm is shown in Fig. 9.4.

9.3.3 Designing DMR schedule

This section describes how to generate DMR schedule of an DSP application available in the form of a ‘C-code’ or transfer function.

```
for(i=0; i<M; i=(i+2)) // M = order of FIR filter
{
        y(n)+=h_i*x(n-i);
        y(n)+=h_{i+1}*x(n-(i+1));
}
```

(a)



(b)

Figure 9.5: Corresponding to the FIR application: (a) the C-code for unrolling factor 2 , (b) equivalent CDFG for unrolling factor 2

A FIR application is used to demonstrate the Trojan secured design process. The generic equation of FIR filter can be expressed as follows:

$$y(n) = \sum_{i=0}^{M} h_i * x(n-i), \tag{9.6}$$

where M is the filter order of FIR, h_i is the FIR coefficient, x(n) is input impulse and y(n) is output impulse. Using the above equation, the FIR filter for unrolling factor can be represented as,

$$y(n) = h_0 * x(n) + h_1 * x(n-1) \tag{9.7}$$

The corresponding C-code of FIR application is shown in Fig. 9.5(a). For unrolling factor 2, the corresponding CDFG of FIR is shown in Fig. 9.5(b).The blue node indicates multiplier, the green node indicates adder and the orange node indicates comparator. The next step is to construct the DMR of the application. In order to obtain DMR CDFG, complete duplication of all the unrolled operations is performed. The unrolling factor is obtained from the candidate design solution X_i. For example, the DMR of FIR filter for unrolling factor 2 is shown in Fig. 9.6, where U^{OG} indicates operations of the original unit and U^{DP} indicates operations of the duplicate unit.

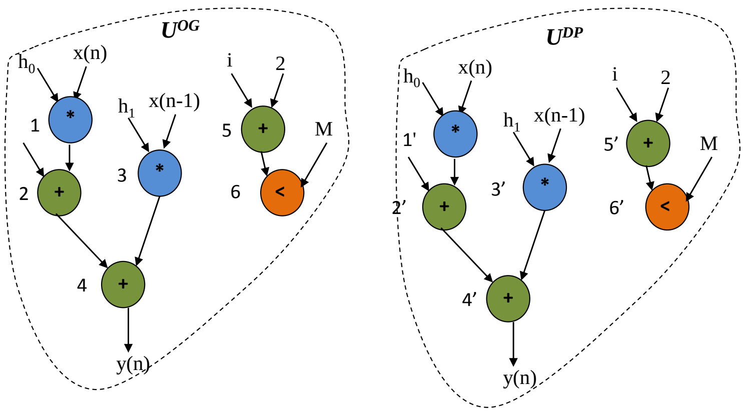


Figure 9.6: DMR CDFG of FIR benchmark for unrolling factor 2

After obtaining the DMR CDFG, the next step is to schedule all the operations of the original and duplication unit simultaneously based on resource configuration. The resource configuration is obtained from the candidate design solution X_i. Once the scheduling of the DMR is performed the next step is to employ allocating 3PIP cores imported from different vendors. In this context, four distinct vendor allocation rules are proposed.

9.3.4 Distinct vendor allocation rules

In order to detect Trojans a minimum of two vendors is always needed to provide distinctness. However, technique of usage of the two vendors during allocation inside the DMR scheduling (i.e. assignment process of each vendor IPs inside the system during allocation) dictates the final latency and area of the entire system. This is because same resource type/IP from two different vendors have different area and delay. Further, it is assumed that the IP characteristics from vendors (V1 and V2) are as follows: Multiplier and adder provided by vendor V1 has area='2468au' & '2034au', latency='10000 ns' & '265 ns'; assuming 1 au = 1 transistor) while multiplier and adder provided by vendor V2 has area='2464au' & '2032au', latency='11000 ns' & '270 ns' respectively. The values of area and delay of modules assumed is with respect to 90 nm technology scale. Hence, merely using distinctive vendor assignment for detection without probing into the procedure of allocation (assignment) of vendor type in DMR system may lead to skipping of an alternate better solution in the context of DSE of Trojan secured schedule. Therefore, exploration of an additional dimension 'A_v' (indicating allocation procedure of IP's from different vendor type) which can either be in one of the following distinct vendor allocation modes, mode 1:'00' or mode 2:'01' or Type 3:'10' or Type 4:'11' is incorporated in the particle encoding along with resource array. The value of 'A_v' as '00' or '01 or '10' or '11' is interpreted as follows:

Rule 1: vendor allocation procedure for $A_v = 00$

- Alternate vendor assignment to operations in control step of a unit (Example, in Fig. 9.7(a), operation 1 & 3 assigned alternatively to 'V1' and 'V2'. Next multiplication if any would have been assigned to 'V1' alternately).
- Similar operations of both U^{OG} and U^{DP} being assigned to different vendors.

Rule 2: vendor allocation procedure for $A_v = 01$

- All operations of a specific unit being strictly assigned to resources of same vendor type (Example, in Fig. 9.7(b),all operations of original unit is assigned to vendor 'V1' and all operations of duplicate unit is assigned to vendor 'V2).
- Similar operations of both U^{OG} and U^{DP} being assigned to different vendors.

Rule 3: vendor allocation procedure for $A_v = 10$

- All operations within critical path of a specific unit being strictly assigned to a vendor type while all operations of non critical path through alternate vendor type (Example, in Fig. 9.7(c),all operations within critical path of original unit is assigned to vendor 'V1' and all operations within critical path of original unit is assigned to vendor 'V2).
- Operations of critical path of U^{OG} and U^{DP} are assigned to distinct vendors.
- Similar operations of both U^{OG} and U^{DP} being assigned to different vendors.

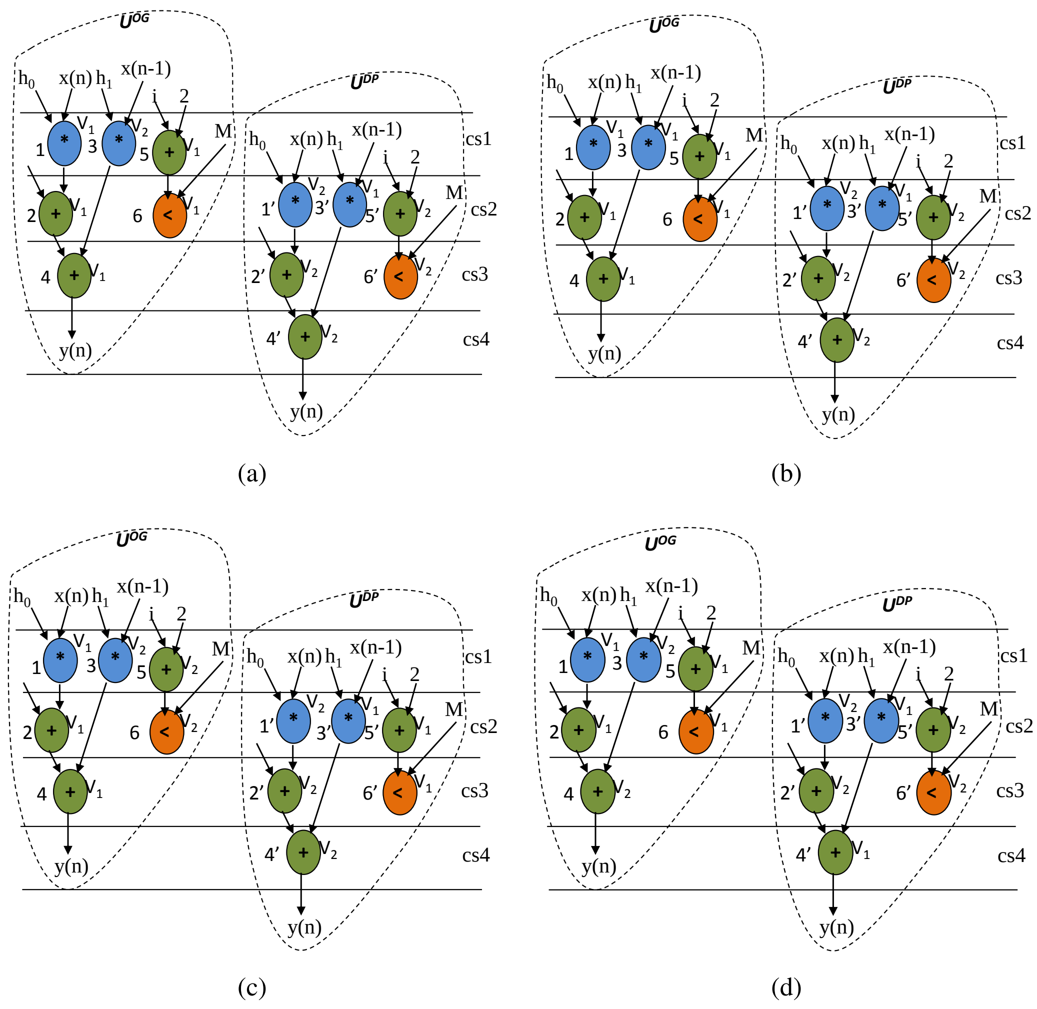


Figure 9.7: Scheduling and Binding of FIR for: (a) X_i = 2(+), 2(*), 2(<), U=2, I=4, $A_v = 00$; L_T^{DMR} = 45080 ns and A_T^{DMR} = 13064 au, (b) X_i = 2(+), 2(*), 2(<), U=2, I=4, $A_v = 01$; L_T^{DMR} = 43080 ns and A_T^{DMR} = 17996 au, (c) X_i = 2(+), 2(*), 2(<), U=2, I=4, $A_v = 10$; L_T^{DMR} = 45080 ns and A_T^{DMR} = 13064 au, (d) X_i = 2(+), 2(*), 2(<), U=2, I=4, $A_v = 11$; L_T^{DMR} = 45070 ns and A_T^{DMR} = 15096 au

Rule 4: vendor allocation procedure for $A_v = 11$

- Alternate vendor assignment to operations belonging to subsequent unrolled nested loop iterations within a unit(Example, in Fig. 9.7(d), all operations within first loop iteration of original unit is assigned to vendor ‘V1’ and all operations within second loop iteration of original unit is assigned to vendor ‘V2).

- Similar operations of both U^{OG} and U^{DP} being assigned to different vendors.

9.3.5 Example of designing a Trojan secured single loop application

Now consider the Trojan secured DMR CDFG of FIR is designed based on candidate solution $X_i = 2(+), 2(*), 1(<), U = 2, A_v = 00$ (illustrated in Fig. 9.7(a)). This indicates the DMR CDFG of FIR filter is unrolled two times, scheduled based on two adders, two multipliers & one comparator, and vendor hardware allocation is performed using mode '00' . The corresponding execution delay and area of the design is: L_T^{DMT} = '45080 ns'and A_T^{DMR}= '13064 au'. As shown in Fig. 9.7(a) based on vendor allocation rule in mode 00, operations of U^{OG} are allocated to vendor in an alternative way in a CS. For example, opn 1 and opn 3 of U^{OG} are allocated alternatively to hardware of 'V1' and 'V2' in control step 1. Further, opn 1' and opn 3' of U^{DP} are allocated to hardware of 'V2' and 'V1' respectively.

If the same DMR CDFG of FIR filter is designed based on candidate solution $X_i = 2(+), 2(*), 1(<), U = 2, A_v = 01$ (illustrated in Fig. 9.7(b)), the corresponding execution delay and area of the design is: L_T^{DMT} = '43080 ns'and A_T^{DMR}= '17996 au'. As shown in Fig. 9.7(b) based on vendor allocation rule in mode 01, all operations of UOG are allocated to hardware of V1 and all operations of UDP are allocated to hardware of 'V2'.

If the same DMR CDFG of FIR filter is designed based on candidate solution $X_i = 2(+), 2(*), 1(<), U = 2, A_v = 10$ (illustrated in Fig. 9.7(c)), the corresponding execution delay and area of the design is: L_T^{DMT} = '45080 nsand A_T^{DMR}= '13064 au'. As shown in Fig. 9.7(c) based on vendor allocation rule in mode 10, all the critical path operations i.e. opn 1, opn 2 and opn 4 of U^{OG} (marked in dotted circle) are allocated to hardware of 'V1' and all non-critical path operations i.e. opn 3, opn 5 and opn 6 of U^{OG} are allocated to hardware of 'V2'. Hardware of distinct vendor is allocated in U^{DP}.

If the same DMR CDFG of FIR filter is designed based on candidate solution $X_i = 2(+), 2(*), 1(<), U = 2, A_v = 11$ (illustrated in Fig. 9.7(d)), the corresponding execution delay and area of the design is: L_T^{DMT} = '45070 nsand A_T^{DMR}= '15096 au. As shown in Fig. 9.7(d) based on vendor allocation rule in mode 11, alternate vendor allocation is performed to the operations belonging to subsequent unrolled loop iterations, for example all operations of first unrolled loop i.e. opn 1, opn 2, opn 5 and opn 6 of U^{OG} are allocated to hardware of 'V1' and all operations of second unrolled loop i.e. opn 3, and opn. 4 of

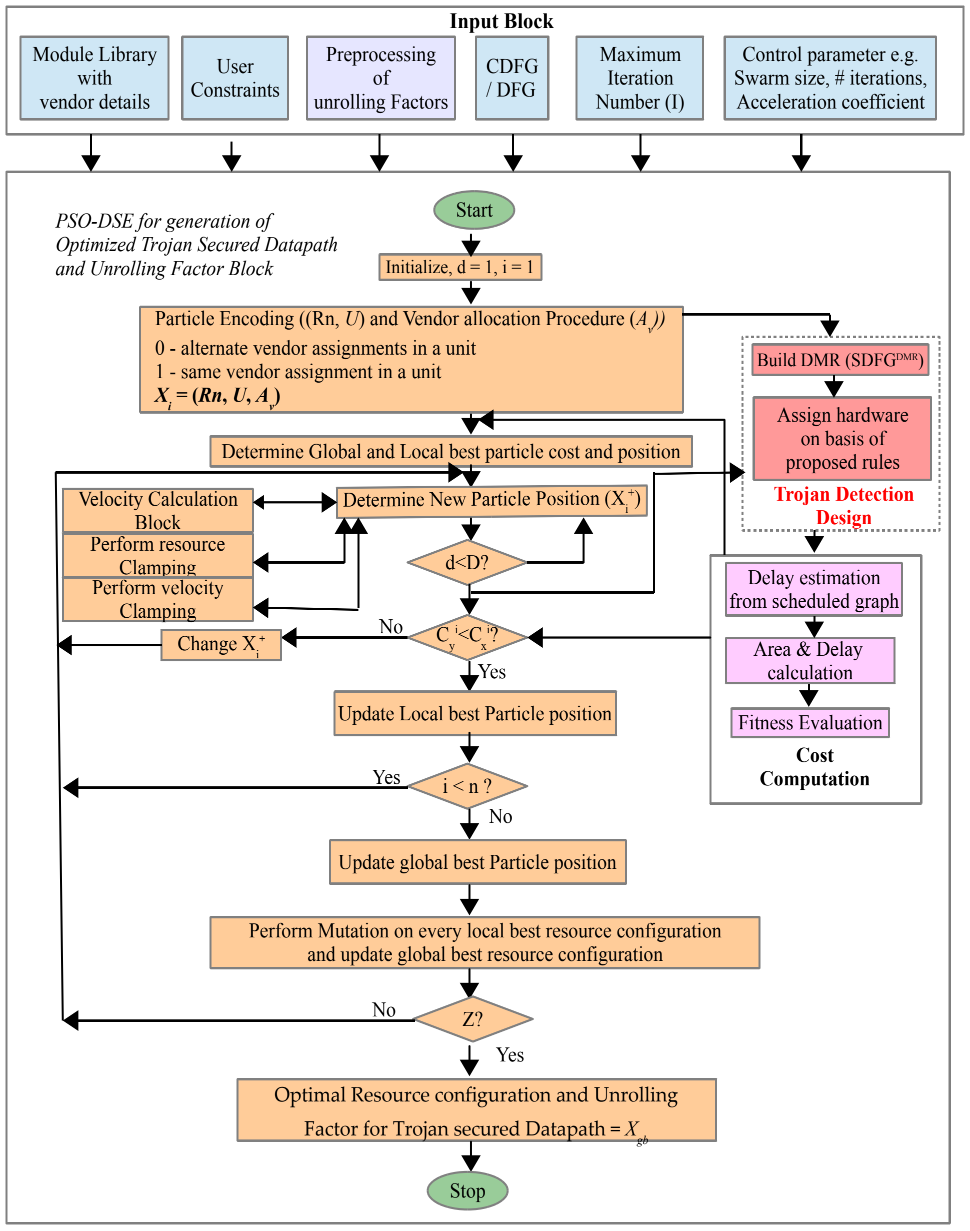
U^{OG} are allocated to hardware of 'V2'. Hardware of distinct vendor is allocated in U^{DP}.

9.4 Generating Low-Cost Trojan Secured Design through PSO-DSE

This section discuss the motivate and the process to generate a low-cost design for Trojan secured design through PSO-DSE approach. Module library, behavioral description of CDFG and predefined user parametric constraints for area and delay, control parameters of PSO (such as inertia weight, acceleration coefficients, and swarm size 'p'), maximum iteration count for the CDFG and the preprocessing algorithm for unrolling factors are provided as inputs to the proposed DSE process (as shown in Fig. 9.8).

9.4.1 Justification of PSO

The module library of an HLS tool consist of maximum available resources of each resource type (such as adder, subtractor, multiplier, comparator, mux, registers etc.). However, among all the possible combinations of resources identifying the low-cost resource configuration for the corresponding application is a challenging and time consuming process. Additionally, as the proposed approach is capable of handling the single and nested loop-based applications, therefore loop unrolling factor plays an important significance in dictating the final area and delay of a design. Finally, the Trojan secured HLS demands distinct vendor allocation to hardwares of operations of original and duplicate units. An efficient vendor allocation procedure for hardware in DMR system also affects the final area-delay of the solution. Therefore, during the design of a Trojan security aware schedule for CDFGs, simultaneously considering the effects of loop unrolling and vendor allocation procedure on its area-delay trade-off is equally critical which increase the search space exponentially. The detection process of hardware Trojan during HLS mandates additional hardware, which upon deployment may not abide by the user area constraint provided. Further, incorporating additional logic for Trojan detection during HLS also results in extra delay for processing output, which again may not abide by the user delay constraints spec-



Input Block
Module Library with vendor details
User Constraints
Preprocessing of unrolling Factors
CDFG / DFG
Maximum Iteration Number (I)
Control parameter e.g. Swarm size, # iterations, Acceleration coefficient
PSO-DSE for generation of Optimized Trojan Secured Datapath and Unrolling Factor Block
Start
Initialize, d = 1, i = 1
Particle Encoding ((Rn, U) and Vendor allocation Procedure (A_v))
0 - alternate vendor assignments in a unit
1 - same vendor assignment in a unit
X_i = (Rn, U, A_v)
Build DMR (SDFG^DMR)
Assign hardware on basis of proposed rules
Trojan Detection Design
Determine Global and Local best particle cost and position
Velocity Calculation Block
Determine New Particle Position (X_i^+)
Perform resource Clamping
Perform velocity Clamping
d<D?
Delay estimation from scheduled graph
Area & Delay calculation
Fitness Evaluation
Cost Computation
No
Change X_i^+
C_y^i<C_x^i?
Yes
Update Local best Particle position
Yes
i < n ?
No
Update global best Particle position
Perform Mutation on every local best resource configuration and update global best resource configuration
No
Z?
Yes
Optimal Resource configuration and Unrolling Factor for Trojan secured Datapath = X_gb
Stop

Figure 9.8: Proposed design flow of low-cost Trojan secured DMR schedule

ified. Therefore, it becomes mandatory to consider the effect of extra delay and hardware cost during DSE in HLS. PSO is a meta-heuristic search methodology where the particles move through a multidimensional search space and it also provides enough options to perform guided/adaptive searching while preserving the exploration-exploitation balance. Therefore PSO based DSE is perform to explore the low-cost Trojan secured design.

9.4.2 Initialization of Particle

In this current proposed approach, a candidate design solution is represented through a particle. The position of a particle in PSO as defined earlier in section 9.3.1. The particle position 'X_i' is given as follows:

$$X_i = (N(R_1), N(R_2), ..N(R_D), U, A_v)\,. \tag{9.8}$$

The particles are uniformly distributed over the design space and are initialized as follows:

$$X_1 = (min(R_1), min(R_2), ..min(R_D), min(U), 00) \tag{9.9}$$

$$X_2 = (max(R_1), max(R_2), ..max(R_D), max(U), 10) \tag{9.10}$$

$$X_3 = \left(\frac{min(R_1) + max(R_1)}{2}, ... \frac{min(R_D) + max(R_D)}{2},\ \frac{min(U) + max(U)}{2}, 01 \right). \tag{9.11}$$

However, rest of the particles ($X_4 ... X_n$) are initialized by the following:

$$\begin{aligned} X_n = \Bigg(& \frac{min(R_1) + max(R_1)}{2} \pm \alpha, \frac{min(R_2) + max(R_2)}{2} \pm \alpha,\ ... \\ & \frac{min(R_d) + max(R_d)}{2} \pm \alpha, \frac{min(U) + max(U)}{2} \pm \alpha, random(A_v) \Bigg). \end{aligned} \tag{9.12}$$

where, α is a random integer between *min* and *max* of particular resource type or unrolling factor.

9.4.3 Particle Movement using Velocity

In PSO-DSE, each dimension (d) of a particle position X_i (except the last dimension) is updated using the following function [55]:

$$R^{+}_{d_i} = R_{d_i} + V^{+}_{d_i}. \tag{9.13}$$

The variable $V_{d_i}^{+}$ is updated by eqn. 9.14 as follows:

$$V_{d_i}^{+} = \omega V_{d_i} + b_1 r_1 [R_{d_{lb_i}} - R_{d_i}] + b_2 r_2 [R_{d_{gb}} - R_{d_i}] \tag{9.14}$$

The local best (X_{lb_i}) and global best (X_{gb}) particle positions are updated using eqn. 9.15 and 9.16:

$$X_{lb_i} = R_{1_{lb_i}}, ...R_{D-1_{lb_i}}, U \tag{9.15}$$

$$X_{gb} = R_{1_{gb}}, ...R_{D-1_{gb}}, U \tag{9.16}$$

Finally, a low-cost DMR schedule ($SDFG^{DMR}$) (with distinct vendor assignment rule to detect the hardware Trojan), is generated corresponding to a particle positions/configurations.

9.4.4 Velocity clamping

The velocity clamping adopted from [55] is performed, when a particle's exploration drift ($V_{d_i}^{+}$) crosses the $\pm V_{d_i}^{max}$ as follows:

$$V_{d_i}^{+} = \begin{cases} +V_{d_i}^{max} & if \quad V_{d_i}^{+} > +V_{d_i}^{max} \\ -V_{d_i}^{max} & if \quad V_{d_i}^{+} < -V_{d_i}^{max} \\ V_{d_i}^{+} & else \end{cases} \tag{9.17}$$

In the above expression, the value of $\pm V_{d_i}^{max}$ is the following:

$$V_{d_i}^{+} = \left(\pm \frac{max(N(R_d)) - min(N(R_d))}{2} \right). \tag{9.18}$$

9.4.5 Terminating condition

The proposed methodology terminates when a) the maximum number of iterations exceeds 100, or when no improvement is visible in X_{gb} over 'δ' number of iterations(where δ =10).

Note: The corresponding experimental results of the proposed methodology is explained in Chapter 10 Section 10.6.

9.5 Summary

This chapter presents a novel low-cost methodology for optimized Trojan secured scheduling at the behavioral level for control data flow graphs (single and nested loop) representing DSP kernels. It is capable of providing secured information processing during high-level synthesis. More explicitly, the novel contributions of this chapter are as follows: (a) a model for execution delay determination of a DMR system for Trojan secured CDFG (b) particle encoding scheme that concurrently explores schedule configuration, unrolling factor and vendor allocation procedure (c) methodology for area-delay trade-off using PSO during optimization of secured schedule. Results indicated improvement in the quality of final solution obtained compared to a similar related work [46].

Chapter 10

Results and Analysis

This chapter describes the complete experimental results and detailed analysis of all the proposed methodologies explained in the previous chapters. This chapter is divided into seven sections, where each section presents the results of the corresponding methodology. All the experimental results are included in one chapter because the readers can observe the improvements/changes in the results when better or equivalent defense mechanisms are used. (*NOTE: All the abbreviations and taxonomy are listed in 'Acronyms' and 'Nomenclature' section respectively.*)

10.1 Experimental Results: Single-Phase IP Core Watermarking of CE Systems

This section explains the experimental results of the proposed single-phase IP core watermarking approach of CE systems explained in Chapter 3 and the improvements obtained compared to two similar approaches [30, 51]. The proposed approach [30] and [51] is implemented in java and executed on Intel Core-i5-3210 M CPU with 4GB DDR3 memory at 2.5 GHz. 15 nm technology scale based on NanGate library [37] is used to determine the area and latency of the benchmarks. The module library information of major function unit in terms of area and delay is shown in Table 10.1. The proposed approach is tested and verified on several benchmarks, such as Differential Equation, Finite Impulse Response Filter, Fast Fourier Transformation, Autoregressive Filter, Mesa Horner and Infinite Impulse Response Butterworth Filter.

Table 10.1: Used module library

Major functional unit	**Size**	**Area** (μm^2)	**Latency** **(ps)**
Adder	16 bit	18.8744	66.2428
Substractor	16 bit	22.0192	79.3910
Multiplier	16 bit	75.4976	264.9712
Register	16 bit	0.7864	25.9108
Mux	2:1	0.6390	12.7494
Demux	1:2	0.7373	22.4165

10.1.1 Analysis of design cost

The design cost of the proposed approach evaluated in terms of two crucial design parameters i.e. design area and execution latency, is calculated based on the following function [30, 51]:

$$C_f(X_i) = w_1 \frac{L_T}{L_{max}} + w_2 \frac{A_T}{A_{max}} \tag{10.1}$$

The total design area and execution latency of the watermarked design consist of area and latency of functional units, registers, multiplexers and demultiplexers (shown in Table 10.1). To provide equal weightage on both the design parameters the value to w_1 and w_2 is set as 0.5. As the calculated design cost shown in Eq. 10.1 is a hybrid normalized function of design area and execution latency, therefore it is unit less. It does not indicate cost in monetary term.

10.1.2 Comparison of design cost

This section discusses the design cost comparison of the proposed approach w.r.t [30] and [51].

Table 10.2: Comparison of proposed approach with [30] in terms of design area, execution latency and design cost.

Benchmarks	Resource configuration	Design Area (μm^2)		Execution latency (ps)		Design cost	
		Proposed	[30]	Proposed	[30]	Proposed	[30]
Diff Eq.	1(+), 1(-), 2(*), 4(UF)	205.06	226.88	2595.11	2595.11	0.67	0.71
FIR	1(+), 2(*), 2(UF)	63.89	84.54	794.91	794.91	0.44	0.49
FFT	2(+), 1(-), 2(*), 2(UF)	146.32	168.15	4555.09	4555.09	0.51	0.54
ARF	1(+), 2(*)	290.18	312.01	2441.10	2441.10	0.83	0.86
MESA	1(+), 2(*)	161.21	175.76	1332.40	1332.40	0.85	0.89
IIRB	1(+), 2(*)	132.90	154.73	885.40	885.40	0.78	0.84

Table 10.3: Comparison of proposed approach with [51] in terms of design area, execution latency and design cost.

Benchmarks	Resource configuration	Design Area (μm^2)		Execution latency (ps)		Design cost	
		Proposed	[51]	Proposed	[51]	Proposed	[51]
Diff Eq.	1(+), 1(-), 2(*), 4(UF)	205.06	224.53	2595.11	2595.11	0.67	0.71
FIR	1(+), 2(*), 2(UF)	63.89	84.54	794.91	794.91	0.44	0.49
FFT	2(+), 1(-), 2(*), 2(UF)	146.32	165.79	4555.09	4555.09	0.51	0.54
ARF	1(+), 2(*)	290.18	312.01	2441.10	2441.10	0.83	0.86
MESA	1(+), 2(*)	161.21	173.41	1332.40	1332.40	0.85	0.88
IIRB	1(+), 2(*)	132.90	154.73	885.40	885.40	0.78	0.84

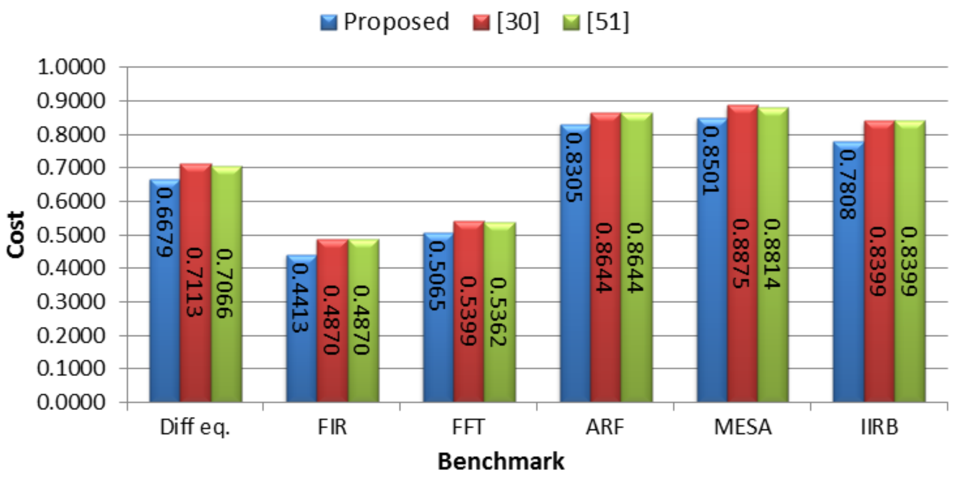


Figure 10.1: Graphical representation of design cost comparison between proposed single-phase watermark approach, [30] and [51].

For comparison purpose a 60-digit signature is implanted in the design through the proposed approach [30] and [51]. Same unrolling factor and resource constraints for all the three approaches are used to generate the watermarked IP core design. The proposed ap-

Table 10.4: Comparison of proposed approach with [51] in terms of design area, execution latency and design cost.

Benchmarks	**# of storage hardware (register)**			
	Pre-watermark	**After proposed watermark**	**After [30]**	**After [51]**
Diff Eq.	12	12	15	14
FIR	3	3	6	6
FFT	4	4	7	6
ARF	8	8	11	11
MESA	4	4	7	6
IIRB	5	5	8	8

proach when compared with [30] and [51] gave substantially better result in terms of design cost. Table 10.2 and Table 10.3 depict the comparison of the proposed approach with [30] and [51] respectively in terms of design area, execution latency and design cost. The reason being the proposed approach yield zero hardware (register) overhead while preserving the same latency for all the tested benchmarks. This is because the proposed approach embeds watermarking constraints through priority based scheduling transformations, thus incurring zero hardware overhead and may also incur negligible latency overhead. On the contrary, both [30] and [51] embeds watermarking constraints during register allocation utilizing more hardware area due to overhead in total register count. The graphical representation of cost comparison between proposed approach, [30] and [51] is shown in Fig. 10.1.

A comparison drawn between pre-watermarked design, proposed watermark design, watermark design obtained by [30] and [51] in terms of storage hardware (register) is reported in Table 10.4. As shown in the table, the proposed approach does not add any extra registers, hence the total register count remains same as pre-watermark design. Since both [30] and [51] embeds watermark during register allocation in HLS thus increases total register count of the design.

10.2 Experimental Results: Triple-Phase IP Core Watermarking of CE Systems

This section discusses the experimental results of the proposed triple-phase IP core watermarking approach of CE systems explained in Chapter 4 and also discusses the improvements over two similar approaches [30, 51]. The original design before embedding any constraint is termed as baseline design. Here, the baseline design, proposed approach [30] and [51] are implemented in java and executed on AMD A8- 4500M APU with 4 GB DDR3 memory at 1.9GHz. 15 nm technology scale based on NanGate library [37] is used to determine the area and latency of the benchmarks. The proposed approach is capable of handling any medium to large size application ranging from 40 components (e.g. EWF) to excess of 100 components (e.g. JPEG IDCT). It is tested and verified on several benchmarks like ARF, DCT, DWT, EWF, IDCT, MPEG motion vector and JPEG. The proposed approach is highly robust for complicated designs such as JPEG IDCT, MPEG MV etc. This section is divided into three sub-sections: (i) evaluation of robustness, (ii) evaluation of tamper-tolerance ability and (iii) evaluation of design cost of proposed triple-phase watermark.

10.2.1 Evaluation of robustness

The strength of robustness of a watermark design is evaluated through probability of coincidence metric. It calculates the probability of generating same design before and after implanting watermark, thus indicating the strength of watermark. Lower Pc value indicates a higher strength of watermark. The metric is partially adopted from [30] and can be represent as:

$$P_c = \left(1 - \frac{1}{c * \prod_{i=1}^{D} N(R_i)}\right)^w \tag{10.2}$$

Table10.5 reports the comparative study of the probability of coincidence between proposed approach, [30] and [51] for signature size 80. It can be observed from the table that the proposed approach achieves $3.2 * 10^{27}$ times lower P_c value in average as compared

Table 10.5: Comparison of probability of coincidence between proposed, [30] and [51] for W=80

Benchmarks	**# of register before watermark**	P_c			**# of times lower P_c of proposed than [30] and [51]**
		Proposed	**[30]**	**[51]**	
ARF	8	$3.3 * 10^{-27}$	$2.2 * 10^{-5}$	$2.2 * 10^{-5}$	$6.9 * 10^{21}$
DCT	8	$3.7 * 10^{-21}$	$2.2 * 10^{-5}$	$2.2 * 10^{-5}$	$6.1 * 10^{15}$
DWT	5	$8.3 * 10^{-35}$	$1.7 * 10^{-8}$	$1.7 * 10^{-8}$	$2.1 * 10^{26}$
EWF	4	$6.8 * 10^{-39}$	$1.0 * 10^{-10}$	$1.0 * 10^{-10}$	$1.5 * 10^{28}$
IDCT	8	$3.3 * 10^{-27}$	$2.2 * 10^{-5}$	$2.2 * 10^{-5}$	$6.9 * 10^{21}$
MPEG MV	14	$3.8 * 10^{-31}$	$2.6 * 10^{-3}$	$2.6 * 10^{-3}$	$6.9 * 10^{27}$
JPEG IDCT	12	$1.9 * 10^{-23}$	$9.4 * 10^{-4}$	$9.4 * 10^{-4}$	$5.0 * 10^{19}$

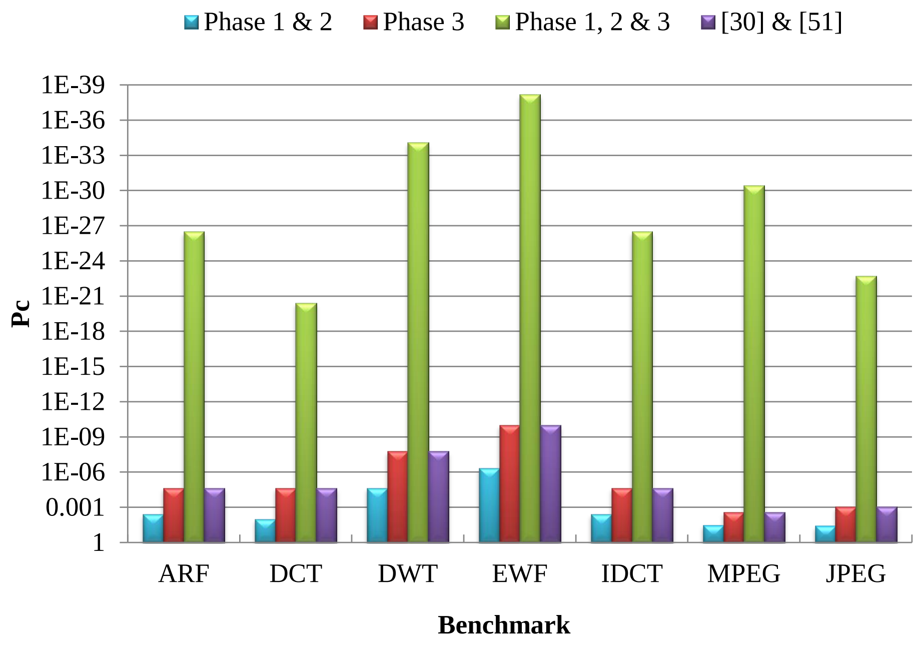


Figure 10.2: Graphical representation of P_c comparison between proposed approach, [30] and [51].

to [30] and [51]. This indicates the higher robustness of the watermarked design. Low P_c value is desirable which is achievable by incrementing the watermark size. For example, the probability of coincidence of MPEG benchmark is $7.7{*}10^{-18}$, $1.5{*}10^{-23}$ and $3.8{*}10^{-31}$ for watermark sizes 45, 60, 80 respectively. Though lower P_c value is desirable but not at the cost of higher design overhead. However, too big a size of watermarking constraints may result in register overhead thereby increasing design cost. It is observed through experiment that the desirable maximum range of signature size should be 80. Fig.10.2 shows the

Table 10.6: Comparison of total number of possible signature combinations between proposed, [30] and [51]

Signature size (digits)	**# of possible signature combinations**			**# of times higher tamper tolerance ability of proposed than [30] & [51]**	
	Proposed	**[30]**	**[51]**	**[30]**	**[51]**
15	$4.8 * 10^{12}$	32768	$10.7 * 10^{8}$	$14.5 * 10^{7}$	4421
30	$2.3 * 10^{25}$	$1.1 * 10^{9}$	$1.2 * 10^{18}$	$2.1 * 10^{16}$	$19.5 * 10^{6}$
45	$1.1 * 10^{38}$	$3.5 * 10^{13}$	$1.2 * 10^{27}$	$3.0 * 10^{24}$	$8.6 * 10^{10}$
60	$5.1 * 10^{50}$	$1.2 * 10^{18}$	$1.3 * 10^{36}$	$4.4 * 10^{32}$	$3.8 * 10^{14}$
80	$4.1 * 10^{67}$	$1.2 * 10^{24}$	$1.5 * 10^{48}$	$3.4 * 10^{43}$	$2.8 * 10^{19}$

comparison of P_c for each watermarking phases (1st, 2nd and 3rd) of proposed approach with [30] and [51]. It can be observed that the robustness of the proposed approach with the combination of three watermarking phases is significantly higher than [30] and [51].

10.2.2 Evaluation of tamper-tolerance ability

A watermark is considered to be highly tamper tolerant if finding the correct signature from a design is extremely complex and time-consuming. For a large signature size with multiple signature variables, the possible signature combination will be huge. Therefore, in such cases identifying the correct watermark through brute-force search is time consuming and cost expending process. The total number of possible signature combinations can be calculated by the following expression:

$$T_t = v^w \tag{10.3}$$

Table 10.6 reports the comparison of the total number of possible signature combinations for different watermark size (i.e. 15, 30, 45, 60 and 80) between the proposed approach, [30] and [51]. It can be observed from the table that the proposed approach achieves higher T_t value for all watermark size compared to [30] and [51]. Ensuring higher tamper tolerance ability.

Table 10.7: Comparison of design area, execution latency and design cost between proposed and baseline design

Benchmarks	**Resource Configuration**	**Area (μm^2)**		**Latency (ps)**		**Cost**		**Cost overhead w.r.t. baseline**
		Baseline	**Proposed**	**Baseline**	**Proposed**	**Baseline**	**Proposed**	
ARF	5(+), 3(*)	191.10	209.19	2.67	3.11	0.77	0.87	12.98
DCT	6(+), 3(*)	250.87	263.45	3.95	4.19	0.80	0.84	5.00
DWT	2(+), 4(*)	162.79	165.94	1.98	2.08	0.78	0.81	3.85
EWF	3(+), 2(*)	184.81	197.39	3.24	3.82	0.85	0.95	11.76
IDCT	5(+), 3(*)	246.15	253.23	3.77	4.16	0.78	0.83	6.41
MPEG	3(+), 8(*)	280.76	287.05	2.44	2.59	0.73	0.76	4.11
JPEG	5(+), 5(*)	747.90	756.55	14.90	15.92	0.72	0.76	5.56

Table 10.8: Comparison of design area, execution latency and design cost between proposed, [30] and [51]

Benchmarks	**Area (μm^2)**			**Latency (ps)**			**Cost**		
	Proposed	**[30]**	**[51]**	**Proposed**	**[30]**	**[51]**	**Proposed**	**[30]**	**[51]**
ARF	209.19	225.71	223.35	3.11	3.11	3.11	0.87	0.92	0.90
DCT	263.45	290.98	288.62	4.19	4.51	4.51	0.84	0.94	0.92
DWT	165.94	182.37	180.01	2.08	2.43	2.43	0.81	0.93	0.92
EWF	197.39	209.19	204.47	3.82	3.89	3.89	0.95	0.99	0.98
IDCT	253.23	280.96	278.40	4.16	4.34	4.34	0.83	0.91	0.89
MPEG	287.05	309.85	309.85	2.59	2.77	2.77	0.76	0.81	0.81
JPEG	756.55	783.29	783.29	15.92	16.52	16.52	0.76	0.79	0.79

10.2.3 Evaluation of design cost

The design cost of the proposed approach evaluated in terms of two crucial design parameters i.e., design area and execution latency are calculated based on Eq.10.1. Table 10.7 reports the comparison of total hardware area, execution latency and design cost between baseline design and the proposed approach. Since the proposed watermarking approach (containing 7 variable signatures) may impose area overhead nominally compared to an un-protected design, thus the power overhead of the design may increase trivially. Therefore, system reliability as function of power is marginally affected sometimes. However, the robustness of the system due to triple phase and 7 variables is guaranteed to increase manifold.

Table 10.8 reports the comparison of total hardware area, execution latency and design cost between the proposed approach [30] and [51]. It can be observed from the table that

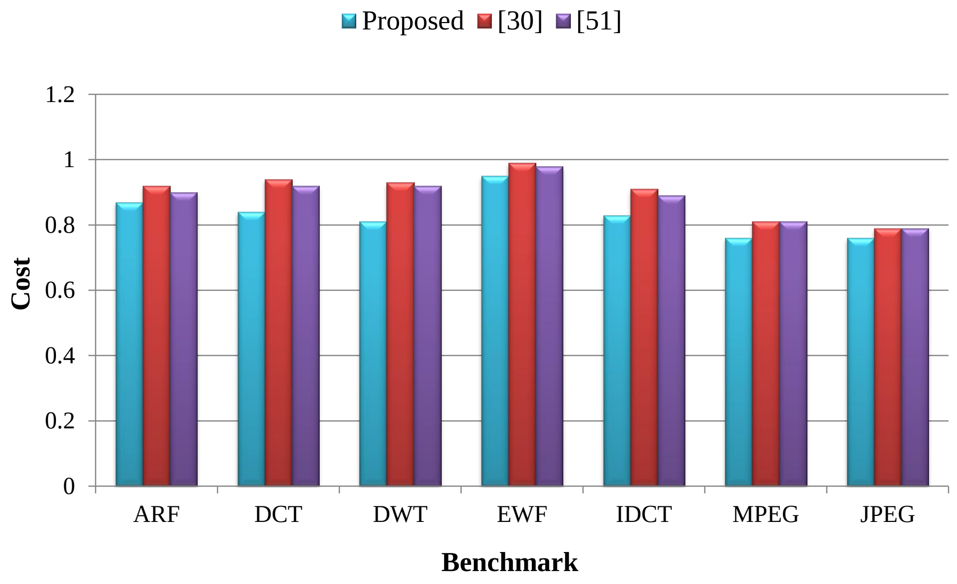


Figure 10.3: Graphical representation of design cost comparison between proposed triple-phase watermark approach, [30] and [51].

the proposed approach instead of providing strong robustness achieves better design cost than the two other approaches. Fig.10.3 shows the comparison of design cost between the proposed approach, [30] and [51]. For all the tested benchmarks, the average reduction % in hardware area, execution latency and design cost are 6.65%, 5.37% and 6.25% respectively as compared to [51] and the average reduction % in hardware area, execution latency and design cost as 7.44%, 5.37% and 7.38% respectively compared to [30].

Table 10.9 reports the comparison of total storage hardware used in the baseline, proposed design, [30] and [51]. Table 10.9 also reports the complexity of the watermarking process in terms of implanting time. As evident from Table 10.9, the proposed approach incurs slightly higher watermark implanting time than [30] and [51] due to triple phase watermark insertion. However, at the cost of nominal increase in runtime, this approach offers stronger robustness (proof of ownership) and greater tamper tolerance ability than other similar approaches.

Despite implanting watermark in three different phases the proposed approach is capable of achieving significant reduction in design cost than other approaches due to the following reasons:

Table 10.9: Comparison of storage hardware and watermark implanting time between proposed, [30] and [51]

Benchmark	# of storage hardware used				Watermark implanting time (ms)		
	Baseline	Proposed	[30]	[51]	Proposed	[30]	[51]
ARF	8	8	9	8	76	24	46
DCT	8	8	9	8	88	29	69
DWT	5	6	7	6	26	15	17
EWF	4	4	6	4	64	28	51
IDCT	8	9	10	9	87	31	62
MPEG	14	14	14	14	80	34	65
JPEG	12	12	12	12	138	68	109

1. In the proposed approach, among all the signature digits few digits are implanted in register allocation phase while other digits are implanted the other two phases (scheduling and FU allocation). On the contrary, in [30] and [51] all the signature digits are implanted during register allocation phase. Therefore, for an 80 digits signature, reported for embedding watermark, the proposed approach uses register allocation based watermark (i, I, T, !) partially, while the remainder signature digits are embedded through hardware allocation and scheduling (other two phases). On the contrary, [30] and [51] employ register allocation based watermark for the entire 80 digit signature. [30] and [51] incur higher register overhead in most cases, thus consumes more hardware area than the proposed approach.

2. The proposed approach uses multi-vendor concept during hardware allocation compared to single vendor hardware allocation in [30] and [51], thus the proposed approach is more likely to have optimized (minimized) area and delay than [30] and [51].

3. The proposed approach implants scheduling constraints (γ digits) in the non-critical path of the design resulting in occasional or zero latency overhead in the watermarked design. Finally, this contributes to achieve lower design cost for the proposed approach.

Table 10.10: Comparison of vendor allocation between proposed, [30] and [51]

Benchmark	**Total operations in the application**	**Component allocation based on multi-vendor (proposed)**		**Component allocation based on single-vendor ([30] & [51])**	**Proposed approach (impact on latency)**	
		Vendor U1	**Vendor U2**	**Vendor U1**	**Critical path length (in CS)**	**Non-critical path length after G insertion (in CS)**
ARF	12(+), 16(*)	8(+), 10(*)	4(+), 6(*)	12(+), 16(*)	8	7
DCT	29(+), 13(*)	18(+), 8(*)	11(+), 5(*)	29(+), 13(*)	8	8
DWT	9(+), 8(*)	7(+), 5(*)	2(+), 3(*)	9(+), 8(*)	10	9
EWF	26(+), 8(*)	14(+), 4(*)	12(+), 4(*)	26(+), 8(*)	14	14
IDCT	29(+), 13(*)	17(+), 7(*)	12(+), 6(*)	29(+), 13(*)	6	5
MPEG	14(+), 14(*)	9(+), 7(*)	5(+), 7(*)	14(+), 14(*)	4	4
JPEG	75(+), 37(*)	44(+), 20(*)	31(+), 17(*)	75(+), 37(*)	8	5

Table 10.10 reports the details of component allocation to the operations of the applications for proposed approach, [30] and [51]. As evident from Table 10.10, the proposed approach optimizes the component allocation through multi-vendor concept, where delay of multiplier and adder from vendor U2 is less than delay of multiplier and adder from vendor U1. On the contrary, for [30] and [51] component allocation of all the operations is entirely done through a single vendor U1. The above explanation can be summarized through the following execution latency (delay) models:

$$L_T^{[30][51]} = \sum_{n=1}^{N} Max(T^{A1}), (T^{M1}) \tag{10.4}$$

$$L_T^{proposed} = \sum_{n=1}^{N} Max(T^{A1}), (T^{M1}), (T^{A2}), (T^{M2}) \tag{10.5}$$

In the context of Eq. 10.4 and Eq. 10.5 $L_T^{proposed} < L_T^{[30][51]}$ due to multi-vendor based hybrid component allocation concept in proposed approach.

Table 10.10 also reports comparison between the length of the critical path of an application and the length of the non-critical path after implanting 'γ' digits in terms of number of CS. As evident from the table, the non-critical path length is always lesser or equal to the critical path length for all the tested benchmarks. Therefore, no latency overhead is achieved due to insertion of 'γ' digit.

10.3 Experimental Results: Symmetrical IP Core Watermarking of CE Systems

This section discusses the experimental results of the proposed symmetrical IP core protection approach of CE systems explained in Chapter 5 and comparison with a non-symmetrical approach (as no symmetrical IP core protection approach exists during HLS). Following metrics are used to confirm the robustness and low overhead of the proposed approach:

- Cost of embedding seller watermark and buyer fingerprint in IP core design which signifies quality of the solution.
- Hardware and storage overhead in terms of functional hardware units and extra register required after embedding seller watermark and buyer fingerprint mark.
- Time consumed to create signature.
- Quality of signature in terms of security/strength.
- Impact of large seller signature and buyer signature on hardware area, latency and embedding cost.
- Scalability of the approach to handle a large benchmark.

The result of the proposed approach is obtained for several benchmarks, namely ARF, DCT, IDCT, BPF, FIR, MPEG motion vector and JPEG. This approach is highly robust for complicated designs such as JPEG IDCT, MPEG MV etc. This section is divided into two sub-sections: (i) result of proposed symmetrical protection approach in terms of latency, hardware area, cost and security metric, and (ii) comparison of the proposed symmetrical IP core protection with a non-symmetrical approach.

10.3.1 Result of the proposed approach in terms of design Cost, security and implementation complexity

Table 10.11 shows the result of proposed symmetrical IP core protection approach in terms of hardware area, execution latency and design cost (for F = 30 and W = 30). (Note: module

Table 10.11: Solution of symmetrically protected IP core design through the proposed approach (F = 30 and W = 30)

Benchmarks	**Proposed Solution**	**Total Area** (μm^2)	**Total Latency** (ps)	**Cost**
ARF (nodes =28)	2(+), 4(*)	196.61	2.59	0.8393
DCT (nodes =42)	4(+), 2(*)	223.35	3.80	0.8343
IDCT (nodes =42)	4(+), 2(*)	224.13	3.73	0.8270
BPF (nodes =29)	2(+), 2(*)	202.90	3.77	0.8787
FIR (nodes =23)	4(+), 4(*)	180.09	1.86	0.7526
MPEG (nodes =28)	3(+), 5(*)	224.13	2.38	0.6645
JPEG (nodes = 33)	4(+), 4(*)	724.30	14.24	0.7349

Table 10.12: Probability of coincidence (P_c) as strength of watermark

Benchmarks	**# of registers before fingerprint**	P_c		
		# of watermark constraints (W)		
		W=10	**W=20**	**W=30**
ARF	8	0.26308	0.06921	0.01821
DCT	8	0.26308a	0.06921	0.01821
IDCT	9	0.30795	0.09483	0.0292
BPF	7	0.21406	0.04582	0.00981
FIR	8	0.26308	0.06921	0.01821
MPEG	14	0.47660	0.22715	0.10826
JPEG	12	0.41890	0.17548	0.07351

library comprises resources with respect to 15 nm technology scale adopted from NanGate open source library [37]). The cost of each watermarked-fingerprinted IP core solution is evaluated using Eqn. 5.3.

Table 10.12 reports the probability of coincidence(P_c) for the watermark in the proposed approach. It measures the probability of generating the same colored solution with the watermark signature and indicates the proof of ownership. The function for evaluation of P_c is derived from [30] is defined as:

$$P_c = (1 - 1/c)^F \tag{10.6}$$

As evident from the table 10.12, as the signature fingerprint strength increases, the P_c decreases. So, in order to obtain a stronger proof of ownership, a larger size signature is to be chosen.

Table 10.13 shows the impact of incrementing buyer fingerprint size on hardware area, latency and embedding cost. In this experiment, the buyer fingerprint size is varied to gauge its relative effect on various design metrics. Three different values of fingerprint size is chosen viz. F = 10, F = 20 and F = 30. As evident from the results, the increment of hardware area with increase in fingerprint strength is zero in most cases. This is because in the proposed fingerprint signature encoding no variable except 'z' embeds fingerprint constraint during register allocation phase (which in turn may add register overhead). Thus likelihood of an increase in hardware area with fingerprint size is minimal. On the contrary, the minimal increment of latency with the increase in fingerprint strength can be noted for some benchmarks (either for F = 20 or F = 30). This increment is a result of adding more fingerprint constraints during scheduling phase. However, the increment is very slight as fingerprint constraints are embedded during operation conflict resolution in scheduling. Consequently, minimal increase in cost with increase in fingerprint strength is also noted for some benchmarks.

Table 10.14 shows the impact of incrementing seller watermark size (post- embedding buyer fingerprint) on hardware area, latency and embedding cost. Since seller watermark is embedded after buyer fingerprint, different values of watermark strength are indicated by both W and F. In this experiment, the seller watermark size is varied (by keeping the buyer fingerprint strength at maximum i.e. F = 30) to gauge its relative effect on the aforesaid design metrics. Three different values of watermark size are chosen viz. W = 10 & F = 30, W = 20 & F = 20 and W = 30 & F = 30. As evident from the results, the increment of latency with an increase in watermark strength is zero for all cases. This is because the

Table 10.13: Variation of hardware area, execution latency and design cost with the increment of fingerprint size

Benchmarks	**Fingerprint size (F)**								
	F=10			**F=20**			**F=30**		
	Area (μm^2)	**Latency** (ps)	**Cost**	**Area** (μm^2)	**Latency** (ps)	**Cost**	**Area** (μm^2)	**Latency** (ps)	**Cost**
ARF	195.82	2.47	0.82	195.82	2.52	0.83	195.82	2.59	0.84
DCT	222.56	3.75	0.83	222.56	3.80	0.83	222.56	3.80	0.83
IDCT	222.56	3.72	0.83	222.56	3.73	0.83	223.35	0.3.73	0.83
BPF	202.11	3.74	0.88	202.11	3.74	0.88	202.11	3.77	0.88
FIR	179.31	1.81	0.74	179.31	1.86	0.75	179.31	1.86	0.75
MPEG	224.13	2.36	0.66	224.13	2.36	0.66	224.13	2.38	0.67
JPEG	724.30	14.24	0.74	724.3	14.24	0.74	724.3	14.24	0.74

Table 10.14: Variation of hardware area, execution latency and design cost with the increment of watermark size after implanting fingerprint

Benchmarks	**# of watermark constraints (W) after implanting fingerprint**								
	F=30, W=10			**F=30, W=20**			**F=30, W=30**		
	Area (μm^2)	**Latency** (ps)	**Cost**	**Area** (μm^2)	**Latency** (ps)	**Cost**	**Area** (μm^2)	**Latency** (ps)	**Cost**
ARF	195.82	2.59	0.84	196.61	2.59	0.84	196.61	2.59	0.84
DCT	222.56	3.80	0.83	223.35	3.80	0.83	223.35	3.80	0.83
IDCT	223.35	3.73	0.83	223.35	3.73	0.83	223.35	3.73	0.83
BPF	202.11	3.77	0.88	202.90	3.77	0.88	202.90	3.77	0.88
FIR	179.31	1.86	0.75	180.09	1.86	0.75	180.09	1.86	0.75
MPEG	224.13	2.38	0.67	224.13	2.38	0.67	224.13	2.38	0.67
JPEG	724.30	14.24	0.74	724.30	14.24	0.74	724.30	14.24	0.74

watermark signature encoding does not embed watermarking constraint during scheduling phase (which in turn adds no latency overhead). Thus increase in latency with the increase in watermark size is nil for any size of W (i.e. W= 10, W= 20 and W= 30). On the contrary, the minimal increment of hardware area with increase in watermark strength can be noted for some benchmarks (either for W = 20 or W = 30). This increment is a result of adding more watermarking constraints during register allocation phase. Consequently, minimal increase in cost with increase in watermark strength is also noted for some benchmarks.

Table 10.15: Comparison of proposed symmetrical IP core design with baseline IP design

Benchmarks	Area (μm^2)			Latency (ps)			Cost		
	B	**P**	**(%)**	**B**	**P**	**(%)**	**B**	**P**	**(%)**
ARF	195.82	196.61	0.40	2.46	2.59	5.02	0.82	0.84	2.48
DCT	222.56	223.35	0.35	3.73	3.80	1.84	0.83	0.83	0.95
IDCT	222.56	223.35	0.35	3.72	3.73	0.27	0.83	0.83	0.25
BPF	202.11	202.90	0.39	3.69	3.77	2.12	0.87	0.88	0.97
FIR	179.31	180.09	0.43	1.80	1.86	3.23	0.74	0.75	2.07
MPEG	224.13	224.13	0	2.36	2.38	0.84	0.66	0.67	0.09
JPEG	724.30	724.30	0	14.24	14.24	0	0.74	0.74	0

Table 10.16: Comparison of proposed symmetrical IP core design with IP core design of [30]

Benchmarks	Area (μm^2)			Latency(ps)			Cost		
	[30]	**P**	**(%)**	**[30]**	**P**	**(%)**	**[30]**	**P**	**(%)**
ARF	196.61	196.61	0	2.46	2.59	5.02	0.8187	0.8393	2.45
DCT	223.35	223.35	0	3.73	3.80	1.84	0.8267	0.8343	0.91
IDCT	223.35	223.35	0	3.72	3.73	0.27	0.8248	0.8267	0.23
BPF	202.90	202.90	0	3.69	3.77	2.12	0.8705	0.8787	0.93
FIR	180.09	180.09	0	1.80	1.86	3.23	0.7375	0.7526	2.01
MPEG	224.13	224.13	0	2.36	2.38	0.84	0.6639	0.6645	0.09
JPEG	724.30	724.30	0	14.24	14.24	0	0.7349	0.7349	0

10.3.2 Comparison of the proposed symmetrical IP Core protection approach with a non-symmetrical approach

The proposed approach is compared with the baseline design (with no signature implanted) in terms of hardware area, execution latency and design cost. As evident from table 10.15, the proposed approach provides symmetrical IP protection to both buyer and seller at minimal area overhead (less than 1%) and minimal latency overhead (avg. 1.02 %). Consequently, this approach provides symmetrical IP ownership protection at extremely low-cost overhead as compared to baseline.

As no symmetrical IP core protection methodology exists during high level synthesis, the proposed approach is compared with an non-symmetrical IP core protection approach

Table 10.17: Comparison of proposed symmetrical IP core design with IP core design of [30] and baseline in terms of storage hardware (register) and signature creation time

Benchmarks	**# of storage hardware (register) used**			**storage hardware (register) overhead %**		**signature creation time(ms)**	
	B	**[30]**	**P**	**w.r.t. B**	**w.r.t. [30]**	**[30]**	**P**
ARF	8	9	9	0.11	0	17	19
DCT	8	9	9	0.11	0	13	17
IDCT	8	9	9	0.11	0	14	17
BPF	7	8	8	0.125	0	11	14
FIR	8	9	9	0.11	0	31	37
MPEG	14	14	14	0	0	14	16
JPEG	12	12	12	0	0	61	70

[30] (embeds only seller watermark) in terms of hardware area, execution latency and design cost. As evident from table 10.16 in proposed approach, it provides symmetrical IP protection to both seller and buyer at zero area overhead (0%) and minimal latency overhead (avg. 1.02 %) as compared to [30]. Consequently, the proposed approach provides symmetrical IP protection at extremely low cost overhead (avg. 1.93 %) compared to unsymmetrical protection approach [30].

Table 10.17 reports the comparison of proposed symmetrical protection approach with non-symmetrical approach [30] and baseline in terms of storage hardware (registers). As evident from the results, the storage hardware overhead of proposed symmetrical approach compared to non-symmetrical approach (only seller watermark and no buyer fingerprint) [30] is zero. Further, the storage hardware overhead of proposed symmetrical protection approach with respect to baseline is minimal (less than 0.2%). As evident, the proposed approach provides complete IP core protection to both the entities at minimal hardware area (overall less than 1%), latency (overall less than 1.1%) and cost (overall less than 2%) overhead.

Table 10.17 also draws a comparison of proposed symmetrical protection approach with [30] in terms of signature creation time. For the proposed approach, signature creation time includes creation time of seller watermark and buyer fingerprint while only seller watermark for [30]. As evident from the results, the proposed approach while embedding

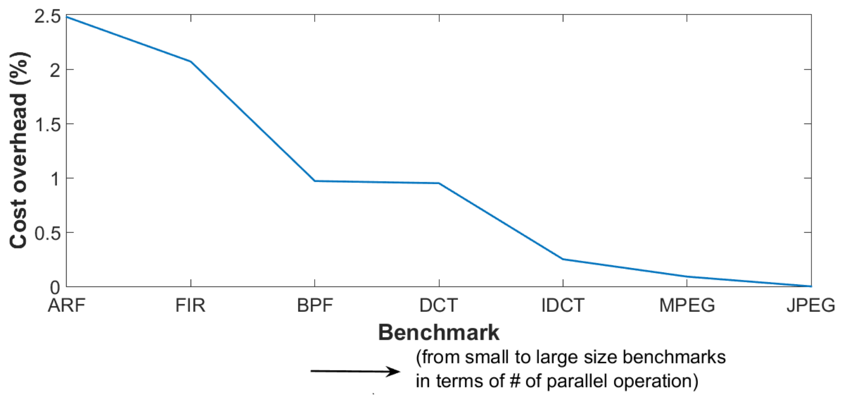


Figure 10.4: Variation of embedded cost overhead (%) for different benchmark sizes.(Note: For the proposed approach embedding cost overhead (%) decreases with the increment in parallelism of different application)

both seller multi-variable watermark and buyer fingerprint, nominally increases the runtime overhead as compared to [30] (that only embeds seller dual-variable watermark). This trend is observed for all size benchmarks, thus demonstrating the scalability of the proposed approach.

Additionally, Fig. 10.4 shows the variation of cost overhead obtained for various benchmarks. As evident, with the increase in size (parallelism i.e. number of parallel operations) of the benchmarks, cost overhead % sharply decreases. In other words, cost overhead of proposed approach is highest for ARF which comprises minimum number of parallel operations (parallelism possible) while is least for JPEG which comprises maximum number of parallel operations (parallelism possible). This is possible because the proposed approach embeds fingerprint signature during operation conflict resolution in scheduling which occurs more often for bigger applications (comprising of large number of parallel operations) working under a resource constraint. In that context, the proposed approach offers lower (negligible) overhead for larger applications which demonstrates the scalability of the proposed approach.

10.4 Experimental Results: Multi-Stage Structural Obfuscation to Secure IP Core of CE Systems

This section discusses the experimental results of the proposed multi-stage structural obfuscation approach explained in Chapter 6 and the improvement obtained when compared to a single-stage structural obfuscation approach [32]. The proposed approach, original non-obfuscated design and [32] are implemented in Java 8 and executed on a computing platform with 4GB DDR3 primary memory and processor frequency of 3.20 GHz. A 15nm technology scale based on NanGate is used to evaluate both the hardware area and execution latency of the IP design [37]. During PSO-DSE process both ϕ_1 and ϕ_2 are kept 0.5 to provide equal preferences as both silicon area and latency are equally essential for a DSP design of a CE device. Further, following optimal settings from [53] are used for PSO framework (refer Eqn. 6.9): ω= linearly decreasing between 0.9 to 0.1; b_1 and b_2 = 2; r_1 and r_2 = 1; $Imax$ = 100 or δ = 10 as stopping criterion; particle size p = 3 or 5 or 7.

The comparison of original non-obfuscated design and proposed multi stage obfuscated design w.r.t. the motivational example shown in Fig. 6.3 in terms of register transfer level and Gate level is reported in Table 10.18. Further, the result of the proposed approach is tested and verified on several benchmarks, such as ARF, Differential, Auto Correlation, DHMC, Adaptive filter (NC & LMS) and FIR. This section is divided into two sub-sections: (i) result of proposed multi-stage structural obfuscation approach in terms of execution latency, hardware area, cost and security metric, and (ii) comparison of the proposed approach with the baseline and [32].

10.4.1 Result of the proposed multi-stage obfuscation approach

Table 10.19 reports the results of low-cost multi-stage structural obfuscation approach in terms of hardware area, execution latency and design cost for different particle size. As evidence from the table, it can be observed except Auto Correlation benchmark no improvement in quality of solution is found with the increment of particle size. However, for Auto Correlation benchmark design cost is -0.4647 for particle size 5 which is better than

Table 10.18: Module and Gate level comparison of original non-obfuscated design and proposed multi stage obfuscated design w.r.t. the motivational example

Types of elements	**Module (RT) level**		**Gate level**		
	Original non-obfuscated design	**Proposed obfuscated design**	**Original non-obfuscated design**	**Proposed obfuscated des ign**	**Gates affected**
Functional units	3 adders, 4 multipliers	3 adders, 1 multiplier	1264	496	768
Switching units	Six 4:1, four 2:1	Eight 8:1	1408	3584	2176
Storage units	10 for input, 1 for output	18 for input, 2 for output	704	1280	576
Delay elements	8	13	512	832	320
Total	-	-	**3888**	**6192**	**3840**

-0.4556, obtained for particle size = 3; though design cost remain same for particle size = 5 and 7. Further, as all the benchmarks satisfy the user provided area-delay constraints, the obfuscated design cost achieved is less than 0 (refer to Eqn. 6.3). The design cost which indicates combined normalized value of crucial design parameters (such as hardware area, execution delay) is unit less as it is a normalized metric.

The robustness of a design obfuscated through the proposed multi-stage obfuscation technique is measured through a metric called power of obfuscation. It calculates the structural mismatch between the original design and the obfuscated design. It is a normalized metric value range between 0 to 1, where 1 indicates highest robustness and 0 indicates lowest robustness of a design. The power of obfuscation is evaluated using the following expression:

$$P_i^{obf} = \frac{n_i}{n_i^T} \tag{10.7}$$

$$P^{obf} = \frac{\sum_{i=1}^{N(HLT)} P_i^{obf}}{N(HLT)} \tag{10.8}$$

The structural mismatch in the context of CDFG can be evaluated based on number of modified node of the obfuscated CDFG compared to the total number of nodes of the original CDFG (before applying any transformation). In this context, the definition of a

Table 10.19: Results of proposed low-cost, obfuscated solution for different particle size

Benchmark	**Particle size**	**Low-cost design solution**	**Obfuscated hardware area (μm^2)**	**Obfuscated execution latency (ps)**	**Obfuscated design cost**
2D Autoregression Lattice Filter(ARF)	3	4A, 2M	241.83	1900.80	-0.37
	5	4A, 2M	241.83	1900.80	-0.37
	7	4A, 2M	241.83	1900.80	-0.37
Differential Equation	3	2A, 3M, 1C, 16UF	330.99	4498.80	-0.33
	5	2A, 3M, 1C, 16UF	330.99	4498.80	-0.33
	7	2A, 3M, 1C, 16UF	330.99	4498.80	-0.33
Auto-Correlation	3	4A, 8M,1C, 8UF	726.17	3557.15	-0.46
	5	7A, 8M,1C, 8UF	800.68	2940.45	-0.47
	7	7A, 8M,1C, 8UF	800.68	2940.45	-0.47
FIR (6-tap)	3	2A, 2M,1C, 8UF	214.99	1477.85	-0.36
	5	2A, 2M,1C, 8UF	214.99	1477.85	-0.36
	7	2A, 2M,1C, 8UF	214.99	1477.85	-0.36
DHMC	3	4A, 4M, 1C, 6UF	419.07	8581.61	-0.63
	5	4A, 4M, 1C, 6UF	419.07	8581.61	-0.63
	7	4A, 4M, 1C, 6UF	419.07	8581.61	-0.63
Adaptive Filter (noise cancellation)	3	4A, 1S, 1M, 1C, 30UF	270.83	3120.01	-0.63
	5	4A, 1S, 1M, 1C, 30UF	270.83	3120.01	-0.63
	7	4A, 1S, 1M, 1C, 30UF	270.83	3120.01	-0.63
Adaptive Filter (least mean square)	3	5A, 1S, 1M, 1C, 30UF	292.26	6833.58	-0.70
	5	5A, 1S, 1M, 1C, 30UF	292.26	6833.58	-0.70
	7	5A, 1S, 1M, 1C, 30UF	292.26	6833.58	-0.70

Table 10.20: Measuring power of obfuscation for each HLT technique

Benchmark	p^{obf} **for ROE**	p^{obf} **for LT**	p^{obf} **for THT**	p^{obf} **for LU**	p^{obf} **for LICM**	**Total** p^{obf}
ARF	0.32	0.61	0	-	-	0.55
Diff-Eqn	0	0.44	0	1	-	0.72
Auto-Co	0	0	0.49	1	-	0.75
FIR	0	0	0.50	1	-	0.75
DHMC	0	0.38	0	1	-	0.69
Ada-NC	0	0.67	0.03	1	-	0.57
Ada-LMS	0	0.75	0	1	-	0.88

modified node is as follows:

- If the parents of a node in the transformed CDFG is changed than its original.

- If the child of a node in the transformed CDFG is changed than its original.

Table 10.21: Comparison of proposed obfuscated design with non-obfuscated design in terms of hardware area, execution latency and design cost

Benchmark	**Original non-obfuscated design**			**Proposed obfuscated design**		
	Hardware area (μm^2)	**Execution latency (ps)**	**Design cost**	**Hardware area (μm^2)**	**Execution latency (ps)**	**Design cost**
ARF	241.8	2573.50	-0.26	241.8	1900.8	-0.37
Diff-Eqn	333.0	7246.5	-0.18	333.0	4498.8	-0.33
Auto-Co	736.4	3399.2	-0.48	736.4	2940.5	-0.46
FIR	215.0	1661.3	-0.31	215.0	1477.8	-0.35
DHMC	419.1	8872.1	-0.27	419.1	8581.6	-0.63
Ada-NC	270.8	3330.98	-0.62	270.8	3120.01	-0.63
Ada-LMS	292.3	12561.3	-0.61	292.3	6833.58	-0.70

Table 10.22: Comparison of proposed obfuscated design with [32] in terms of hardware area, execution latency and design cost

Benchmark	**Design of [32]**			**Proposed obfuscated design**		
	Hardware area (μm^2)	**Execution latency (ps)**	**Design cost**	**Hardware area (μm^2)**	**Execution latency (ps)**	**Design cost**
ARF	788.8	1511.5	-0.08	241.8	1900.8	-0.37
Diff-Eqn	3590.1	2326.4	-0.15	333.0	4498.8	-0.33
Auto-Co	3196.1	1144.9	-0.32	736.4	2940.5	-0.46
FIR	399.5	1315.7	-0.25	215.0	1477.8	-0.35
DHMC	4818.9	2891.1	-0.28	419.1	8581.6	-0.63
Ada-NC	2959.0	2826.5	-0.28	270.8	3120.01	-0.63
Ada-LMS	6075.2	4385.8	-0.30	292.3	6833.58	-0.70

- If the operation type of a node in the transformed CDFG is changed than its original.
- If a node present in the obfuscated design does not exist in the original CDFG or vice versa.

Table 10.20 reports the P_i^{obf} due to each HLT technique as well as the total P^{obf} after consecutively applying multiple HLTs. As evidence from the table, for ARF benchmark, the P_i^{obf} is 0.32, 0.61, 0 for ROE, LT and THT respectively (applied individually), further, the total normalized P^{obf} is 0.55 for multi-stage obfuscation (all transformations are applied consecutively).

Table 10.23: Comparison of proposed obfuscated design with [32] in terms of P^{obf}

Benchmark	P^{obf} **for proposed**	P^{obf} **for [32]**	P^{obf} **improvement %**
ARF	0.55	0.43	22.36
Diff-Eqn	0.72	0.50	30.62
Auto-Co	0.75	0.50	32.99
FIR	0.75	0.50	33.33
DHMC	0.69	0.50	27.27
Ada-NC	0.57	0.33	41.18
Ada-LMS	0.88	0.50	42.86

10.4.2 Comparative analysis

This section compares the proposed multi-stage obfuscation approach with the baseline design and a single-stage obfuscation approach [32]. Table 10.21 reports the comparison between the proposed approach with its corresponding original (non-secured) design in terms of hardware area, execution latency and design cost. It can be observed from Table 10.21 that an obfuscated design may achieve lesser execution time as compared to the original design due to multi-stage transformations. For example, applying ROE, THT, LT, unrolling etc., may result in reduction of operations in the graph. Therefore, the structurally transformed graph may optimize the latency after scheduling.

Table 10.22 reports the comparison between the proposed approach with [32] in terms of hardware area, execution latency and design cost. It can be observed from Table 10.22 that the propose obfuscated approach explores better design solution for all the tested benchmarks. Finally, the proposed approach achieves an average 55% reduction of design cost for standard benchmarks [17, 4], compared to [32]. This is obtained as the approach incorporates PSO-DSE framework to maintain area-delay trade-off of the obfuscated design.

The comparative results of the proposed approach with [32], in terms of P^{obf} are reported in Table 10.23. As evidence from the table, the proposed multi-stage based structural obfuscation methodology provides higher robustness for all the tested benchmarks compared to [32] as it has higher P^{obf} value. Further, it can be observed that the proposed approach achieves an average 22% higher robustness than [32].

10.5 Experimental Results: Obfuscation to Secure Multimedia Processor IP Core of CE Systems

This section discusses the experimental results of the proposed low-cost obfuscated JPEG CODEC IP core explained in Chapter 8. To verify the designed JPEG CODEC IP cores, standard 512x512 gray scale test images [15] and NASA images [39] are used as image dataset to generate compressed/decompressed images through the proposed IP cores. Both the proposed IP cores are designed, implemented and simulated in Intel Quartus v7.2 software with respect to Cyclone II family [14], device no. EP2C35F672C6 at 50MHz clock frequency. However, it can be emulated in any FPGA device. Table 10.24 reports the FPGA device utilization summary of the proposed JPEG CODEC IP cores.

The proposed low-cost obfuscated JPEG CODEC IP design is compared with a non-optimized obfuscated JPEG CODEC IP design to show the impact of design PSO-DSE based optimization. Table 10.25 reports the comparison of the two aforementioned designs in terms of hardware area, execution latency and design cost. The design cost is calculated using Eqn. 8.1, which is a weighted function of normalized hardware area and execution latency of the design. Further, the NanGate library is used to evaluate both the hardware area and the execution latency of a design [37]. It can be observed from Table 10.25, that the proposed JPEG CODEC IP core achieves reduction of greater than 5% in design cost compared to the non-optimized obfuscated JPEG. This is due to the tree height transfor-

Table 10.24: Device utilization summary of proposed JPEG CODEC IP cores w.r.t Intel Cyclone II FPGA

Device utilization summary	**Total used**	**Total available**	**Used %**
Total logic elements	12148	33216	37
4 input functions	5018	11072	45
3 input functions	6608	11072	60
<= 2 input functions	494	11072	4
Total registers	1826	34593	5
Total pins	322	475	68

Table 10.25: Comparison between non-optimized obfuscated JPEG CODEC IP core with proposed low-cost obfuscated IP core in terms of hardware area, execution latency and design cost

Design metrics	**Non-optimized obfuscated JPEG IP core**	**Proposed low-cost obfuscated IP core**
Resource configuration	4A, 4M	3A, 3M
Hardware area (μm^2)	397.94	298.45
Execution latency (ps)	5860.40	7603.22
Design cost	0.3884	0.3671

mation applied on the JPEG CODEC data flow graph followed by integration of particle swarm optimization driven DSE framework. Tree height transformation in the proposed methodology drastically reduces the length of the critical path of the DFG thus minimizing schedule delay. This impacts reduction of cost. Subsequently, this transformed DFG is fed into PSO-DSE which iteratively prunes the design space and explores an optimal low cost design resource. As these are novel solutions to reduce the design cost and never applied during JPEG CODEC IP core design before, these two layers of optimizations are not performed by synthesis tools. Since the proposed CODEC achieves reduction of design cost/overhead thus it has been called 'low-cost JPEG CODEC IP core'.

Further, Table 10.26 reports the comparison between proposed low-cost obfuscated JPEG CODEC IP core with a non-obfuscated JPEG CODEC IP core in terms of design area, latency, cost and 'Power of Obfuscation'. The PoO metric is explained in Chapter 6 and can be express as:

$$P^{obf} = \frac{a_i}{a_i^T} \tag{10.9}$$

This metric indicates how strong an obfuscated JPEG CODEC design is concealed in terms of structural identity. The more the design is obfuscated, higher is the complexity in discovering the functionality through the architecture, minimizing chances of RE. Modification of a node in a DFG indicates either, (a) the source/parents of a node is changed, or, (b) the child of a node is changed, or, (c) operation type of a node is changed, compared to the original. As shown in Table 10.26, PoO of 76% is achieved through the proposed

Table 10.26: Comparison between non-obfuscated JPEG CODEC IP core with proposed low-cost obfuscated IP design in terms of area, latency cost and power of obfuscation

Design metrics	**Non-obfuscated JPEG IP design**	**Proposed low-cost obfuscated IP design**	**Structure change due to obfuscation**	**Improved PoO through proposed design (%)**
Resource configuration	4A, 8M, 12 (8:1) Mux, 12 (16:1)Mux, 6 (1:8)Demux, 6 (1:16) Demux, 25Reg	3A, 3M, 10 (32:1) Mux, 2 (16:1)Mux, 5 (1:8)Demux, 1 (1:16) Demux, 13Reg	10064 Gates	-
Hardware area (μm^2)	483.66	298.45	-	-
Execution latency (ps)	6533.08	7603.22	-	-
Design cost	0.4582	0.3671	-	-
Power of obfuscation	0	0.7574	-	76%

Table 10.27: Storage size, reduction percentage, MSE and PSNR of compressed image for Q_{90} (Images 1 to 6 have been extracted from standard image datasets [15] and [39])

Images	**Original size (bits)**	**Compressed size (bits)**	**Compression efficiency (%)**	**MSE**	**PSNR**
Image 1	1048576	236568	77.44	3.84	17.68
Image 2	1048576	200000	80.93	3.50	18.08
Image 3	1048576	170640	83.73	2.00	20.51
Image 4	1048576	194144	81.49	2.63	19.32
Image 5	1048576	210760	79.90	3.25	18.40
Image 6	1048576	216136	79.39	2.20	20.09

approach compared to standard non-obfuscated JPEG CODEC IP. Further, as shown in this table, the change (reduction) in gates due to proposed obfuscation is more than 10 thousand. This indicates massive structural transformation at gate level of JPEG CODEC IP core architecture due to transformation in functional resources, multiplexers, demultiplexers and registers. This massive transformation makes the architecture/structure of JPEG IP core non-obvious to an adversary in terms of functionality. An adversary would find it difficult to discover the actual functionality of the design structure.

Table 10.28: Storage size, reduction percentage, MSE and PSNR of compressed image for Q_{70} (Images 1 to 6 have been extracted from standard image datasets [15] and [39])

Images	**Original size (bits)**	**Compressed size (bits)**	**Compression efficiency (%)**	**MSE**	**PSNR**
Image 1	1048576	106768	89.82	4.85	16.66
Image 2	1048576	99120	90.55	4.23	17.26
Image 3	1048576	90584	91.36	2.91	18.88
Image 4	1048576	91952	91.23	3.50	18.08
Image 5	1048576	102112	90.26	3.83	17.69
Image 6	1048576	105168	89.97	3.11	18.60

To perform image compression and decompression using the devised JPEG CODEC IP core, total six images are chosen from 'Dataset of standard 512x512 grayscale test images' [15] and 'NASA images' [39] datasets. The compression is performed for two different quality levels i.e. quality level 90 (Q90) and 70 (Q70). Additionally, the MSE and the PSNR of the corresponding compressed images obtained through devised JPEG CODEC IP core design are also calculated. As PSNR is commonly used to calculate the quality of the reconstructed image of lossy compression, it is used to evaluate image degradation of compressed images obtained through devised JPEG CODEC IP core. PSNR is defined for two images A and B of size M x N as [29]:

$$PSNR(A,B) = 20\log_{10}\frac{MAX_A}{\sqrt{MSE(A,B)}} \tag{10.10}$$

where, MSE(A, B) can be defined as [29]:

$$MSE(A,B) = \frac{1}{M*N}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\big(A(i,j)-B(i,j)\big)^2 \tag{10.11}$$

Table 10.27 and Table 10.28 reports the storage size, reduction percentage, MSE and PSNR of the compressed image for quality level 90 and 70, respectively. It can be observed that higher compression efficiency is achieved for Q_{70} than Q_{90} while compromising the image quality (indicated by PSNR).

10.6 Experimental Results: Hardware Trojan Secured IP Core of CE Systems

This section discusses the experimental results of the proposed low-cost hardware Trojan security aware design methodology explained in Chapter 9. This section is divided into following sub-sections: (a) Experimental setup and benchmarks which discusses the machine setup, control parameters and benchmarks used (b) Analysis of results which presents the proposed results for single and nested loop based CDFGs based on user constraints and (c) Comparison with prior related research in terms of implementation cost and final architectural solution is performed.

10.6.1 Experimental setup and benchmarks

The proposed approach as well as [46] both have been implemented in Java and run on Intel Core-i53210M CPU with 3MB L3 cache memory, 4 GB DDR3 primary memory and processor frequency of 2.5 GHz. During experiment, 15 runs were executed for proposed PSO-DSE. During the experimental conduct it was found that the proposed approach is scalable and is able to handle problems of medium/large size. Further during the experiments, the following optimal settings from [55] for PSO framework were fixed: ω (inertia weight) = linearly decreasing between 0.9 and 0.1; b_1 & b_2 (acceleration coefficient) = 2; r_1 & r_2 (random numbers) = 1; p (swarm size) = 3 or 5 or 7. The benchmarks used during experiments comprises of loop based control intensive applications derived from multimedia and signal processing domain. More explicitly, single and nested loop based control data flow graphs were used as standard benchmarks from [18, 17]. These benchmarks are typically data and control intensive applications which consume huge power and processing delay. The benchmark application is made up of operations that utilize third party micro IPs such as adder, subtractor, comparator etc. present in the tool library. For evaluating the Trojan detection capability (i.e. testing purpose) of the proposed approach, Trojan logic was inserted into these micro 3PIPs by altering the digital logic of the circuit through a combination of additional gates, multiplexers, flip-flops, control signals etc. A demonstra-

Table 10.29: Total hardware area and execution latency of the proposed approach

Benchmark	L_{cons} (μs)	L_T^{DMR} (μs)	$A_{cons}(au)$	$A_T^{DMR}(au)$
Autocorrelation filter (U1 = 2, U2 = 3)	50.04	34.62	19000	17996
Autocorrelation filter (U1 = 3, U2 = 3)	65.00	35.43	23000	22928
Autocorrelation filter (U1 = 4, U2 = 2)	55.00	24.43	28000	27860
DHMC (U1 = 2, U2 = 3)	420.00	262.33	35000	33524
DHMC (U1 = 3, U2 = 3)	620.00	507.35	35000	26994
DHMC (U1 = 4, U2 = 2)	550.00	327.80	36000	35992
Differential Eqn. (U = 4)	182.21	120.81	43301	26994
FFT (U = 2)	200.00	156.85	41000	26128
FIR (U = 6)	40.00	23.62	27000	22928

tive example which shows the procedure for Trojan insertion in BCD adder present as 3PIP module in HLS tool library is shown in Fig. 9.2 (Section 9.1.2). This type of modeling is realistic as mostly in real life cases the Trojan gets triggered by activation of a control line (such as 'En' in Fig. 9.2).

10.6.2 Analysis of results

For single loop based CDFGs representing DSP cores

For single loop based applications, the proposed approach is able to attain final solutions which lie within the user provided constraints of area and execution delay (as well as minimizes the hybrid cost in Eqn. 9.5), as shown in Table 10.29. For example, for Differential Equation benchmark, the proposed approach generates the final solution, (X_i): $N(R_{adder})$, $N(R_{subtractor})$, $N(R_{multiplier})$, $N(R_{comparator})$, U, A_v = 3(+), 3(−), 5(∗), 2(<), U = 4, $A_v = 00$, with L_T^{DMR} = 120.81 μs and A_T^{DMR} = 29136 au, which completely satisfies the area-delay constraints (A_{cons} = 43301 au and L_{cons} = 182.21 μs). (Note: A_{cons}

and L_{cons} could be any value between the minimum (A_{min}^{DMR}, L_{min}^{DMR}) and maximum value (A_{max}^{DMR}, L_{max}^{DMR}). Similar behavior is observed for the other benchmarks.

For nested loop based CDFGs

In case of nested loop based CDFGs, the proposed approach is able to attain final solutions which lie within the user provided constraints of area and execution delay (as well as minimizes the hybrid cost in Eqn. 9.5), as shown in Table 10.29. For example, for Auto correlation Filter benchmark, the proposed approach generates the final solution, (X_i): $N(R_{adder})$, $N(R_{multiplier})$, $N(R_{comparator})$, U1, U2, A_v = 2(+), 4(∗), 2(<), U1 = 2, U2 = 3, $A_v = 00$, with L_T^{DMR} = 34.620 μs and A_T^{DMR} = 17996.00 au, which completely satisfies the area-delay constraints (A_{cons} = 19000.00 au and L_{cons} = 50.04 μs).

10.6.3 Comparison with related prior research

It is important to note that there is no work in the literature that explores the low cost Trojan secured schedule for nested loops DSP kernels. The costs function for both [46] and the proposed approach consider the area of single comparator/error detection block responsible for runtime Trojan detection at the final output. However, the cost function does not have any impact of comparator/error detection block as it is used in both the approaches. Table 10.30 shows the results of the approach [46], while Table 10.31 shows the comparative results of the proposed approach with [46] in terms of cost and final Trojan secured hardware solution found. Although [46] does not handle single and nested loop CDFGs during Trojan security, however for comparison purpose, [46] have been fully re-implemented by feeding the completely unrolled version of each CDFG (both single and nested loop). Complete unrolling of the CDFG (by converting it into a straightforward DFG) was required as [46] cannot handle CDFGs concurrently with resource configuration during exploration. The proposed approach generates superior results than [46] due to the following reasons:

- The integration of allocation procedure of vendors A_v with the encoding for exploration, bears an impact on the final optimization quality.

Table 10.30: Total hardware area and execution latency of [46]

Benchmark	L_{cons} (μs)	L_T^{DMR} (μs)	$A_{cons}(au)$	$A_T^{DMR}(au)$
Autocorrelation filter (U1 = 2, U2 = 3)	50.04	33.81	19000	30396
Autocorrelation filter (U1 = 3, U2 = 3)	65.00	45.08	23000	31278
Autocorrelation filter (U1 = 4, U2 = 2)	55.00	45.08	28000	31404
DHMC (U1 = 2, U2 = 3)	420.00	286.35	35000	52824
DHMC (U1 = 3, U2 = 3)	620.00	440.89	35000	66054
DHMC (U1 = 4, U2 = 2)	550.00	327.80	32000	35992
Differential Eqn. (U = 4)	182.21	131.27	43301	29640
FFT (U = 2)	200.00	179.24	41000	33974
FIR (U = 6)	40.00	34.08	27000	24692

- In case of single loop based applications, the proposed approach consists of another supplementary option in encoding for exploration of an optimal unrolling factor.
- The proposed approach provides exploration of schedule resources based on user constraints which also bears an effect on the final design quality.

It can be seen from the results that even though for some benchmarks the number of resources in the final solution for [46] are lower than the proposed solution, since it has no feature for exploring an optimal unrolling factor and distinct vendor allocation, it uses the maximum unroll factor value as well as same vendor allocation rule to each unit ($A_v = 01$) during DMR design thereby resulting in higher cost. Therefore, distinct vendor allocation type $A_v = 00$ always yields lower cost final solution than [46].

Table 10.32 reports the impact of population size p on the runtime of proposed approach. For all the benchmarks, the runtime to find the solutions escalates with the growth in population size. This is due to the increase in computational complexity per iteration. However, no improvement in the quality of the solution is found.

Table 10.31: Comparison of proposed approach with [46] in terms of design cost

Benchmark	**Design solution for proposed**	**Design solution for [46]**	**Design cost of proposed**	**Design cost of [46]**
Autocorrelation filter (U1 = 2, U2 = 3)	2(+), 4(*), 2(<) U1=2, U2=3, Av=00	3(+), 5(*), 2(<) U1=2, U2=3, Av=01	-0.134	0.096
Autocorrelation filter (U1 = 3, U2 = 3)	2(+), 8(*), 2(<) U1=3, U2=3, Av=00	3(+), 5(*), 2(<) U1=3, U2=3, Av=01	-0.149	0.025
Autocorrelation filter (U1 = 4, U2 = 2)	2(+), 8(*), 2(<) U1=4, U2=2, Av=00	3(+), 5(*), 2(<) U1=4, U2=2, Av=01	-0.173	-0.017
DHMC (U1 = 2, U2 = 3)	4(+), 6(*), 2(<) U1=2, U2=3, Av=00	3(+), 5(*), 2(<) U1=2, U2=3, Av=01	-0.140	0.057
DHMC (U1 = 3, U2 = 3)	3(+), 4(*), 2(<) U1=3, U2=3, Av=00	3(+), 5(*), 2(<) U1=3, U2=3, Av=01	-0.131	0.173
DHMC (U1 = 4, U2 = 2)	4(+), 6(*), 2(<) U1=4, U2=2, Av=00	3(+), 5(*), 2(<) U1=4, U2=2, Av=01	-0.136	-0.103
Differential Eqn. (U = 4)	2(+), 2(-), 5(*), 2(<), U=4, Av=00	2(+), 2(-), 5(*), 2(<), U=4, Av=01	-0.232	-0.193
FFT (U = 2)	4(+), 2(-), 4(*), 1(<), U=2, Av=00	5(+), 2(-), 5(*), 2(<), U=2, Av=01	-0.185	-0.088
FIR (U = 6)	4(+), 8(*), 2(<) U=6, Av=00	3(+), 5(*), 2(<) U=6, Av=01	-0.175	-0.074

Table 10.32: Comparison of exploration time for various swarm size p

Benchmark	**Exploration time (ms) for swarm size ‘p’**		
	p = 3	**p = 5**	**p = 7**
Autocorrelation filter (U1 = 2, U2 = 3)	2144	4305	4893
Autocorrelation filter (U1 = 3, U2 = 3)	3297	4091	8080
Autocorrelation filter (U1 = 4, U2 = 2)	4928	5536	7860
DHMC (U1 = 2, U2 = 3)	21836	43218	62345
DHMC (U1 = 3, U2 = 3)	29823	49876	68243
DHMC (U1 = 4, U2 = 2)	29066	45327	65827
Differential Eqn. (U = 4)	2084	6359	9243
FFT (U = 2)	6856	9132	9205
FIR (U = 6)	638	1132	1775

Chapter 11

Conclusion

This thesis has presented novel methodologies for securing/protecting data and control intensive applications used in CE systems. The protection/security algorithms are integrated with these application specific datapath processors for DSP and multimedia kernels during the in-synthesis process at architectural level. Therefore, this thesis has accomplished the following objectives:

- Proposes single-phase watermarking methodology to solve the problem of fraudulent claim of ownership, IP piracy and IP cloning. The experimental results over the standard applications indicate an average reduction in the final embedding cost of 6% and higher security compared to recent approach.

- Proposes multi-phase watermarking methodology to solve the problem of tampering or removal of the implanted signature. The experimental results over the standard applications indicate it yields zero delay and minimal area overhead compared to the baseline and achieves average cost reductions of 7.38% and 6.25% compared to two similar approaches. Further, it achieves a lower P_c value by 3.2×10^{27} times in magnitude compared to similar approaches. Additionally, it is 3.4×10^{43} and 2.8×10^{19} times more tamper tolerant than similar approaches.

- Proposes symmetrical IP core protection methodology to solve the problem of abusing the rights of the IP Core buyer and seller. Experimental results over the standard applications indicate an average 1% design cost overhead compared to baseline (unprotected) design and <1% design cost overhead compared to a non-symmetrical approach.

- Proposes multi-stage structural obfuscation methodology to solve the problem of reverse engineering attack. Experimental results over the standard applications indicate an improvement in the power of obfuscation by 22% and reduction in obfuscated design cost by 55% is achieved compared to recent prior work.

- Proposes an enhancement in the functional obfuscation for DSP/MP IP core methodology to solve the problem of SAT and removal attacks. A custom lightweight AES module is proposed and designed in a standard CAD tool to achieve the purpose. The AES module utilizes <1% of the available design logic elements of the FPGA.

- Proposes a design methodology for generating structurally obfuscated JPEG CODEC to prevent RE attacks on it. The proposed approach optimized the design cost of the obfuscated JPEG CODEC IP core using PSO based DSE by 5% compared to a non-obfuscated design and enhanced the PoO by 76% compared to a non-obfuscated design.

- Proposes a Trojan secured DMR schedule methodology to solve the problem of generating a low-cost Trojan secured datapath. A sequence of test vectors generated through 8th order polynomial Linear-Feedback Shift Register (LFSR) as ATPG are fed into the DMR schedule for comprehensive coverage of Trojan detection. The results generated indicate that a detection rate of 100% was achieved while handling such Trojans.

All these methodologies handle attacks through HLS framework. Therefore, both higher and lower design abstraction level will be secured through the proposed approaches. Security analysis of all the defense mechanisms ensures strong robustness. In this context, the proposed method promises significant improvement in terms of security and design overhead in comparison to recent prior works.

References

[1] S. An and C. Wang. Recursive algorithm, architectures and FPGA implementation of the two-dimensional discrete cosine transform. *IET Image Processing*, 2(6):286–294, December 2008.

[2] C. Barria, D. Cordero, C. Cubillos, and R. Osses. Obfuscation procedure based in dead code insertion into crypter. In *6th International Conference on Computers Communications and Control (ICCCC)*, pages 23–29, May 2016.

[3] A. Baumgarten, A. Tyagi, and J. Zambreno. Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Design Test of Computers*, 27(1):66–75, Jan 2010.

[4] BDTI DSP Kernel Benchmarks. `https://www.bdti.com/Services/Benchmarks/DKB`.

[5] M. Brzozowski and V. N. Yarmolik. Obfuscation as Intellectual Rights Protection in VHDL Language. In *6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM'07)*, pages 337–340, June 2007.

[6] B. Cakr and S. Malik. Hardware Trojan detection for gate-level ICs using signal correlation based clustering. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 471–476, March 2015.

[7] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J. H. Anderson, S. Brown, and T. Czajkowski. Legup: High-level synthesis for fpga-based processor/accelerator systems. In *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '11, pages 33–36, 2011.

[8] E. Castillo, U. Meyer-Baese, A. Garcia, L. Parrilla, and A. Lloris. IPP@HDL: Efficient Intellectual Property Protection Scheme for IP Cores. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(5):578–591, May 2007.

[9] R. S. Chakraborty and S. Bhunia. HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(10):1493–1502, Oct 2009.

[10] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia. MERO: A Statistical Approach for Hardware Trojan Detection. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 396–410, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[11] P. Coussy, D. D. Gajski, M. Meredith, and A. Takach. An Introduction to High-Level Synthesis. *IEEE Design Test of Computers*, 26(4):8–17, July 2009.

[12] P. Coussy and A. Morawiec. *High-Level Synthesis: From Algorithm to Digital Circuit*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[13] X. Cui, K. Ma, L. Shi, and K. Wu. High-level synthesis for run-time hardware trojan detection and recovery. In *51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2014.

[14] Cyclone II FPGA. `https://www.altera.com/products/fpga/cyclone-series/cyclone-ii/overview.html`.

[15] Dataset of standard 512x512 grayscale test images. `http://decsai.ugr.es/cvg/CG/base.htm`.

[16] A. R. Desai, M. S. Hsiao, C. Wang, L. Nazhandali, and S. Hall. Interlocking obfuscation for anti-tamper hardware. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, CSIIRW '13, pages 8:1–8:4, 2013.

[17] DSP benchmark suite. `http://www.ece.ucsb.edu/EXPRESS/benchmark/`.

[18] S. Dupuis, P. Ba, M. Flottes, G. D. Natale, and B. Rouzeyre. New testing procedure for finding insertion sites of stealthy hardware trojans. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 776–781, March 2015.

[19] S. Dupuis, P. Ba, G. D. Natale, M. Flottes, and B. Rouzeyre. A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans. In *IEEE 20th International On-Line Testing Symposium (IOLTS)*, pages 49–54, July 2014.

[20] Q. Gang and M. Potkonjak. *Intellectual Property Protection in VLSI Designs: Theory and Practice*. Springer Science and Business Media, 2007.

[21] S. H. Gerez. *Algorithms for VLSI Design Automation*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1999.

[22] I. Hong and M. Potkonjak. Behavioral synthesis techniques for intellectual property protection. In *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361)*, pages 849–854, June 1999.

[23] iPhone 5 A6 SoC reverse engineered, reveals rare hand-made custom CPU, and tri-core GPU. `https://www.extremetech.com/computing/136749-iphone-5-a6-soc-reverse-engineered-reveals-rare-\hand-made-custom-cpu-and-a-tri-core-gpu`.

[24] Y. Jin and Y. Makris. Hardware trojan detection using path delay fingerprint. In *IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 51–57, June 2008.

[25] D. Kachave and A. Sengupta. Shielding CE Hardware Against Reverse-Engineering Attacks Through Functional Locking. *IEEE Consumer Electronics Magazine*, 7(2):111–114, March 2018.

[26] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor. Trustworthy hardware: Identifying and classifying hardware trojans. *Computer*, 43(10):39–46, Oct 2010.

[27] J. Kim, E. Jung, Y. Lee, and W. Ryu. Home appliance control framework based on smart tv set-top box. *IEEE Transactions on Consumer Electronics*, 61(3):279–285, Aug 2015.

[28] P. Kollig and B. M. Al-Hashimi. Simultaneous scheduling, allocation and binding in high level synthesis. *Electronics Letters*, 33(18):1516–1518, Aug 1997.

[29] E. Kougianos, S. P. Mohanty, G. Coelho, U. Albalawi, and P. Sundaravadivel. Design of a high-performance system for secure image communication in the internet of things. *IEEE Access*, 4:1222–1242, 2016.

[30] F. Koushanfar, I. Hong, and M. Potkonjak. Behavioral Synthesis Techniques for Intellectual Property Protection. *ACM Trans. Des. Autom. Electron. Syst.*, 10(3):523–545, July 2005.

[31] J. Lach, W. H. Mangione-Smith, and M. Potkonjak. Fingerprinting techniques for field-programmable gate array intellectual property protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(10):1253–1261, Oct 2001.

[32] Y. Lao and K. K. Parhi. Obfuscating dsp circuits via high-level transformations. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(5):819–830, May 2015.

[33] B. Le Gal and L. Bossuet. Automatic low-cost ip watermarking technique based on output mark insertions. *Des. Autom. Embedded Syst.*, 16(2):71–92, June 2012.

[34] Y. Lee and N. A. Touba. Improving logic obfuscation via logic cone analysis. In *16th Latin-American Test Symposium (LATS)*, pages 1–6, March 2015.

[35] G. D. Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education, 1st edition, 1994.

[36] S. P. Mohanty, N. Ranganathan, E. Kougianos, and P. Patra. *Low-Power High-Level Synthesis for Nanoscale CMOS Circuits*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[37] NanGate FreePDK15 Open Cell Library NanGate. `http://www.nangate.com/?page_id=2328`.

[38] S. Narasimhan, D. Du, R. S. Chakraborty, S. Paul, F. Wolff1, C. Papachristou, K. Roy, and S. Bhunia. Multiple-parameter side-channel analysis: A non-invasive hardware trojan detection approach. In *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 13–18, June 2010.

[39] Nasa image and video library. `https://images.nasa.gov/`.

[40] T. Nie, L. Zhou, and Y. Li. Hierarchical watermarking method for fpga ip protection. *IETE Technical Review*, 30(5):367–374, 2013.

[41] K. K. Parhi. Verifying equivalence of digital signal processing circuits. In *Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 99–103, Nov 2012.

[42] P. G. Paulin and J. P. Knight. Algorithms for high-level synthesis. *IEEE Design Test of Computers*, 6(6):18–31, Dec 1989.

[43] P. G. Paulin and J. P. Knight. Force-directed scheduling for the behavioral synthesis of asics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(6):661–679, June 1989.

[44] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri. Security analysis of logic obfuscation. In *DAC Design Automation Conference 2012*, pages 83–89, June 2012.

[45] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri. Security analysis of integrated circuit camouflaging. In *ACM SIGSAC Conference on Computer & Communications Security*, pages 709–720, 2013.

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

[47] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri. Fault analysis-based logic encryption. *IEEE Transactions on Computers*, 64(2):410–424, Feb 2015.

[48] J. A. Roy, F. Koushanfar, and I. L. Markov. EPIC: Ending Piracy of Integrated Circuits. In *2008 Design, Automation and Test in Europe*, pages 1069–1074, March 2008.

[49] A. Sengupta. Intellectual property cores: Protection designs for ce products. *IEEE Consumer Electronics Magazine*, 5(1):83–88, Jan 2016.

[50] A. Sengupta and S. Bhadauria. Exploring Low Cost Optimal Watermark for Reusable IP Cores During High Level Synthesis. *IEEE Access*, 4:2198–2215, 2016.

[51] A. Sengupta, S. Bhadauria, and S. P. Mohanty. Embedding low cost optimal watermark during high level synthesis for reusable IP core protection. In *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, pages 974–977, May 2016.

[52] A. Sengupta, S. Bhadauria, and S. P. Mohanty. Low-cost security aware hls methodology. *IET Computers Digital Techniques*, 11(2):68–79, 2017.

[53] 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(4):655–668, April 2017.

[54] A. Sengupta and S. Kundu. Guest editorial securing iot hardware: Threat models and reliable, low-power design solutions. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(12):3265–3267, Dec 2017.

[55] A. Sengupta and V. K. Mishra. Swarm intelligence driven simultaneous adaptive exploration of datapath and loop unrolling factor during area-performance tradeoff. In *IEEE Computer Society Annual Symposium on VLSI*, pages 106–111, July 2014.

[56] A. Sengupta, R. Sedaghat, and Z. Zeng. A high level synthesis design flow with a novel approach for efficient design space exploration in case of multi-parametric optimization objective. *Microelectronics Reliability*, 50(3):424 – 437, 2010.

[57] D. L. Springer and D. E. Thomas. Exploiting the special structure of conflict and compatibility graphs in high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(7):843–856, July 1994.

[58] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 137–143, May 2015.

[59] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Design Test of Computers*, 27(1):10–25, Jan 2010.

[60] S. Thavalengal and P. Corcoran. User authentication on smartphones: Focusing on iris biometrics. *IEEE Consumer Electronics Magazine*, 5(2):87–93, April 2016.

[61] R. Torrance and D. James. The State-of-the-Art in IC Reverse Engineering. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 363–381, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[62] A. Vijayakumar, V. C. Patil, D. E. Holcomb, C. Paar, and S. Kundu. Physical Design Obfuscation of Hardware: A Comprehensive Investigation of Device and Logic-Level Techniques. *IEEE Transactions on Information Forensics and Security*, 12(1):64–77, Jan 2017.

[63] X. Wang, X. Jia, Q. Zhou, Y. Cai, J. Yang, M. Gao, and G. Qu. Secure and low-overhead circuit obfuscation technique with multiplexers. In *International Great Lakes Symposium on VLSI (GLSVLSI)*, pages 133–136, May 2016.

[64] X. Wang, M. Tehranipoor, and J. Plusquellic. Detecting malicious inclusions in secure hardware: Challenges and solutions. In *IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 15–19, June 2008.

[65] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley Publishing Company, USA, 4th edition, 2010.

[66] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty. Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme. In *Design, Automation and Test in Europe*, pages 1362–1365, March 2008.

[67] J. Yang, G. Zhu, and Y. Shi. Analyzing the Effect of JPEG Compression on Local Variance of Image Intensity. *IEEE Transactions on Image Processing*, 25(6):2647–2656, June 2016.

[68] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri. On Improving the Security of Logic Locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(9):1411–1424, Sept 2016.

[69] J. Zhang. A Practical Logic Obfuscation Technique for Hardware Security. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(3):1193–1197, March 2016.

[70] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu. VeriTrust: Verification for Hardware Trust. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(7):1148–1161, July 2015.

[71] D. Ziener and J. Teich. FPGA core watermarking based on power signature analysis. In *IEEE International Conference on Field Programmable Technology*, pages 205–212, Dec 2006.

[72] D. Ziener and J. Teich. Power signature watermarking of ip cores for fpgas. *J. Signal Process. Syst.*, 51(1):123–136, Apr. 2008.