TRANSIENT FAULT RELIABILITY AND SECURITY OF IP CORE

Ph.D. Thesis

By
DEEPAK KACHAVE

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
DECEMBER 2018

TRANSIENT FAULT RELIABILITY AND SECURITY OF IP CORE

A THESIS

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

by
DEEPAK KACHAVE

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
DECEMBER 2018

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **TRANSIENT FAULT RELIABILITY AND SECURITY OF IP CORE** in the partial fulfillment of the requirements for the award of the degree of **DOCTOR OF PHILOSOPHY** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING, INDIAN INSTITUTE OF TECHNOLOGY INDORE**, is an authentic record of my own work carried out during the time period from JUNE, 2015 to DECEMBER 2018 under the supervision of Dr. ANIRBAN SENGUPTA, Assistant 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
(DEEPAK KACHAVE)

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

Signature of Thesis Supervisor with date
(ANIRBAN SENGUPTA)

DEEPAK KACHAVE has successfully given his/her Ph.D. Oral Examination held on

Signature of Chairperson (OEB) Date:

Signature of External Examiner Date:

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

Signature of PSPC Member #1 Date:

Signature of PSPC Member #2 Date:

Signature of Convener, DPGC Date:

Signature of Head of Discipline
Date:

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Dr. Anirban Sengupta for providing me the opportunity to do work under his supervision. I would like to thank him for his persistence and faith in me, without his relentless effort, guidance and *deadlines* I would have not been to understand the importance of research and the sacrifice it requires to reach a certain level.

Further, I would like to thank my parents and TA supervisors for their continuous support. I would like to thank all the faculty members and colleagues for their continuous support. The time spent at IIT Indore will be one of the most valuable memory of my life.

I would like to thank Ministry of Electronics and Information Technology (MEITY) for financial support.

Dedicated to OLD HANDS who care for their children

(parents)

&

STRONG HANDS who mold careers

(Teachers)

ABSTRACT

The rapid growth of consumer electronics (CE) industry has led to cut-throat competition of developing sophisticated devices. As the complexity of the CE design increases along with shortening of time-to-market deadlines, the designers are becoming heavily reliant on reusable Intellectual Property (IP) cores generated at higher levels of design abstraction. A malicious attacker may exploit dependency on IP cores through security issues/vulnerabilities such as piracy, Trojan insertion, overbuilding, reverse engineering etc. Hence, methodologies are required to ensure security of the IP cores.

Further similar to IP core security, IP core reliability is also becoming a major concern. As the demand for CE devices with sophisticated features such as low-power consumption, smaller silicon area, etc. increases, the IP core designers are heavily depending upon technology scaling to meet these design objectives. However, technology scaling enhances several reliability concerns such as bias temperature instability, multi-cycle and multi-unit transient faults, electromigration etc. Hence, methodologies are required for designing reliable IP cores.

To advance the state-of-the-art for designing reliable and secured IP cores, this thesis makes following contributions: (a) A novel methodology for generating a DSP IP core that is simultaneously resilient/secure against multi-cycle (temporal) and (multi-unit) spatial effect of transient fault. (b) A novel methodology for generating a DSP IP core that is simultaneously tolerant against multi-cycle temporal and multi-unit spatial effect of transient fault for data intensive applications. (c) A novel methodology for generating a DSP IP core that is simultaneously tolerant against multi-cycle temporal and multi-unit spatial effect of transient fault for loop-based control intensive applications. (d) A novel methodology for generating a low-cost, highly secure, functionally obfuscated DSP IP core. (e) A novel methodology for analyzing the aging effect of NBTI stress on performance of DSP IP core. (f) A novel computational forensic engineering methodology for resolving ownership conflict of DSP IP core generated using high level synthesis.

LIST OF PUBLICATIONS

PEER-REVIEWED JOURNALS (10):

1. Anirban Sengupta, Deepak Kachave, "Spatial and Temporal Redundancy for Transient Fault Tolerant Datapath," in ***IEEE Transactions on Aerospace and Electronic Systems (TAES)***, Volume: 54, Issue:3, June 2018, pp. 1168-1183
2. Anirban Sengupta, Deepak Kachave, Dipanjan 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 (TCAD)***, Accepted, 2018.
3. Deepak Kachave, Anirban Sengupta, "Shielding CE Hardware Against Reverse-Engineering Attacks Through Functional Locking", in ***IEEE Consumer Electronics,*** vol. 7, no. 2, pp. 111-114, March 2018.
4. Deepak Kachave, Anirban Sengupta, "Performance Degradation of DSP Cores due to NBTI Stress Attack (Invited Paper)", ***IEEE Potentials***, 2018.
5. Deepak Kachave, Anirban Sengupta, "Applying digital forensic for hardware protection : resolving false claim of IP ownership", ***IEEE VLSI Circuits & Systems Letter***, Volume 4, Issue 1, pp. 10 - 13, Feb 2018.
6. Deepak Kachave, Anirban Sengupta, Shubha Neema, Panugothu Sri Harsha" Effect of NBTI Stress on DSP cores used in CE Devices: Threat Model and Performance Estimation", ***IET Journal on Computers & Digital Techniques (CDT)***, Volume 12, Issue 6, November 2018, p. 268 – 278.
7. Anirban Sengupta, Deepak Kachave "Particle Swarm Optimisation Driven Low Cost Single Event Transient Fault Secured Design during Architectural Synthesis (Invited Paper)", ***IET Journal of Engineering***, p. 184-194, 2017
8. Anirban Sengupta, Deepak Kachave "Low Cost Fault Tolerance against kc-cycle and km-unit Transient for Loop Based Control Data Flow Graphs during Physically Aware High Level Synthesis", ***Elsevier***

Journal on Microelectronics Reliability, Volume 74, July 2017, pp. 88-99.

9. Anirban Sengupta, Deepak Kachave "Forensic Engineering for Resolving Ownership Problem of Reusable IP Core generated during High Level Synthesis", ***Elsevier Journal on Future Generation Computer Systems***, Volume 80, Pages 29-46, March 2018.
10. Deepak Kachave, **Anirban Sengupta,** "Integrating Physical Level Design and High Level Synthesis for Simultaneous Multi-Cycle Transient and Multiple Transient Fault Resiliency of Application Specific Datapath Processors", ***Elsevier Journal on Microelectronics Reliability*****,** Volume 60, Pages 141-152, May 2016.

BOOK CHAPTER (1):

11. Deepak Kachave, Anirban Sengupta, "Hardware Reliability Analysis of DSP Cores", ***IET Book: VLSI and Post-CMOS Devices, Circuits and Modelling***, Invited Book Chapter, 2017.

PEER-REVIEWED CONFERENCES (3):

12. Anirban Sengupta, Deepak Kachave, "Generating Multi-cycle and Multiple Transient Fault Resilient Design During Physically Aware High Level Synthesis," ***2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)***, Pittsburgh, PA, 2016, pp. 75-80.
13. Anirban Sengupta, Deepak Kachave, Shubha Neema, Panugothu Sri Harsha, "Reliability and Threat Analysis of NBTI Stress on DSP Cores," ***2017 IEEE International Symposium on Smart Electronic Systems (IEEE-iSES, formerly IEEE-iNIS)***, Bhopal, 2017, pp. 11-14.
14. Deepak Kachave, Anirban Sengupta, "Protecting Ownership of Reusable IP Core Generated during High Level Synthesis," ***2016 IEEE International Symposium on Smart Electronic Systems (IEEE-iSES, formerly IEEE-iNIS)***, Gwalior, 2016, pp. 80-82.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

X_i	Particle encoding
L[k]	List of conflicting hardware
kc	Strength of temporal effect of transient fault
km	Strength of spatial effect of transient fault
O^U	Original unit
D^U	Duplicate unit
t(v)	control step (time) at which operation *v* is scheduled
t(v')	control step (time) at which operation *v'* is scheduled
t(v'')	control step (time) at which operation *v''* is scheduled
S(Mv)	Starting point of placement of hardware module (M) allocated to operation *v*
S(Mv')	Starting point of placement of hardware module (M) allocated to operation *v'*
cij	Connectivity between i^{th} and j^{th} hardware units
dij	Manhattan distance between i^{th} and j^{th} hardware units
$C_f(X_i)$	Cost/fitness of design solution with respect to resource configuration (X_i)
L^{DMR}	Latency of kc-cycle transient fault resilient design solution
L_{max}^{DMR}	Maximum latency of kc-cycle transient fault resilient design space
A^{FP}	Area of transient fault resilient design solution
A_{max}^{FP}	Maximum area of kc-cycle transient fault resilient design space
W^{FP}	Wirelength of transient fault tolerant floorplan
W_{max}^{FP}	Maximum wirelength of transient fault resilient design space
φ1, φ2, φ3	User defined weights
O^C	Original copy
D^C	Duplicate copy
T^C	Triplicate copy
v	Operation belonging to original copy
v'	Operations belonging to duplicate copy
v''	Operations belonging to triplicate copy
[Ri]	List of hardware resources in the kc-cycle fault tolerant schedule
$(Z_{R_i}[R_j])$	List of hardware in conflict with i^{th} resource due to spatial effect of TF
Pi	i^{th} particle of the swarm
NR_D	Number of resources in the D^{th} dimension of the design space
L^{FT}	Latency of fault tolerant design solution explored during PSO-DSE
L_{max}^{FT}	Maximum latency of the fault tolerant design space
A^{FT}	Area of the fault tolerant design solution
A_{max}^{FT}	Maximum area of the fault tolerant design solution
R_{d_i}	Number of resources of i^{th} particle in d^{th} dimension
$R_{d_i}^{+}$	Updated number of resources of ith particle in dth dimension

V_{d_i}	Velocity of the i^{th} particle in the d^{th} dimension of the design space
$V_{d_i}^{+}$	Updated velocity of the i^{th} particle in the d^{th} dimension of the design space
ω	Inertia weight
b1, b2	Acceleration coefficients
r1, r2	Random numbers
$R_{d_{gb}}$	Global best in the d^{th} dimension
$R_{d_{lb_i}}$	Local best in the d^{th} dimension
L^{TMR}	Latency of kc-cycles transient fault tolerant TMR design
L_{seq}	Latency of sequential body
L_{par}	Latency of the parallel body
μ	A random integer between 1 and T_{ILB} ($1 \leq \mu \leq T_{ILB}$).
T_{ILB}	Total number of ILBs in the initial design space before AES block integration
P^{OB}	Power of the obfuscated design solution explored during PSO-DSE
P_{max}^{OB}	Maximum power of the obfuscated design in the design space
D^{OB}	Delay of the obfuscated design solution explored during PSO-DSE
D_{max}^{OB}	Maximum delay of the obfuscated design in the design space
ΔV_{th}	Change in the threshold voltage
a	Input signal probability
b	Constant
t	Time in seconds
n	Time exponential constant
T	Delay of PMOS transistor
K	Technology based proportionality constant
V_{th}^{new}	New threshold voltage
IP_{ID}	Intellectual property core whose ownership is to be identified
$IP_{CT}n$	Intellectual property core generated using n^{th} claimant's tool
FUi	Functional unit of i^{th} type
CSs(FUi)	Starting control step of i^{th} functional unit
$CS_E(FUi)$	Ending control step of i^{th} functional unit
$CS_S(N)_1$	Starting control step of the data set 1
$CS_E(N)_1$	Ending control step of the data set 1
m	Matching percentage

ACRONYMS

HLS	High level synthesis
VLSI	Very Large Scale Integration
IP	Intellectual property
DSP	Digital signal processor
CE	Consumer Electronics
IC	Integrated Circuits
SoC	System on chip
RTL	Register Transfer Level
VHDL	Very High Speed Integrated Circuit Hardware Description Language
GDS	Graphic Database System
ALU	Arithmetic Logic Unit
DFG	Data Flow Graph
CDFG	Control Data Flow Graph
FSM	Finite State Machine
TF	Transient Fault
MTF	Multi-unit Transient Fault
MCT	Multi-cycle Transient Fault
SET	Single Event Transient
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
PMOS	P-channel Metal Oxide Semiconductor
NBTI	Negative Bias Temperature Instability
CFE	Computational Forensic Engineering
PSO	Particle Swarm Optimization
DSE	Design Space Exploration
DMR	Dual Modular Redundant
TMR	Triple Modular Redundant
ILB	IP functional Locking Block
LET	Linear Energy Transfer
SDFG	Scheduled Data Flow Graph
TFH	Transient Fault Hazards
CS	Control Step
FP	Floor Plan
FU	Functional unit
DWT	Discrete Wavelet Transform
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
ASP	Application Specific Processor
ASIC	Application Specific Integrated Circuit
UF	Unrolling Factor
3PIP	3rd party Intellectual Property
ASAP	As Soon As Possible
ALAP	As Late As Possible
opn	operation
Mux	Multiplexer
Demux	Demultiplexer

Chapter 1

Introduction

The invention of transistor in the mid-20th century has led to unimaginable progress of electronics industry. Since its invention, the reduction in transistor's dimensions has followed a well-known prediction termed as Moore's law [1]. In the 1970-80's the devices made from transistors such as computers could only be afforded by the large-scale industries/business-houses due to their features such as large size, high power consumption, high cost etc. However, as the transistor scaling continues, devices having low power consumption, compact form-factor, better heat dissipation, were made possible. These advances have led to a whole new industry, centered toward manufacturing electronics devices for personal/home usage known as consumer electronics (CE). Along with transistor scaling other technological advances such as internet, smart phones, etc. have made consumer electronics a major market force (with estimated sales in multi-billion dollars [2]). Due to huge demand of CE devices, the competition for designing best product and launching them as fast as possible has increased tremendously. The cut-throat competition has resulted in very stringent (short) time-to-market deadlines. Additionally, the increasing demand for miniscule devices possessing as many features as possible has resulted in enhanced design complexity (for devices such as smart phones, smart watches, etc.). In order to meet these stringent time-to-market deadlines as well as reduce design complexity, the device designers are highly dependent on third-party Intellectual property (IP) cores designed at higher levels of design abstraction through high-level synthesis / behavioral synthesis / architectural synthesis [3-5].

As more and more sophisticated electronics devices are becoming integral part of business-critical and mission-critical systems, along with globalization of supply-chain, the chances of a malicious attack on an electronics device in a mission-critical system has increased tremendously [3-5]. Therefore, it is mandatory to devise algorithms that can ensure ***security*** of IP cores.

Furthermore, the devices designed using scaled transistors are becoming more sensitive to their environment than earlier technology scales. Therefore, as the

technology scaling continues in the sub-nanometer range, the reliability of contemporary and future IP cores has become a major concern. Thus, methodologies are required for developing ***reliable*** IP core for mission-critical systems [6, 66-68].

This chapter briefly presents the background of the methodologies proposed in this thesis for designing reliable and secured IP cores. The first section discusses IP cores and its relevance in electronics industry. The second section briefly discusses various design abstraction levels of a generic integrated circuit (IC) design flow. The third section elaborates on the higher level of design abstraction through a process called 'high level synthesis (HLS)'. Subsequently, the fourth, fifth, and sixth sections discuss the proposed reliability and security methodologies. Finally, the seventh section discusses the organization of the thesis.

1.1. IP core and its background

An intellectual property core in electronics refers to a reusable logic block that is an intellectual property of an IP owner. Reusable IP cores plays a vital role in reducing the design complexity and helps the designers to meet time-to-market deadlines. An IP core is analogues to a library in the context of a computer program. Like a library, an IP core can be utilized to design a system on chip (SoC) quickly and easily. An IP buyer could purchase IP core (s) from third party IP vendors and combine them along with in-house technologies (if any) to generate a 'market-ready' product. For instance, consider a company interested in developing a personal computer, it may buy IP cores for digital signal processor (DSP), memory, etc. and combine it with its in-house components to create a 'market-ready' product. Thereby, reducing time, effort and cost to build in-house IP cores. An IP core can be of three types; soft IP core or hard IP core or firm IP core [7]. A soft IP core is typically delivered as a synthesizable Register transfer level (RTL) code in a hardware description language (such as Verilog or VHDL) or schematic design. Similarly, a hard IP is typically delivered as a layout design in the form of a GDS II file [4]. A soft IP core is comparatively more modifiable/tweakable than a hard IP core. The word hard and soft represents modifiability of these IP core. A question arises

several times that whether an IP core should be provided as a soft IP core or hard IP core? A hard IP core is easily predictable but not portable for instance, a hard IP core cannot be ported from initially targeted foundry to another foundry. On the other hand, a soft IP core is portable but not predictable i.e., its performance may vary significantly as it gets converted into lower levels of design abstractions. Therefore, a third type of IP core is required that is simultaneously predictable and portable. This type of an IP core is termed 'firm IP core' [7, 69]. An IP core design process can be clearly understood with the help of a generic integrated circuit design flow as discussed in section 1.2.

1.2. Generic VLSI design flow

A generic integrated circuit design flow is based on divide and conquer technique. As shown in fig. 1.1, a complex design is divided into various abstraction levels. At each level, design is optimized to achieve certain objectives/goals. A generic IC design flow takes system specification as input in the form of a programming language or a hardware description language. Subsequently, high level synthesis is performed to obtain register transfer level (RTL) datapath as discussed in section 1.3. Later, the RTL datapath is converted into gate level netlist using logic synthesis. The gate level netlist thus obtained is converted into layout design (typically in the form of a GDS II file) during physical design step of the IC design flow. The layout file thus generated is analyzed to check whether the layout design meets the design objectives (specification/constraints). Once the layout is verified, it is sent for fabrication. Once, the fabrication is completed, a 'die' is created. Subsequently, the die is packaged and tested. The test approved ICs are made

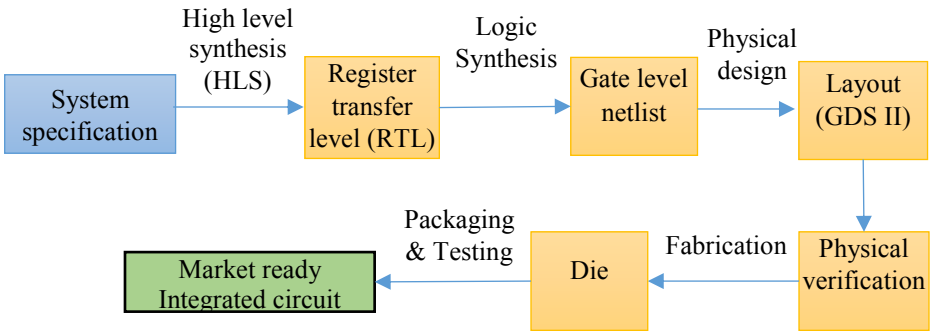


Fig. 1.1 Generic IC design flow

available in the market [3-7, 21].

1.3. Background on High Level Synthesis

High level synthesis (a.k.a. behavioral or architectural synthesis) is a technique to convert a behavioral description of a system into a register transfer level design. The HLS methodology takes behavioral description of a system (such as processors) and converts it into register transfer level design (having elements such as ALU, muxes, demuxes, registers, etc.). The first step of the HLS is to convert behavioral description in the form of a programming language or hardware description Language into an internal representation. Two types of internal representation are typically used during HLS: parse tree and graphs [8,9]. In our proposed methodologies we have utilized graphical representation. The graphical representation can further be in the form of a data flow graph (DFG) or a control data flow graph (CDFG). The next two steps of high level synthesis namely scheduling and allocation are closely related to each other [8, 9, 46]. Scheduling step is responsible for assigning the operations to the control steps, while allocation step assigns the hardware resources to the operations i.e. functional units, storage and communication elements (such as muxes, demuxes, buses). The aim of scheduling is to minimize the number of control steps or time required for completion of the program, while the aim of allocation is to minimize the number of hardware resources required for complete execution of the program. Once the scheduling and allocation steps are completed, binding step is executed. The aim of binding is to determine the size of the switching elements (muxes/demuxes) of the datapath. Once binding step is completed, the register transfer level datapath is obtained. However, controller to drive the datapath (as per the schedule's requirement) is yet to be built.

A controller is typically designed as hardwired or micro-coded. In hardwired controller design a control step corresponds to a state in the finite state machine (FSM). Similarly, in a micro-coded controller, a control step corresponds to a microprogram step [8, 9]. Subsequently, the controller is optimized and synthesized. Once the controller and datapath of design are available in the form RTL design. The lower level design steps are performed to obtain the 'market ready' integrated circuit as shown in the fig. 1.1.

1.4. Transient fault resiliency/tolerance of IP cores

As the transistor scaling continues in the sub-nanometer range, the amount of charge stored in a circuit's nodes continues to shrink, enhancing its susceptibility to reliability concerns such as multi-cycle and multi-unit transient fault [10, 11]. A transient fault may occur when a particle with moderate energy strikes a circuit node. As the amount of charge stored in a node is reduced, so does the critical charge required for changing the logic level of a circuit, thereby increasing chances of transient fault due to particle with moderate energy. Additionally, a particle with moderate energy that was capable of affecting a single node in previous technology scale, can affect more than one node placed within the same nanometer area in subsequent technology scales (spatial effect) [27]. Therefore, the resulting impact of transient fault could affect multiple hardware units placed in the neighborhood. This spatial effect of TF is termed as multi-unit transient fault.

Similarly, as a result of continuous technology scaling, the supply voltage of the device and clock-cycle time is decreasing (frequency is increasing). Therefore, the temporal effect of a single particle strike will last for multiple clock cycles in current and future technology scales [12, 17]. Hence, methodologies are required to tackle both multi-cycle (temporal) as well as multi-unit (spatial effect) of single event transient. The thesis proposes novel solutions to these problems.

1.5. Security of IP cores

In past few years, the globalization of the market has presented several opportunities for growth. However, globalization comes with its own set of drawbacks. As the number of components of a device that are manufactured outside the homeland continue to increase, the threat of a malicious attack is also increasing. Further, the lack of strict laws for punishing attackers, has resulted in higher vulnerability against these security threats. Traditionally, intellectual property were protected using techniques / tools such as patents, trademarks, copyright, trade secret, etc. However, these methodologies are either not applicable or are inefficient in protecting IP cores of digital systems [5, 13].

An IP core is vulnerable against various threats such as IP piracy, IP overbuilding, trojan insertion, etc. Hence, methodologies are required to protect IP cores against these threats. The methodologies presented in this thesis provides protection / security to IP cores against these threats as discussed in upcoming chapters.

Although most of the approaches, either address only security or only reliability. However, negative bias temperature instability based accelerated aging attack belongs partially to both reliability as well as security domain [14, 15]. The thesis proposes novel solutions to these problems.

1.6. NBTI stress analysis based accelerated aging attack on IP cores

Aging is a natural process of any electronic device. As a result of it, the performance of aged systems become un-reliable. Natural aging is a reliability concern that can be ***accelerated*** by a malicious attack that aims to reduce the life-span of the device [15]. This type of attack is known as accelerated aging attack.

Negative bias temperature instability is a physical phenomenon observed in metal oxide semiconductor field effect transistors (MOSFETs). NBTI is a major factor contributing to natural aging process. A malicious attacker can accelerate the aging of third-party IP core by applying input vectors causing maximum performance degradation when the device is not in active use. Thereby, device is degraded at maximum rate in inactive (standby mode) state. Thus, causing maximum degradation without detection (as testing and validation is typically performed in active states). This calls for methodology to identify presence of accelerated aging attack in IP cores. The thesis has proposed a novel methodology to perform NBTI stress analysis on DSP IP cores, that can further be applied to predict/identify the presence of accelerated aging attack on DSP IP cores.

1.7. Organization of thesis

The upcoming chapters of the thesis are organized as follow: Chapter 2 presents state-of-art with respect to proposed methodologies. Chapter 3 presents the proposed methodology to provide **simultaneous resiliency**

against multi-cycle temporal and multi-unit spatial effect of single event transient in DSP IP cores. Chapter 4 presents the proposed methodology to provide **simultaneous tolerance** against multi-cycle temporal and multi-unit spatial effect of single event transient for *data intensive applications*. Chapter 5 presents the proposed methodology to generate a low-cost (low-area, low-delay) **optimized** DSP IP core simultaneously tolerant against multi-cycle temporal and multi-unit spatial effect of transient fault for *loop-based control-intensive applications*. Chapter 6 will present presents a methodology to generate low-cost, highly-secure, logic obfuscated DSP IP cores to provide security against key-sensitization based attacks. Chapter 7 presents methodology to analyze effect of NBTI stress on DSP IP core and identify the presence of accelerated aging attack. Chapter 8 presents computational forensics engineering based methodology to resolve ownership of DSP IP core. Chapter 9 presents the experimental results of the proposed methodologies and compares them with their respective state-of-the-arts. Chapter 10 concludes the thesis and briefly discusses the future work.

Chapter 2

State of the art

This chapter discusses state-of-the-art related to the proposed methodologies presented in this thesis. The first section presents state-of-the-art on transient fault (TF) reliability. The second section presents approaches related to security of DSP IP cores. The third section presents state-of-the-art on NBTI stress analysis of DSP IP cores. The fourth section describes the objective of this thesis. The fifth section summarizes the contributions of this thesis.

2.1. State of the art on transient fault security/tolerance of an IP core

As discussed in previous chapter, a transient fault may occur due to particle strike. Reliability against transient fault can be achieved either through security (resiliency) or tolerance. A **security** mechanism aims to **detect** occurrence of transient fault in a circuit. However, it cannot prevent the impact of transient fault from affecting the correct functionality of the circuit. On the other hand, a **tolerance** mechanism aims to preserve **correct** functionality of the circuit. In other words, a tolerant IP core guarantees generation of *correct* output in the presence of transient fault. Whereas, a secure IP core *only detects* the occurrence of transient fault but cannot guarantee generation of correct output in presence of transient fault.

State-of-art on transient fault security: methodologies for creating transient fault secured circuits can be designed at various levels of design abstractions. A few approaches such as [16], [17], and [18] consider transient fault security at behavioral level. However, none of these approaches provide simultaneous security against multi-cycle temporal and multi-unit spatial effect of transient fault.

Multi-cycle transient fault security: The approaches presented in [16-18] have adopted a dual modular redundancy (DMR) based technique for detecting concurrent error due to transient fault. The primary motive of the DMR structure is to isolate the impact of the transient fault in one of the modules, such that the other unaffected module could produce correct output. Hence, when the outputs of two modules are compared, a difference indicates

the occurrence of transient fault in the device. However, there is no technique to identify which one of these two modules has produced the correct output. Hence, only detection is possible through DMR based approaches.

The approach presented in [17] is more sophisticated than [16, 18]. This is because in [16, 18], at-least two-distinct hardware were required for ensuring security, which is not mandatory in [17]. The methodology presented in [17] ensures transient fault detection using a single hardware resource of a particular type. All these techniques consider *only* multi-cycle temporal effect of transient fault. However, they do not consider spatial effect of single event transient.

Multi-unit transient fault security: Most approaches in the literature consider multiple event transient fault on memory. However, a few approaches such as [19, 20] consider effect of multiple transient fault at logic level. Nonetheless, these approaches do not consider security at behavioral level.

The proposed approach presents a novel methodology to provide simultaneous security against *multi-cycle temporal* and *multi-unit spatial* effects of single event transient on DSP IP cores generated using high level synthesis.

State-of-art on transient fault tolerance:

Multi-cycle transient fault tolerance: There is only one work that presents a technique to create a multi-cycle transient fault tolerant design using high level synthesis [12]. *However, it fails to provide either security or tolerance against spatial effect of transient fault.*

Multi-unit transient fault tolerance: There is no technique present in the literature to generate multi-unit TF tolerant design using high level synthesis. However, the techniques such as [19], [20] are present in the literature that only considers security (no tolerance) against multi-unit spatial effect of transient fault. The approaches [19], [20] do not consider multi-cycle temporal effect of TF. Further, these approaches do not take measures to reduce design overhead and are not applicable on loop-based applications.

This thesis presents novel techniques for generating a low-cost DSP IP core that is simultaneously tolerant against multi-cycle temporal and multi-unit

spatial effect of single event transient for loop-based control intensive and non-loop based data intensive DSP applications.

2.2. State of the art on security of IP core

An IP core is vulnerable against several security threats such as IP piracy, IP overbuilding, false claim of ownership, Trojan insertion etc. To tackle these security threats, several approaches are present in the literature such as IP metering, structural obfuscation, functional obfuscation, etc. However, in this section, we only discuss the state-of-the-art approaches that are closely related to our proposed methodologies for ensuring security of IP cores i.e., functional obfuscation and hardware watermarking of DSP core.

State-of-art on functional obfuscation: The aim of functional obfuscation is to protect an IP core from a malicious attacker present in the third-party fabrication facility. Functional obfuscation (a.k.a. functional locking) is a technique that locks an IP core by inserting locking units (such as logic gates, multiplexers/demultiplexers). Thereby, only the person who knows the valid key can unlock the IP core. The state-of-the-art functional obfuscation techniques are presented in [21], [22]. Authors of [21] and [22] have presented some novel attacks based on 'key-sensitization' technique. Subsequently, they have suggested few security features that can enhances resiliency against key-sensitization based attacks.

The proposed functional obfuscation methodology enhances resiliency against key-sensitization attacks with the help of novel locking units termed as 'IP functional locking blocks (ILBs)'. The proposed ILBs are 8-key bit (per ILB) intertwined structures of many logic gates such as AND, NAND, NOT, XOR, XNOR, etc. On the other hand, function obfuscation technique of [21], [22] uses only XOR and/or XNOR gates as locking units (1-key bit per locking unit). The novel security features of the proposed ILBs enormously enhances resiliency against 'key-sensitization' attacks. Furthermore, the proposed approach integrates particle swarm optimization based design space exploration (PSO-DSE) framework for exploring low-cost functionally obfuscated design solution. However, no effort was made in [21] or [22] to obtain low-cost design solution.

State-of-art on ownership protection of DSP IP cores: digital watermarking based approaches (such as [13], [23]) were the state-of-the-art techniques to resolve ownership conflict of DSP IP core generated using high level synthesis. However, the security of a watermarked IP core can be breached using attacks such as signature tampering, reverse engineering etc. Furthermore, integral step of digital watermarking such as signature insertion can cause performance degradation, design overhead, etc. Hence, a more sophisticated signature-free methodology was required to resolve ownership of an IP core. The proposed computational forensics engineering (CFE) based methodology overcome these drawbacks as it does not depend on in-design based step such as signature insertion and there is no known attack on the proposed approach.

2.3. State of the art on NBTI stress analysis of DSP IP core

NBTI stress is a physical phenomenon observed in PMOS transistors that partially contributes to natural aging of these transistors. There was no effort made in the literature to study and analyze the impact of aging on IP cores generated using high level synthesis. The proposed approach presents a novel methodology for analyzing the aging effect of NBTI stress on performance of DSP IP core generated using high level synthesis. The phenomenon of natural aging due to NBTI stress can be utilized to perform accelerated aging attack. An attacker can accelerate the natural aging process of a transistor by continuously applying NBTI stress when the device is in inactive usage (such as in standby mode). The aim of an attacker is to accelerated aging process of a device such that it fails within the warranty period [15]. The proposed methodology to analyze the natural aging of DSP IP core can further be utilized to detect the presence of accelerated aging attack on the IP cores generated using high level synthesis.

2.4. Objective of the thesis

The objective of the thesis is to develop novel methodologies for ensuring reliability and security of DSP IP core against specific hardware threats/concerns. To achieve this aim following objectives were set:

1. To develop a methodology for generating a DSP IP core that is *simultaneously secure/resilient* against multi-cycle temporal and multi-unit spatial effect of transient fault.
2. To develop a methodology for generating a *low-cost* DSP IP core that is *simultaneously tolerant* against multi-cycle temporal and multi-unit spatial effect of transient fault for *data intensive applications.*
3. To develop a methodology for generating a *low-cost* DSP IP core that is *simultaneously tolerant* against multi-cycle temporal and multi-unit spatial effect of transient fault for *loop-based control intensive applications.*
4. To develop a methodology for generating a low-cost, highly secure, functionally obfuscated DSP IP core.
5. To develop a methodology for analyzing the aging effect of NBTI stress on performance of DSP IP core.
6. To develop a methodology for resolving ownership conflict of DSP IP core.

2.5. Summary of the contributions

This thesis presents several novel methodologies for ensuring/enhancing reliability and security of DSP IP core. In order to advance the state-of-the-art, following contributions were made:

- A novel methodology for generating a DSP IP core that is simultaneously resilient/secure against multi-cycle temporal and multi-unit spatial effect of transient fault. (publications: J7, J10, B1, C1)
  - Proposes a novel security-aware floor-planning technique / rules for providing resiliency against multi-unit spatial effect of transient fault.
  - Proposes an integrated approach for providing security simultaneously against multi-cycle temporal and multi-unit spatial effect of transient fault.
  - Presents a novel cost function for evaluating cost of the design solution based on schedule latency, chip area and wire-length.
- A novel methodology for generating a DSP IP core that is **simultaneously tolerant** against multi-cycle temporal and multi-unit

spatial effect of transient fault for **data intensive applications**. (publications: J1, B1)

- Propose novel scheduling rules for generating multi-cycle transient fault tolerant triple modular redundant (TMR) schedule.
- Propose novel tolerance-aware floor-planning rules for ensuring tolerance against multi-unit spatial effect of transient fault.
- Integrates a particle swarm optimization based design space exploration (PSO-DSE) framework for exploring low-cost transient fault tolerant design solution for *data intensive DSP applications*.
- Proposed methodology is applicable on data intensive DSP application.

- A novel methodology for generating a DSP IP core that is **simultaneously tolerant** against multi-cycle temporal and multi-unit spatial effect of transient fault for **loop-based control intensive applications**. (publications: J8, B1)
  - Integrates a *modified* particle swarm optimization based design space exploration (PSO-DSE) framework for exploring low-cost design solution for *loop-based control-intensive DSP applications*.
  - Integrates a pre-processing technique for generating optimal unrolling factor for loop-based control-intensive DSP applications.
- A novel methodology for generating a low-cost, highly secure, functionally obfuscated DSP IP core. (publications: J2, J3)
  - Proposes a novel Functional obfuscation methodology for obfuscating DSP IP cores.
  - Proposes a set of novel locking units termed as *IP functional locking blocks (ILBs)*.
  - Presents security enhancing features/properties of proposed ILBs.
  - Integrates a *modified* PSO-DSE framework for exploring low-cost obfuscated design solution.
  - Presents a novel technique for insertion of proposed ILBs.
  - Security comparison of proposed approach with state-of-art approach, shows a minimum security enhancement of 4.29 e+9 times for the tested benchmarks.

- A novel methodology for analyzing the aging effect of NBTI stress on performance of DSP IP core. (publications: J4, J6, C2)
  - Proposes a technique to identify input vector that causes maximum performance degradation due to NBTI stress on DSP IP core.
  - Proposes a methodology to analyze the effect of NBTI stress with respect to varying stress times on critical path delay of DSP cores.
  - Presents a performance comparison of stress v/s no-stress condition of DSP cores with respect to various input vector samples.
  - Presents a technique to predict the presence of accelerated aging attack on DSP IP core.
- A novel computational forensic engineering methodology for resolving ownership conflict of DSP IP core generated using high level synthesis. (publications: J5, J9, C3)
  - Proposes a novel feature-set containing ten features that can be utilized for resolving ownership conflict of an IP core.
  - Proposes novel feature extraction rules/algorithms for each of the proposed features.
  - The proposed technique incurs zero-overhead, zero-performance degradation compared to watermarking based IP core protection (due to its signature independence).

Chapter 3

Methodology for generating a DSP IP core - Simultaneously resilient/secure against multi-cycle temporal and multi-unit spatial effect of transient fault

This chapter presents a novel methodology for detecting the presence of transient fault due to temporal and spatial effects of single event transient. The first section introduces the problem. The second section provides a detailed description of the proposed approach. Subsequently, the proposed methodology is illustrated with the help of a demonstrative example in third section. Further, the advantages and disadvantages of the proposed approach are presented in the fourth section and conclusions are drawn in the fifth section.

3.1. Introduction

As discussed in earlier chapters, a transient fault (TF) may occur when a particle with moderate energy strikes a circuit. A particle with linear energy transfer (LET) value more than critical charge can change the logic state of the affected node. An example of such a particle capable of causing transient fault is 'α-particle' (present in packaging material of an integrated circuit). In the past, the impact of a single particle strike was assumed (modelled) to be capable of affecting only a single node. However, as the technology scale reaches 130 nanometer range, it becomes evident that this assumption can no longer hold true for current and future technology scales [24-27]. In future, a single particle strike is more likely to affect more than one node placed adjacent to each other [27]. Additionally, if these nodes belong to different hardware units, then all these hardware units will produce faulty outputs. This spatial impact of transient fault on more than one hardware unit is termed as multi-unit transient fault (MTF). In our proposed approach, the worst-case spatial impact of transient fault is considered as 'km-units'. The value of 'km' is estimated by the designer based on the environment in which the circuit will be deployed and fed as an input to the proposed approach.

In a manner similar to the spatial effect, the temporal effect of a single event transient is expected to last for multiple clock cycles [12, 17, 24]. This is due to factors such as input voltage scaling, increasing frequency of the devices, etc. This temporal effect of transient fault is termed as multi-cycle transient fault (MCT). In the proposed approach, the worst-case temporal effect of transient fault is considered as 'kc-cycles'. The value of 'kc' is estimated by the designer and fed as an input to the proposed methodology.

Moreover, as the technology scaling continues and the demand for smaller and faster devices increases, the design complexity has also increased. Therefore, to reduce the effort required to design complex circuits, many designers have moved to higher level of design abstraction such as architectural (a.k.a. behavioral / high) level [3-6]. Hence, novel methodologies are required at architectural level to identify the presence of temporal and spatial effect of transient fault. The proposed approach presents a novel methodology that integrates 'high level synthesis (HLS)' and 'physical design' frameworks for generating a DSP IP core that is simultaneously resilient/secure against multi-cycle temporal and multi-unit spatial effects of transient fault.

3.2. Proposed approach

This section provides a detailed description of our proposed methodology.

3.2.1. Problem formulation

Given a DSP application in the form of data flow graph (DFG) along with module library, strength of multi-cycle transient fault (kc-cycles), strength of

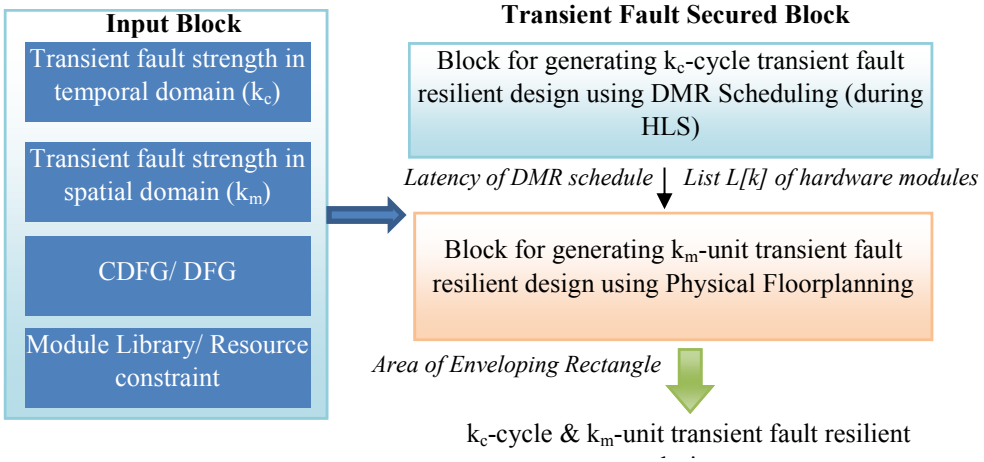


Fig.3.1. Overview of proposed transient fault security approach

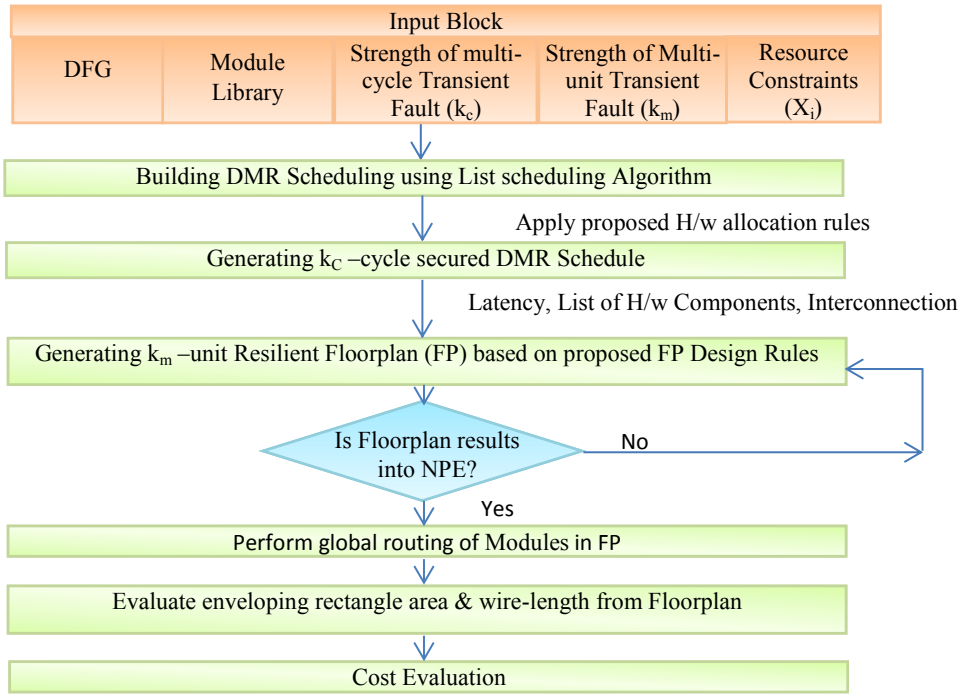


Fig.3.2 Flow diagram of proposed methodology for generating simultaneously k_c and k_m resilient DSP IP core

multi-unit transient fault (km-units), user-provided resource constraint X_i, generate a k_c-cycle and k_m-unit transient fault resilient design.

3.2.2. Overview of proposed methodology

As discussed earlier, in future technologies, transient fault occurring due to radiation strike can last for multiple cycles as well as can affect multiple hardware units placed in the neighborhood of the affected unit (node). Hence, it is necessary for future technologies to consider both the temporal and spatial effect of transient fault during the creation of transient fault resilient (secured) design. A single particle strike could simultaneously cause multi-cycle and multi-unit transient faults. However, as MCT affects in temporal domain and MTF affects in spatial domain. Therefore, domain specific independent techniques are required to detect the effect of transient fault in their respective domains. As shown in fig.3.1, the proposed approach integrates multi-cycle transient fault resilient 'high level synthesis' framework with a novel multi-unit TF resilient 'physical design' framework to generate a simultaneously MCT and MTF resilient DSP IP core design.

A detailed flow diagram of the proposed approach is shown in fig.3.2. In the initial step of proposed approach, a dual modular redundant (DMR) system is created by duplicating all the operations of DFG application. Subsequently,

these operations are concurrently scheduled based on the user specified resource constraint (X_i). The scheduled DFG (SDFG) thus obtained, along with the strength of multi-cycle transient fault (kc-cycles) are fed into a multi-cycle transient fault resiliency algorithm (adopted from [28, 17]) to obtain a kc-cycle transient fault resilient SDFG DMR. The schedule latency of kc-cycle resilient design is extracted and stored for cost/fitness evaluation in the future. Once temporal resiliency is achieved, the MCT resilient design along with strength of multi-unit TF (km-units) are fed into spatial resiliency framework. In the first step of spatial resiliency framework, a list 'L[k]' of hardware modules comprising of functional units, multiplexers/demultiplexers units etc. is generated. Subsequently, a physical level floorplan ([70]) is generated based the proposed km-unit transient fault resiliency rules. Further, global routing of modules is performed based on which wirelength is estimated. Subsequently, wirelength and rectangular chip area of the km-unit transient fault tolerant floorplan along with schedule delay (stored earlier) are utilized for evaluating the cost of the generated design solution as discussed in section 3.2.6. The upcoming sections 3.2.3 and 3.2.4 will discuss framework for multi-cycle and multi-unit resiliency respectively.

3.2.3. Methodology for generating a kc-cycle transient fault resilient design

This section provides a detailed description of the methodology for designing kc-cycle fault resilient SDFG DMR (adopted from [28, 17]). The MCT resiliency algorithm takes resource constraints (Xi), DFG application, strength of MCT (kc-cycles) and module library as inputs and produces a kc-cycle transient fault resilient DMR schedule. The initial step of resiliency algorithm is to create a DMR system by duplicating all the operations of original (input) DFG as duplicate DFG. The DMR system thus created has original unit (O^U) and duplicate unit (D^U) as shown in fig 3.3. In the next step, both O^U and D^U are concurrently scheduled (a step of HLS) based on list scheduling algorithm and the user specified resource constraints Xi. Once scheduled DMR system is generated, the hardware allocation of both the units (O^U and D^U) is performed based on the following fault resiliency conditions as stated below:

i. Allocate opn (v) $\in$ O^U and opn (v') $\in$ D^U to distinct operators (hardware units) based on availability.

ii. If unavailable, then:

Keep same assignment for v′ (as v) in D^U such that:

$$t(v') - t(v) \geq kc \quad \textbf{(3.1)}$$

iii. If the above condition (Eq. (3.1)) is false, then:

Push v′ (and its successors) ∈ D^U one CS below until Eq. (3.1) is true.

Hardware allocation of duplication unit's operations without obeying conditions (i), (ii) or (iii) may result in transient fault hazards (TFH) between similar operations of O^U and D^U. In other words, TFH occurs if:

$$t(v') - t(v) \leq kc; \text{ where } (v) \in O^U \text{ and } (v') \in D^U \quad \textbf{(3.2)}$$

The TFHs are resolved by pushing the affected operation of the duplicate unit (along with its successors) in later control steps. The pushing of operations ensures that the time interval between *v* and *v′* is greater than (or equals to) kc-cycles [28]. Hence, the temporal effect of transient fault will remain isolated in the affected module. Therefore, when a single event transient will cause a fault in one of the modules, other module will produce correct output. Thus, difference between the output of original unit and duplicate unit will indicate presence of transient fault in a DSP IP core. The outputs of the O^U and D^U are compared with the help of a special circuit as discussed in the upcoming sub-section.

Protecting the guard in DMR schedule

As shown in fig. 3.3, error detection block comprises of two stages. In the first stage, outputs of the original & duplicate units of the scheduled DMR are fed into three comparators (C1, C2 & C3). In the second stage, the output of the comparators C1, C2 & C3 are subsequently fed to a voter (V). This multi-

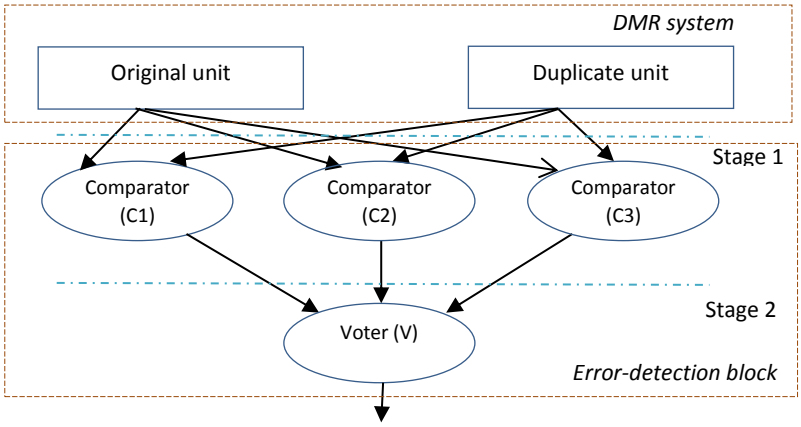


Fig.3.3. Protecting the guard: Error-detection block

stage setup (adopted from [29]) protects the transient fault resilient design against a possible vulnerability of transient fault due to a particle strike on the comparator.

The transient fault can affect the comparator(s) in two possible scenarios: (a) *faulty comparator & fault in hardware of original unit or duplicate unit*: In this scenario, any two faultless comparators will produce logic '1' as output indicating difference in outputs of original and duplicate unit. On the contrary, the faulty comparator will yield a logic '0' indicating no difference in output of O^U and D^U. Therefore, when the outputs of these three comparators are fed into voter, a logic '1' will be produced at voter output thereby, indicating presence of transient fault in the DMR system. (b) *faulty comparator & no fault in hardware original or duplicate*: In this scenario, two faultless comparators will produce logic '0' as output indicating no difference in outputs of O^U and D^U while faulty comparator will produce logic '1' indicating a difference in outputs of O^U and D^U. Therefore, when the outputs of three comparators are fed into voter, a logic '0' will be produced at voter output thereby, indicating no occurrence of transient fault in the DMR system.

Both the scenario shows that the multi-stage setup will always detect the presence of the transient fault in the circuit even if the particle strike affects a comparator. Further, note that the voter adopted in our proposed approach is tolerant against temporal effect of transient fault [30].

3.2.4. Methodology for generating a km-unit transient fault resilient design

The proposed algorithm takes kc-cycle transient fault resilient schedule and obtain the list 'L[k]' of hardware modules (functional units, interconnect units etc.). The hardware module list L[k], along with strength of multi-unit transient fault (km) are fed as input to the proposed km-resiliency algorithm. Subsequently, the hardware modules present in the L[k] are placed based on the proposed resiliency/security aware floorplanning rules:

1. Select a pair of sister operations (v & v′) in kc-cycle resilient SDFG DMR.
2. Find corresponding sister hardware functional modules (Mv & Mv′) assigned to sister operations in DMR SDFG.

Table 3.1 Conflict details of sister operations in 2-cycle transient fault resilient SDFG DMR of IIR

Operation of U_{OG}	Operation of U_{DP}	Corr. H/w of U_{OG}	Corr. H/w of U_{DP}
1	1′	M1	M2
2	2′	M2	M1
3	3′	M1	M2
4	4′	A1	A2
5	5′	A1	A2
6	6′	M2	M1
7	7′	A1	A2
8	8′	M1	M2
9	9′	A1	A2
10	10′	C1	C2

3. Place sister hardware modules in a floorplan such that they are at least km units apart i.e. S(Mv′) ≥ S(Mv) + km; where S(Mv′) and S(Mv) are the starting point of placement of modules Mv & Mv′ along x-axis or y-axis (spatial domain) in a floorplan.
4. Repeat steps 2–3 for all remaining pair of sister operations present in the kc-cycle resilient DMR SDFG.

The aim of the proposed floorplanning rules is to isolate spatial effect of transient fault within a single module of the DMR system. To this end, FP rules ensures that any pair of functional modules allocated to sister operations are bi-directionally placed at least km units apart from each other in a floorplan. This is because, if functional modules allocated to sister operations are bi-directionally placed within km units, then the spatial effect of transient fault due to a potential radiation strike may affect both the units similarly. In such a scenario, both O^U and D^U will produce same erroneous output (concurrent error). Therefore, error detection block will not be able to (distinguish between the output of O^U and D^U) detect fault. Thus, proposed floorplanning rules ensures a minimum bi-directional distance of km units between functional units allocated to sister operations.

In our proposed methodology, the strength of multiple transient fault is considered in terms of km-units. Where 1 unit = 0.768 μm has been assumed based on sample values of MTF (in nanometer range) presented in [19,31]. the strength of multiple transient fault (km) represents the worst possible impact

of MTF provided to the designer as an input. For the purpose of demonstration km = 4 is assumed. However, our proposed algorithm is applicable for any value of km. In practical scenario, the km value depends on the expected energy of the particle. (Note: in our approach we have assumed spatial impact of transient fault between functional units such as adders, multipliers, etc. but not on multiplexers / demultiplexers)

3.2.5. Wirelength estimation

Once kc-cycle and km-unit transient fault resilient floorplan is generated, wirelength is estimated as per the following equation.

$$W^{FP} = \sum_{i,j} cij \cdot dij \qquad \textbf{(3.3)}$$

Where cij is connectivity between hardware units i & j and dij is Manhattan distance between center of rectangles i & j. For evaluating Manhattan distance, the I/O connectivity is assumed to be at the center of each module.

3.2.6. Cost evaluation

In prosed approach, cost is evaluated as the normalized weighted sum of wirelength, chip area (enveloping rectangular area), and latency as shown by the following equation:

$$C_f(X_i) = \varphi_1 \frac{L^{DMR}}{L_{\max}{}^{DMR}} + \varphi_2 \frac{A^{FP}}{A_{\max}{}^{FP}} + \varphi 3 \frac{W^{FP}}{W_{\max}{}^{FP}} \qquad \textbf{(3.4)}$$

Where, $C_f(X_i)$, is the cost/fitness function of transient fault resilient design

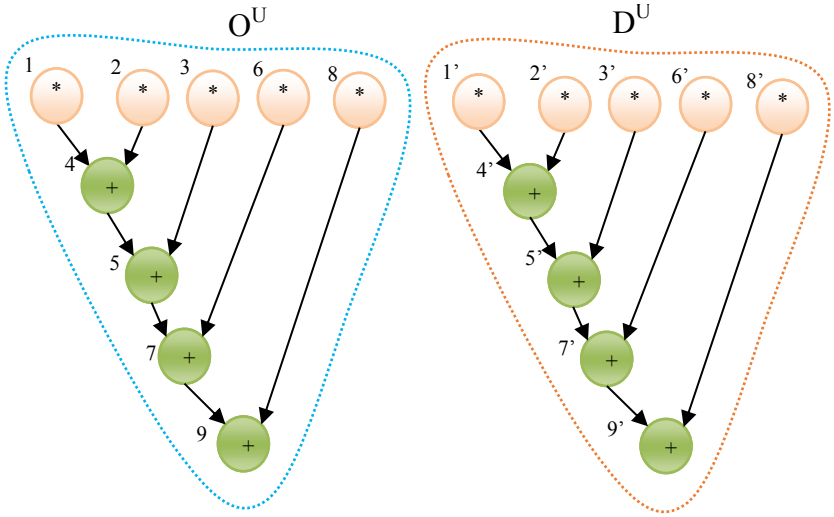


Fig.3.4. A dual modular redundant system of IIR Filter

based on resource constraint Xi; φ1 = φ2 = φ3 are the user specified weights of schedule latency, floorplan chip area and floorplan wirelength respectively. Equal weightage is assumed for φ1 = φ2 = φ3 = 0.333. L^{DMR} = latency of k_c-cycle transient fault resilient DMR schedule, based on user provided resource constraint X_i; L_{max}^{DMR} = latency of k_c-cycle transient fault resilient DMR schedule, based on maximum resources available for each type in the design space; A^{FP} = floorplan chip area of k_m-unit TF resilient floorplan based on user provided resource constraints; A_{max}^{FP} = floorplan chip area of k_m-unit multiple transient fault resilient floorplan based on maximum number of resources in the design space; W^{FP} = wirelength of FP based on user provided resource constraints; W_{max}^{FP} = wirelength of FP based on maximum number of resources in the design space.

3.3. Demonstrative example

This section provides a detailed description of the proposed approach with the help of an example of IIR filter benchmark. In demonstrative example, strength of multi-cycle and multi-unit transient faults are assumed to be kc=2 cycles and km=4 units (where, 1 unit=768 nm) respectively. Further, in the

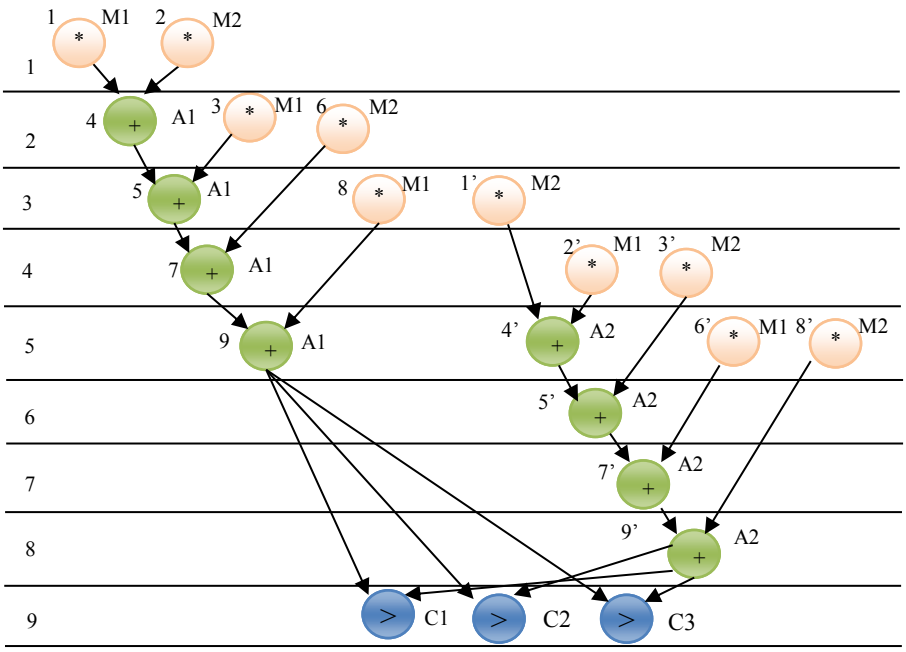


Fig.3.5. 2-cycle transient fault resilient dual modular redundant schedule of IIR Filter

demonstrative example 1 cycle or control step is equal to 100 ps. In the initial step of proposed approach, a DMR system is created by duplicating all the operations of original DFG application as duplicate unit D^U as demonstrated with IIR benchmark shown in fig. 3.4. Subsequently, scheduling (using list

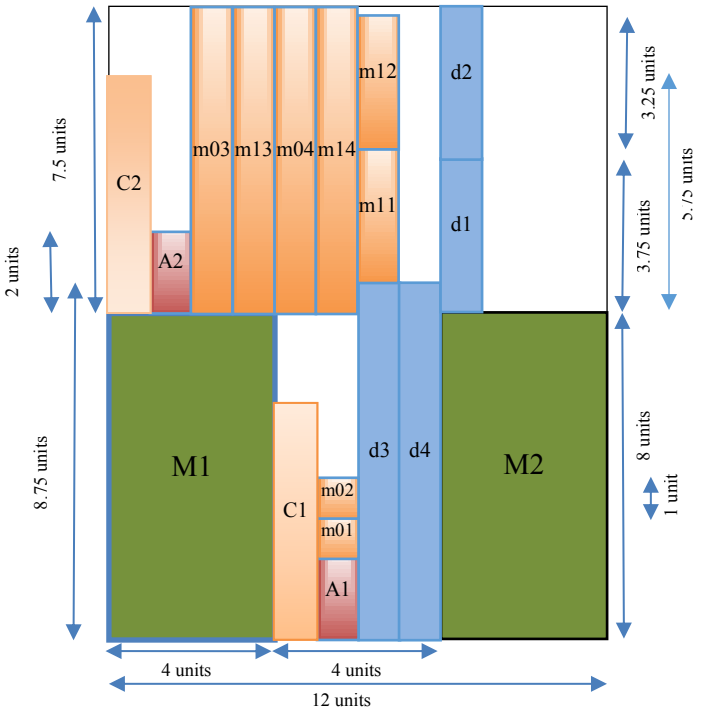


Fig.3.6. 4-unit transient fault resilient floorplan based on the 2-cycle transient fault resilient SDFG of IIR (2A, 2M)

scheduling algorithm) of the DMR system is performed based on user specified resource constraints Xi = (2A, 2M). Once Scheduled DMR system is generated proposed kc-cycle transient fault resilience rules are applied to generate 2-cycle transient fault resilient design as shown in fig. 3.5.

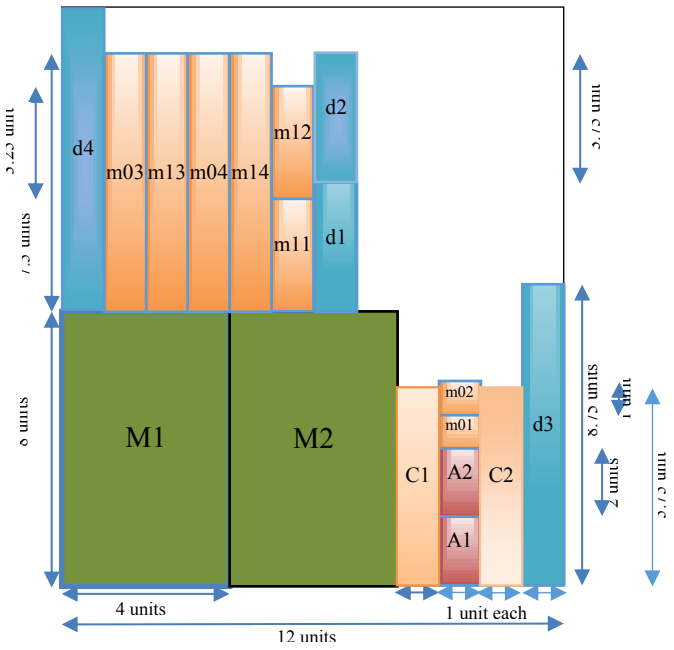


Fig. 3.7. IIR floorplan (2A, 2M) with no rules of multiple Transient fault security

Table 3.2. Library details based on 15nm NanGate

Module name	Height		width	
	nm	units	nm	units
Multiplier	6144	8	3072	4
Adder	1536	2	768	1
Comparator	4480	5.75	768	1
Subtractor	1792	2.25	768	1
2:1 MUX	832	1	768	1
4:1 MUX	2496	3.25	768	1
8:1 MUX	5824	7.5	768	1
16:1 MUX	12480	16.25	768	1
32:1 MUX	25792	33.5	768	1
1:2 demux	960	1.25	768	1
1:4 demux	2880	3.75	768	1
1:8 demux	6720	8.75	768	1
1:16 demux	14400	18.75	768	1
1:32 demux	29760	38.75	768	1

The kc-cycle transient fault tolerant design thus obtained is used to create a list of hardware modules L[k]. The list L[k] of the SDFG DMR thus obtained is

L[k] = {(Adders: A1, A2), (Comparator: C1, C2, C3), (Multipliers: M1, M2), (2:1MUX: m01, m02), (4:1MUX: m11, m12), (8:1MUX:m03,m13,m04,m14), (demux1:4:d1,d2), (demux1:8: d3,d4) }

After list L[k] is created, a table comprising of conflicting hardware resources (hardware resources allocated to sister operations within kc control steps) is generated as shown by table 1. Subsequently, the hardware modules are floorplanned based on the proposed km-unit transient fault resilient floorplanning rules discussed in section 3.2.4. (Note that the geometric dimensions of the modules based on NanGate 15 nm open source technology library [31] are shown in table 2.) For example, consider a pair of conflicting hardware M1 and M2 allocated to operation 1 and 1' respectively (within kc-cycles). Hence, to avoid transient fault impacting both the operations 1 and 1', hardware modules M1 and M2 must be placed at least km-units apart from each other. Hence, as shown in fig. 3.6, M1 and M2 are placed km=4 units distance apart from each other. Similarly, other conflicting hardware modules are placed. The floorplan thus obtained is km-unit & kc-cycles transient fault resilient. On the contrary, fig. 3.7 shows the non-resilient floorplan. In this normal floorplan hardware modules M1 and M2 are placed adjacent to each

other. Hence, in such a design although kc-cycle (temporal) resiliency is achieved. However, the design is still vulnerable to spatial effect of transient fault. Therefore, to ensure complete resiliency against transient faults, it is mandatory that the resiliency is provided against both the temporal as well as spatial effect of single event transient (SET). The proposed approach ensures resiliency against both temporal and spatial effects of SET.

3.4. Advantages and disadvantages of proposed approach at behavioral level

3.4.1. Advantages

(i) Offers lower implementation runtime than existing fault secured approaches at lower level.

(ii) Offers greater reliability (i.e. temporal & spatial transient fault aware digital design synthesis flow) than lower level techniques.

(iii) Offers automated generation of multiple alternative hardware implementations that are simultaneously resilient against multi-cycle and multi-transient fault compared to lower level techniques.

(iv) Offers flexibility to design resilient digital systems against any kc-cycle and km-unit transient fault as per user requirement compared to lower level techniques where specification of worst case transient fault range (strength) may not be possible as input.

3.4.2. Disadvantages

(i) Area, power and delay overhead may be larger compared to lower level techniques.

(ii) Lower level interconnection/wirelength/datapath details are not available much at behavioral level which makes solution cost evaluation complicated.

3.5. Summary

The proposed methodology is the first approach in the literature that simultaneously consider temporal and spatial effects of transient fault. It integrates ‘high level synthesis’ and ‘physical design’ frameworks for providing security/resilience against multi-cycle temporal and multi-unit spatial effects of transient fault. Further, the proposed approach presents novel

security-aware floor-planning rules for providing resiliency against multi-unit spatial effect of transient fault. Additionally, the proposed approach presents a novel cost function for evaluating cost of the design solution based on schedule latency, chip area and wire-length. By virtue of these novel contributions the proposed approach can generate a DSP IP core that is simultaneously resilient against multi-cycle temporal and multi-unit spatial effects of transient fault.

Chapter 4

Methodology for generating a low-cost DSP IP core that is simultaneously tolerant against multi-cycle temporal and multi-unit spatial effects of transient fault for data intensive applications

This chapter presents the proposed approach to generate a DSP IP core that will produce correct output even on the occurrence of transient fault. The first section introduces the problem. The second section presents a brief overview of the proposed methodology. The third, fourth and fifth section describes the major blocks of the proposed approach. The sixth section summarize the major contributions of the proposed approach.

4.1. Introduction

As discussed in preceding chapters, radiation induced transient fault in digital systems has become a major reliability concern. Although, detection of transient faults can be sufficient in many applications. However, only detection of transient fault is not enough for mission-critical applications. Due to criticality of the application, it is mandatory to ensure that correct output is generated even on the occurrence of transient fault.

For instance, consider mission-critical application such as aircraft control system. The aircraft control system comprises of important sub-systems such as computers (involving processors), sensors and actuators. The criticality of these control systems mandates ensuring correct operation of processing cores such as application specific processing (ASPs) cores or integrated circuits (ASICs) even on the occurrence of transient fault. Moreover, due to typical working environment of aircrafts, they remain exposed to radiations that may result into transient faults. Further, due to demand of high operational speeds (high frequency), low area, low power application specific processors in the aerospace systems. The chances of temporal effect of transient fault lasting for multiple cycles has increased manifold. Similarly, the chances of spatial effect of transient fault affecting multiple units placed in the neighborhood has also increased. Hence, it is mandatory to consider both the temporal as well as

spatial impact of transient fault while designing applications for mission-critical systems.

The proposed approach presents a novel methodology for generating a '*low cost optimized transient fault tolerant hardware against multi-cycle (temporal) and multi-unit (spatial) effect of transient fault for data-intensive digital signal processing (DSP) applications'*.

4.2. Proposed approach

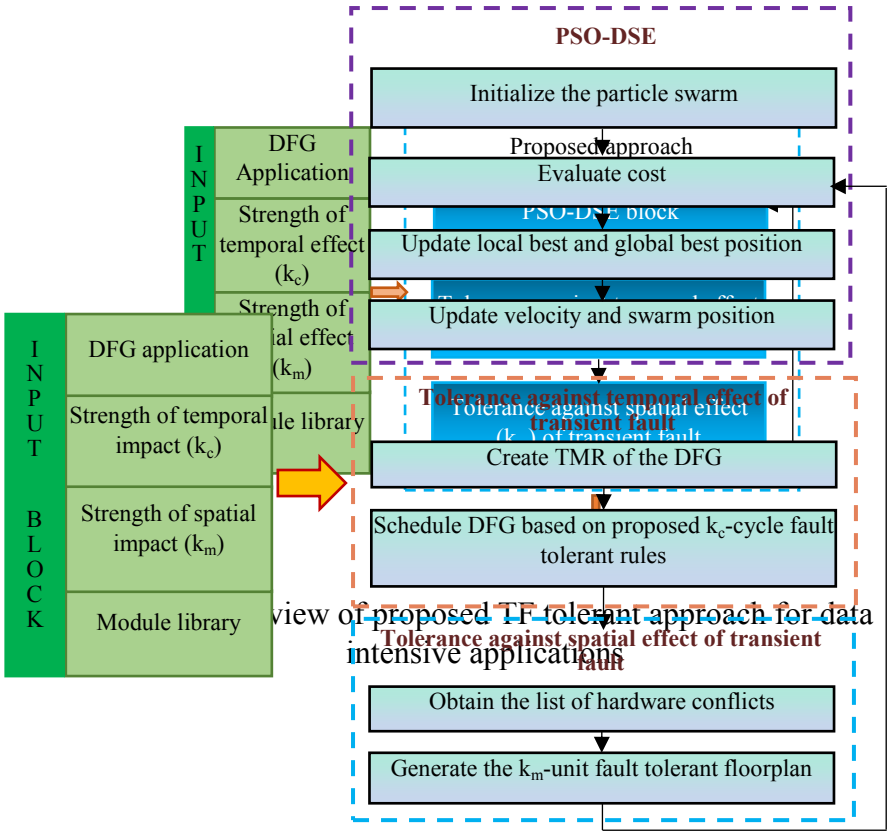
This section provides a brief overview of our proposed methodology.

4.2.1. Problem formulation

Given a data intensives DSP application in the form of data flow graph (DFG) along with module library, strength of multi-cycle transient fault (kc), strength of multi-unit transient fault (km), as inputs, generate a k_c-cycle and k_m-unit transient fault tolerant low-cost design solution as output.

4.2.2. Overview of proposed methodology

As shown in fig 4.1, the proposed methodology comprises of three major components. The first component particle swarm optimization-based design space exploration (PSO-DSE) is primarily responsible for generating low-cost design solution. The second component is responsible for providing tolerance against temporal effect of transient fault. The third and the last component provides tolerance against spatial effect of transient fault.



view of proposed TF tolerant approach for data intensive applications

Fig.4.2. Flow graph of the proposed TF tolerant methodology for data intensive applications

As shown in fig.4.2, The first step of proposed approach is to initialize the particle swarm [32, 33]. Subsequently, cost along with PSO-DSE parameters such as velocity, local best and global best are initialized. Afterwards, for each particle of the swarm, a triple modular redundant (TMR) system is created, and the proposed kc-cycle transient fault tolerant rules are applied to obtain kc-cycle transient fault tolerant schedule. The latency of the schedule thus generated, is stored for cost evaluation. Subsequently, a list of conflicting hardware is created, and proposed km-unit fault tolerant design rules are applied to obtain km-unit transient fault tolerant floorplan. The overall system thus generated is kc-cycle and km-unit transient fault tolerant design. The rectangular floorplan (chip) area thus obtained is stored for cost evaluation. Further, the cost of the transient fault tolerant design is evaluated and PSO-DSE parameters (local best, global best, velocity, particle's position) are updated. The process is repeated till one of the PSO-DSE termination criteria is met [33,32]. The optimal design solution thus explored is the low-cost kc-cycle and km-unit transient fault tolerant design solution.

The upcoming sections describe major components of proposed methodology in detail.

4.3. Proposed Methodology for generating a kc-cycle transient fault tolerant design

This section provides a detailed description of the proposed methodology for designing kc-cycle transient fault tolerant scheduled DFG (SDFG) TMR system. The aim of the proposed methodology is to isolate the impact of transient fault in any one of the three modules (copy) of the TMR system such that remaining two modules (copies) should function correctly even in the presence of transient fault. Hence, when a voter is applied to the TMR system then voter will always vote-in the correct output.

The proposed algorithm takes resource constraints (Xi), DFG application, strength of multi-cycle transient fault (kc) and module library as inputs and produces a kc-cycle transient fault tolerant TMR schedule. The initial step of proposed approach is to create a triple module redundant system by copying all the operations of original (input) DFG (O^C) as duplicate copy (D^C) and triplicate copy (T^C) as shown in fig 4.3. Subsequently, scheduling and allocation of TMR system is performed based on resource constraints (particle position Xi, produced from PSO-DSE block) using the proposed kc-cycle transient fault tolerant scheduling and allocation rules.

The temporal effect of transient fault may cause hardware conflicts during scheduling and allocation. The hardware conflict arises when a hardware resource allocated to an operation of a copy is re-allocated to another operation of its cloned copies within kc-cycles. The proposed kc-cycle transient fault tolerant scheduling and allocation rules to resolve these hardware conflicts are:

a. Hardware resource (R) allocated to an operation of a copy can be re-allocated to an operation of the same copy within k_c control steps (cycles).
b. Shift operation of a copy if no hardware resource can be allocated without conflicts. Thus, allocations are made based on the following:

i. Resource 'R' allocated to an operation of O^C ($v \in O^C$) can be re-allocated to an operation of D^C ($v' \in D^C$) or an operation of T^C ($v'' \in T^C$) only after a distance of k_c control steps (cycles).

   i.e. $t(v')-t(v) > k_c$, and

   $t(v'')-t(v) > k_c$

ii. Resource 'R' allocated to an operation of D^C ($v' \in D^C$) can be re-allocated to an operation of O^C ($v \in O^C$) or an operation of T^C ($v'' \in T^C$) only after a distance of k_c control steps (cycles).

   i.e. $t(v)-t(v') > k_c$, and

   $t(v'')-t(v') > k_c$

iii. Resource 'R' allocated to an operation of T^C ($v'' \in T^C$) can be re-allocated to an operation of O^C ($v \in O^C$) or an operation of D^C ($v' \in D^C$) only after a distance of k_c control steps (cycles).

   i.e. $t(v)-t(v'') > k_c$, and

   $t(v')-t(v'') > k_c$

Proposed scheduling and allocation rules ensure fault isolation within a single copy i.e., a single particle strike causing transient fault in a copy (O^C, D^C or T^C) of the TMR system will not affect the remaining two copies. Hence, even in the presence of (temporal effect of) transient fault due to a single particle strike, two copies will always produce correct output thus voter will ensure correct output is always produced as final output of the TMR system. The delay of the kc-cycle transient fault tolerant design thus generated is stored for future utilization during cost evaluation.

4.3.1 Demonstrative example of proposed methodology for generating a kc-cycle transient fault tolerant design

This section illustrates proposed kc-cycle transient fault tolerant scheduling and allocation rules with the help of an example of DWT DFG benchmark. For the demonstrative purpose, the realistic delay value of one control step is taken as 100 ps [24]. Further the values of area and delay of hardware resources are based on 15nm technology open source NanGate library [31]. Additionally, for demonstrative purpose strength of transient fault is assumed to be (kc =) 4 control steps/cycles (equivalent to 400 ps) as adopted from [24].

However, note that the proposed approach is applicable for any other kc values.

Fig. 4.3 shows a basic TMR system of DWT benchmark. The proposed kc-cycle transient fault tolerant scheduling and allocation rules are applied on the TMR system to obtain a 4-cycle transient fault tolerant scheduling based on particle

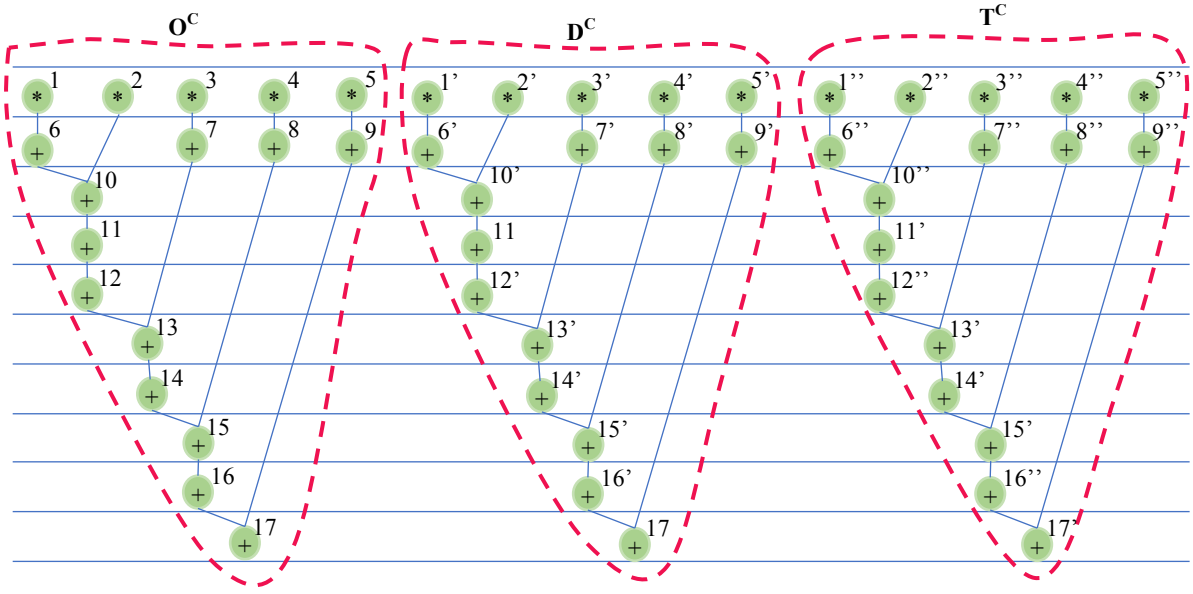


Fig.4.3. Un-timed TMR system for DWT DFG benchmark

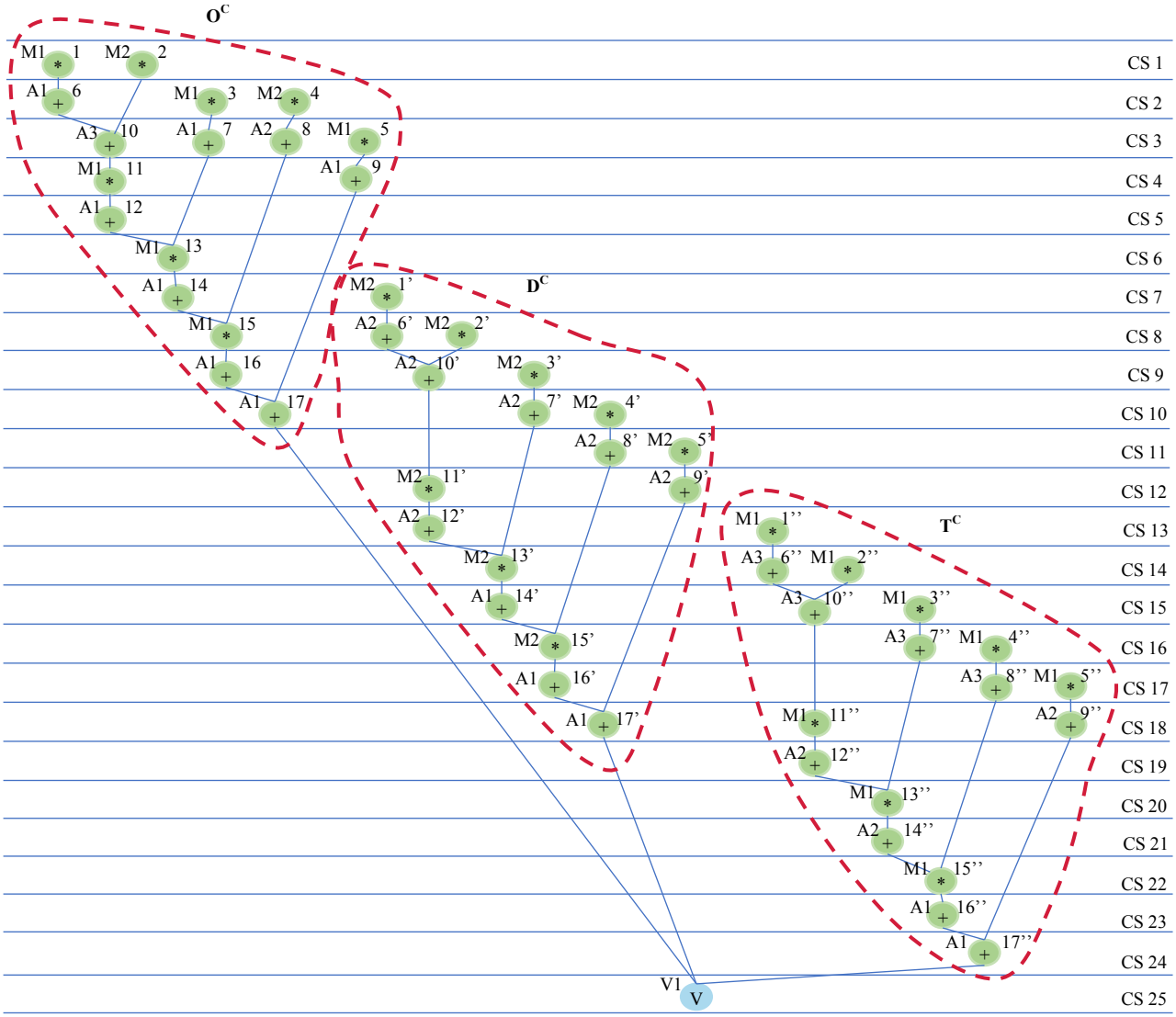


Fig.4.4. 4-cycle TF tolerant schedule of DWT DFG for particle position X_i = {3A, 2M}

position Xi = {3A, 2M} as shown in fig.4.4. The proposed rule 'a' permits a hardware resource allocated in pervious control steps to an operation of a copy to be re-allocated within kc cycles to another operation of same copy. This is because fault affected hardware will perform operations of same copy within kc cycles, hence fault will remain isolated in the same copy and will not propagate to other copies. Further, it results in better hardware resource utilization leading to reduction in delay of the scheduled DFG. Thus, fault isolation within the same copy is ensured as long as rules *b* is also satisfied. For example, rule 'a' permits hardware M1 allocated to opn 1 (of O^C) to be re-allocated to opn 3 of the same copy within kc-control steps/cycles. As per proposed rule *'b'*, opn 1' of D^C has been shifted to CS7 since no allocation was possible due to hardware conflicts. Further as per rule *b i.,* hardware resource A1 allocated to opn 17 of O^C at CS10 is re-allocated to opn 14' (of D^C) at CS15 only after 4 cycles (control steps). Similarly, M1 allocated to opn 15 of O^C at CS8 is re-allocated to operation 1'' of T^C at CS13 only after 4-cycles. Additionally, according to rule *b.ii.*, hardware A2 allocated to opn 9' can only be re-allocated to opn 9'' (of T^C) in CS 18 after 4 cycles. Further, according to rule *b.iii.,* M1 allocated to operation 1'' (of T^C) could only be re-allocated to an operation of D^C or O^C after 4 cycles. Thus, M1 allocated to 1'' could not be re-allocated to 13' or 15'.

4.4. Proposed Methodology for generating a km-unit transient fault tolerant design

The proposed methodology for generating a km-unit transient fault tolerant design takes kc-cycle fault tolerant TMR system along with strength of spatial effect of transient fault (km) as input and generates kc-cycles & km-unit fault tolerant floorplan as output.

The proposed methodology considers spatial effect of transient fault in term of hardware conflicts. A hardware conflict due to spatial effect occurs when a hardware resource allocated to an operation of a copy is placed within km-unit distance to any hardware resource allocated to an operation of remaining two copies within kc-CS (cycles). In such a scenario if two hardware resources allocated to different copies are placed less than km-unit to each other then,

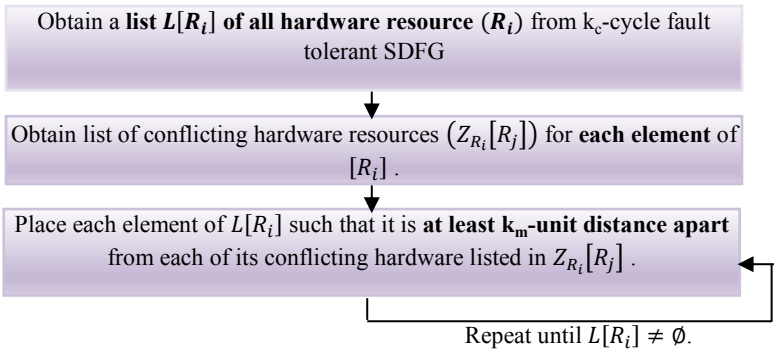


Fig.4.5 Proposed k_m-unit transient fault tolerant floorplanning rules

fault may propagate from one copy to another due to spatial effect of transient fault. Hence, more than one copy will generate incorrect output leading to incorrect output by voter. Therefore, resolving hardware conflicts due to spatial effect of transient fault is important to provide complete tolerance against transient faults. (Note that the voter utilized in the proposed approach is transient fault tolerant [30]).

As shown in fig. 4.5, the first step of proposed methodology is to obtain list of all hardware resources [Ri] present in kc-cycle fault tolerant design. In the next step, a list of conflicting hardware ($Z_{R_i}[R_j]$) due to spatial effect of transient fault is generated for all the resources present in the list [Ri]. Subsequently, the hardware resources are placed during floorplanning such that each resource Ri is placed at least km-unit distance apart from its conflicting resources Rj. These steps are repeated till all the resources are placed. The floorplan thus obtained is kc-cycle and km-unit transient fault tolerant floorplan.

4.4.1 Demonstrative example of proposed methodology for generating a km-unit transient fault tolerant floorplan

This section illustrates proposed km-unit transient fault tolerant methodology with the help of an example of DWT DFG benchmark. In the initial step, list of all hardware resource is obtained from 4-cycle transient fault tolerant TMR system (discussed earlier in section 3.2.4) as L[R] = {M1, M2, A1, A2, A3}. Subsequently, for each of the hardware resources, list of conflicting hardware is created. For instance, consider hardware resource M1, the M1 allocated to an operation 1 of OC at CS 1 will be in conflict with all the hardware allocated to any operation of DC or TC within kc = 4 cycles. However, there is no other

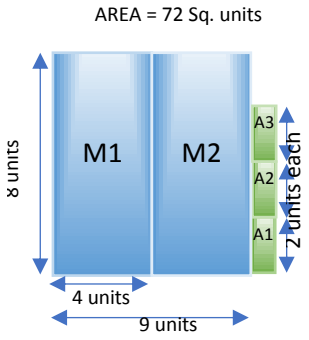


Fig.4.6. Non-tolerant Floorplan of DWT benchmark

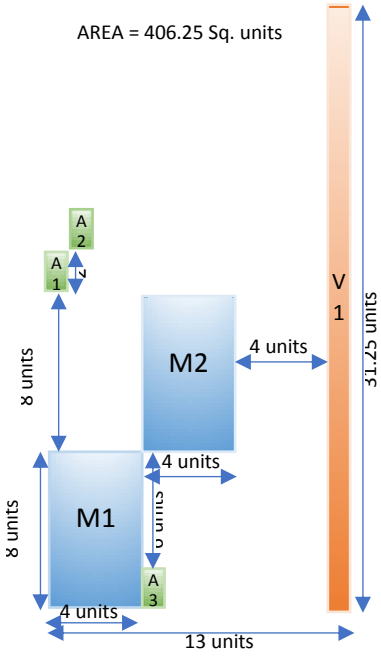


Fig.4.7. kc=4 and km=4 fault-tolerant floorplan of DWT benchmark

operation of DC or TC scheduled till CS5. Hence, for M1 allocated to operation 1 of OC there is no conflict. Similarly, M1 allocated to opn 3 at CS2 has no conflict. However, M1 allocated to opn 5 of OC at CS3 conflicts with M2 allocated to opn 1' of DC. likewise, M1 allocated to opn 11 at CS4 conflicts with M2 and A2 allocated to opn 1' and 6' of DC respectively. Similarly, other conflicts of resource M1 is evaluated and the list of conflicting hardware of resource M1 thus obtained is $Z_{M1}[R_j] = \{M2, A1, A2\}$. In similar manner list of all conflicting hardware is obtained. Subsequently, in the third and final step of the proposed km-unit transient fault tolerant approach, the conflicting hardware are placed at least km-unit (=4) bidirectional distance apart from each other. For example consider the list of conflicting hardware of M1 : $Z_{M1}[R_j] = \{M2, A1, A2\}$ and A3 : $Z_{A3}[R_j] = \{M2, A1, A2\}$. Since both the conflicting list does not contain A3, or M1 respectively. Hence, both M1 and A3 can be placed adjacent to each other as shown in fig.4.7. Similarly, as list of M1 contain A2 hence M1 and M2 are placed at least km=4 unit distance apart from each other.

On the contrary, in case of spatially non-tolerant floorplan all the hardware resources are compactly placed as shown in fig.4.6. Although, such a floorplan has lesser area compared to proposed approach, it is vulnerable to spatial

effect of transient fault. The main crux of the proposed approach is to provide tolerance against temporal as well as spatial effect of the transient fault. Additionally, the proposed approach reduces the impact of area overhead by exploring low-cost design solution with the help of PSO-DSE framework.

4.5. PSO-DSE framework for generating low-cost kc-cycle and km-unit transient fault tolerant design

This section provides a detailed description of particle swarm optimization based design space exploration PSO-DSE framework [32, 33]. The PSO-DSE framework comprises of four major steps as follows:

4.5.1 Particle encoding and swarm initialization

In the initial step of the PSO-DSE framework, particles of the swarm (Pi) are encoded as Xi = {NR_1, NR_2, …, NR_D} where Xi denotes position of i^{th} particle in the design space, NR_D represents the number of resources of type R_D in the D^{th} dimension of the design space[32, 33]. Each particle of the swarm represents number of hardware resources utilized for generating transient fault tolerant design solutions. Subsequently, particles are initialized in the design space. The first three particles (P1, P2 and P3) are initialized as:

X_1={min(R_1), min(R_2), … , min(R_D)}

X_2={max(R_1), max(R_2), … , max(R_D)}

X_3={[min(R_1) + max(R_1)]/2, … , [min(R_D) + max(R_D)]/2}

Representing minimum, maximum, and middle positions of the design space. Hence, ensuring good coverage of design space. Afterwards, the remaining particles (Pi) are initialized as:

X_i={[min(R_1) + max(R_1)]/2 $\pm\ \alpha$, … , [min(R_D) + max(R_D)]/2 $\pm\ \alpha$}

Where, min(R_D) and max(R_D) denotes minimum and maximum resource in D^{th} dimension respectively. α is a random integer between the min(R_D) and max (R_D).

4.5.2 Fitness / cost evaluation

Each particle's position in the design space represent the number of hardware resources utilized for generating kc-cycle and km-unit transient fault tolerant

design solution. Based on the varying resource configuration (particle position) fault tolerant design solutions are generated and evaluated for analyzing fitness based on the following cost function.

$$C_f(X_i) = \varphi_1 \frac{L^{FT}}{L_{\max}{}^{FT}} + \varphi_2 \frac{A^{FT}}{A_{\max}{}^{FT}} \quad \textbf{(4.1)}$$

where $C_f(X_i)$ represents the cost/fitness of fault tolerant design solution based on the (resource configuration) particle position X_i, $\Phi 1$ and $\Phi 2$ are weightage of schedule latency and area of floorplan respectively. L^{FT} is the latency of transient fault tolerant design, $L_{max}{}^{FT}$ is the maximum latency of transient fault tolerant design solution in the design space (derived using minimum number of hardware resources), A^{FT} is the enveloping floorplan chip area of the fault tolerant design solution, $A_{max}{}^{FT}$ is the maximum floorplan area of the transient fault tolerant design (derived using maximum number of hardware resources).

4.5.3 Updating local best and global best

In each iteration of the PSO-DSE framework, particle 'P' of the swarm explores some position 'Xi' in the design space. The local best denotes least cost (best fit) position explored by an individual particle 'P' of the swarm till the current iteration. Whereas, global best represents the best fit design solution explored by the entire particle population till the current iteration.

In each iteration, local best of a particle 'P' is updated if a lower cost design solution compared to current local best is explored by particle 'P' in current iteration. Similarly, in each iteration global best of entire particle swarm is updated, if a lower cost design solution compared to previous global best is explored by particle swarm in current iteration.

4.5.4 Updating Velocity and particle's position

After the local best and global best are updated, the velocity of a particle is updated using eq. 4.2.

$$V_{d_i}^{+} = \omega V_{d_i} + b_1 r_1 [R_{d_{\mathrm{lb_i}}} - R_{d_i}] + b_2 r_2 [R_{d_{gb}} - R_{d_i}] \quad \textbf{(4.2)}$$

Subsequently, the position of a particle is updated using 4.3.

$$Rd_i^{+} = Rd_i + V_{d_i}^{+} \quad \textbf{(4.3)}$$

Where $V_{d_i}^{+}$,V_{d_i} ,ω, $R_{d_{\mathrm{lb_i}}}$, $R_{d_{gb}}$, R_{d_i} ,b_1, b_2 ,r_1 and r_2 are as defined in nomenclature of this thesis ([32, 33]).

Subsequently, for the new particle positions, k_c-cycle and k_m-unit transient fault tolerant designs are generated and finesses are evaluated. This process continues till one of the termination criteria is satisfied:

1. The global best is not updated for last 10 iterations.
2. The user-defined maximum number of iterations have been executed.

The PSO-DSE process generates optimal low-cost k_c-cycle and k_m-unit transient fault tolerant design solution upon termination.

4.6. Summary

The proposed methodology is the first approach in the literature to generate DSP IP cores that are simultaneously tolerance against multi-cycle temporal and multi-unit spatial effects of transient fault for data intensive applications. The proposed approach presents novel TF tolerant Scheduling and floorplanning techniques for generating DSP IP cores simultaneously tolerant against temporal and spatial effect of transient fault. Further, the proposed approach generates low-cost design solution with the help of integrated PSO-DSE framework.

Chapter 5

Methodology for generating a low-cost DSP IP core that is simultaneously tolerant against multi-cycle temporal and multi-unit spatial effects of transient fault for loop-based control intensive applications

The previous chapter has presented the methodology for generating transient fault tolerant DSP IP core for *data intensive* applications. In this chapter we will discuss methodology for generating transient fault tolerant DSP IP core for *loop-based control intensive* applications. The chapter is organized in five sections. In the first section we will introduce the problem. In the second section we will present a brief overview of the proposed solution. The third, fourth and fifth section will describe the major blocks of the proposed solution with the help of a demonstrative example. The fifth and the last section will conclude the chapter.

5.1. Introduction

As discussed in previous chapter, it is necessary to consider tolerance against radiation induced transient faults while designing applications for mission-critical systems. Further, due to very stringent requirements such as low-power, low-area, low-delay of mission-critical systems, it is equally (if not more) important to consider *optimization* while designing *reliable* systems. The mission critical systems require both data intensive as well as control intensive applications. Therefore, although technique discussed in previous chapter generates optimal design solutions for data intensive applications, it is not applicable to loop-based control intensive applications. Hence, novel methodology is required for generating optimal designs for control intensive DSP applications.

The proposed approach presents a novel methodology for generating a '*low cost optimized transient fault tolerant hardware against multi-cycle (temporal) and multi-unit (spatial) effect of transient fault for loop-based control intensive digital signal processing (DSP) applications'*

5.2. Proposed approach

This section briefly describes major components of proposed methodology.

5.2.1 Problem formulation

Given a control intensives DSP application in the form of control data flow graph (CDFG) along with module library, strength of multi-cycle transient fault (kc), strength of multi-unit transient fault (km), generate a low-cost k_c-cycle and k_m-unit transient fault tolerant design solution.

5.2.2 Overview of proposed methodology

As shown in fig 5.1, the proposed methodology comprises of four major components namely PSO-DSE block, pre-processing block, kc-cycle tolerance block and km-unit tolerance block. The particle swarm optimization-based design space exploration (PSO-DSE) block is primarily responsible for exploring low-cost design solution. The pre-processing block takes CDFG application as input and determines the optimal unrolling factor. The kc-cycle tolerance block is responsible for providing tolerance against temporal effect of transient fault. The fourth and final block provides tolerance against spatial effect of transient fault.

As shown in fig. 5.2, The first step of proposed methodology is to perform pre-processing of the CDFG application for identifying optimal unrolling factors (UF) for the design space. Subsequently, based on the pre-processed unrolling factors, particle swarm is initialized as Xi = {NR_1, NR_2, …, NR_D,

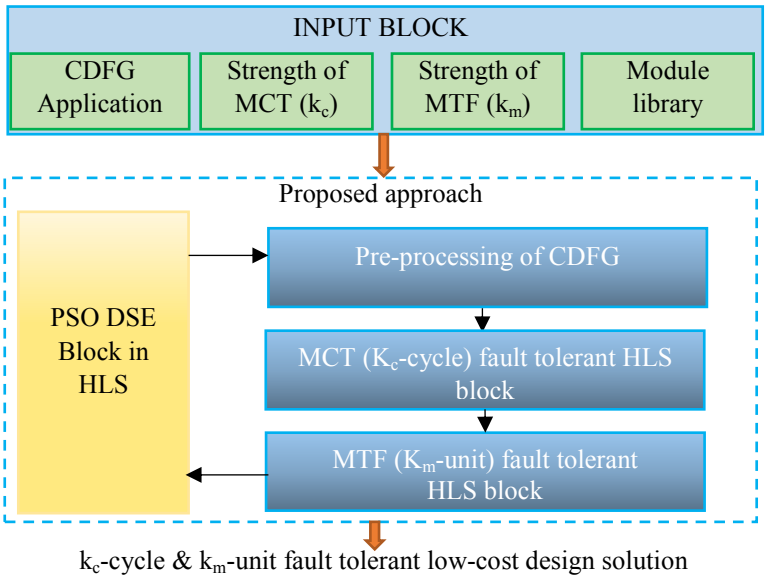


Fig.5.1. Overview of proposed TF tolerant approach for loop-based control intensive applications

UF} where Xi denotes position of i^{th} particle in the design space, NR_D is the number of resources of type R_D in the D^{th} dimension of the design space, UF is unrolling factor. Further, for each particle position Xi, CDFG application is unrolled based on unrolling factor UF. Subsequently, a TMR system of unrolled CDFG is created with respect to each particle position Xi. Afterwards, proposed transient fault tolerant rules are applied to generate kc-cycle transient fault tolerant schedule. The kc-cycle transient fault tolerant schedule thus obtained is utilized for creating a list of hardware conflicts. Subsequently, the proposed km-unit fault tolerant rules are applied for generating kc cycle and km unit transient fault tolerant floorplan. Once kc-cycle and km-unit transient fault tolerant design is generated, cost is evaluated and PSO-DSE parameters such as velocity, local best and global best are updated. The process is repeated till one of the PSO-DSE termination criteria is met. The optimal design solution thus explored is the low-cost kc-cycle and km-unit transient fault tolerant control intensive DSP application. The upcoming sections describe major components of proposed methodology in

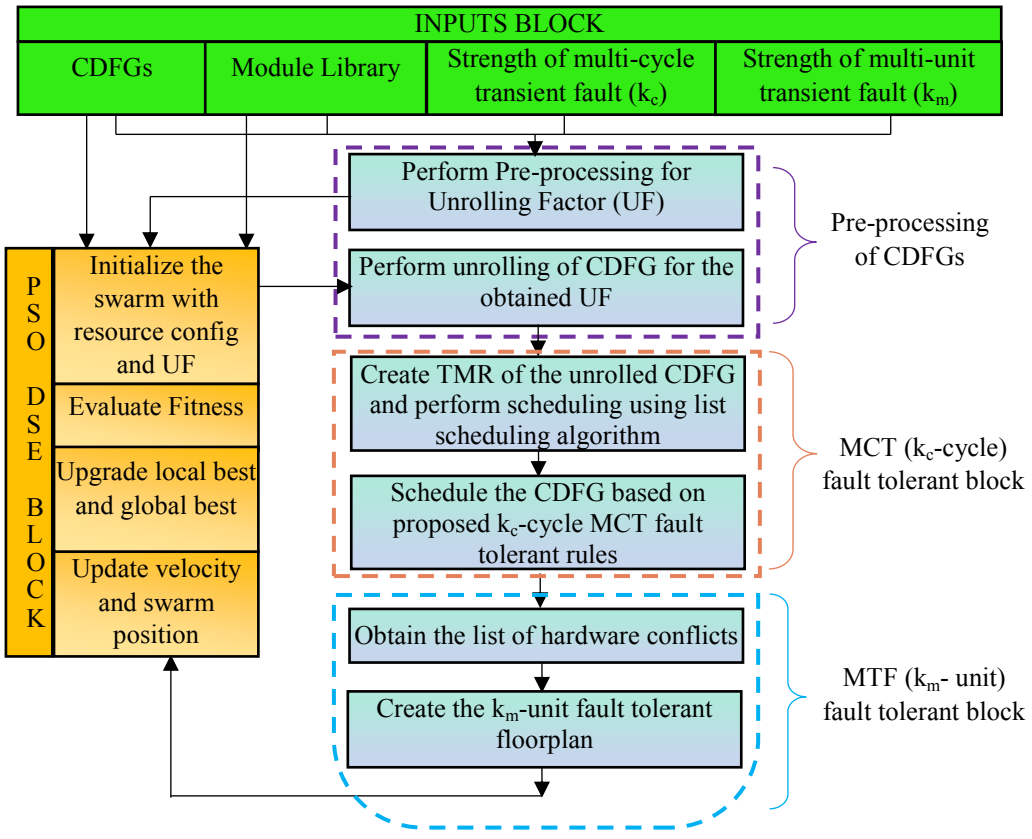


Fig. 5.2. Flow graph of the proposed TF tolerant methodology for loop-based control intensive applications

detail.

5.3. Preprocessing of CDFG

The pre-processing of CDFG application is a process by which optimal unrolling factors for the given application are determined. The pre-processing step perform optimization by removing non-optimal UFs. Thereby, reducing design space to include only optimal unrolling factors. As shown in fig. 5.2, pre-processing step comprises of two sub-steps as described below

5.3.1 Preprocessing of CDFG application for determining optimal unrolling factors

The pre-processing approach is adopted from []. The pre-processing step takes CDFG application as input and determine the desirable unrolling factors as per follo

$$desirable\ UF = ((I \bmod UF < \frac{UF}{2}) \ \&\& \ (UF <= \frac{I}{2})) \quad \textbf{(5.1)}$$

win g equation

where 'I' is total number of loop iterations and UF is unrolling factor. The UFs thus obtained are most desirable UFs as shown in [VCAL vol.2 issue2 etc. papers].

5.3.2 Unrolling of CDFG

In our proposed approach, each particle position Xi = {NR_1, NR_2, …, NR_D, UF} comprises of a desirable UF. For each Xi, CDFG application is unfolded 'UF-1' times to get unrolled CDFG. For instance, as shown in fig. 5.3, The original CDFG application (1st iteration) is unfolded once more (2nd iteration) to obtain unrolled CDFG with UF=2. The 1st and 2nd iterations are represented

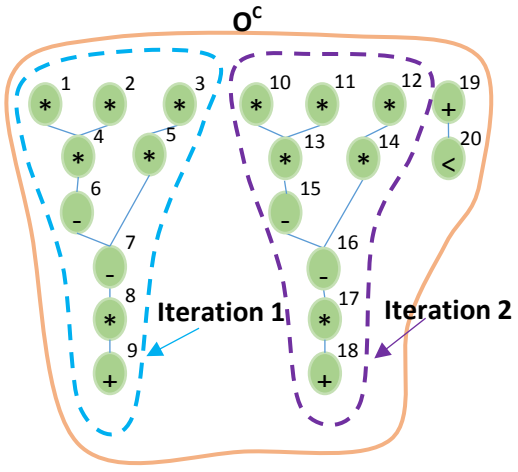


Fig. 5.3. Unrolled CDFG of differential equation benchmark for UF = 2

by light

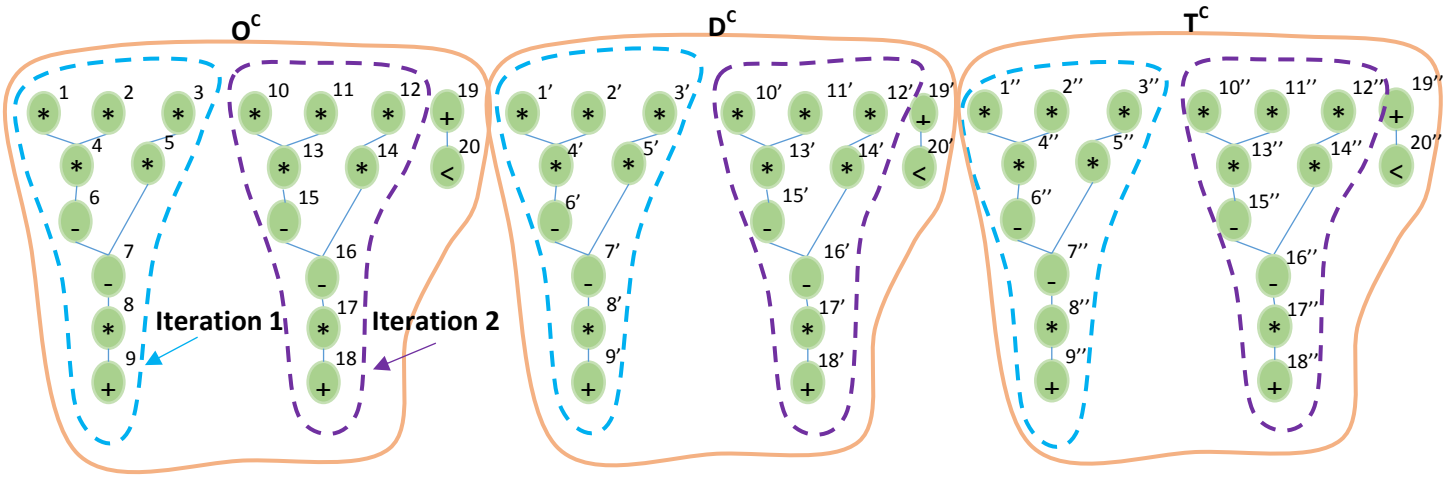


Fig. 5.4. TMR system of unrolled CDFG (UF = 2) of differential equation benchmark

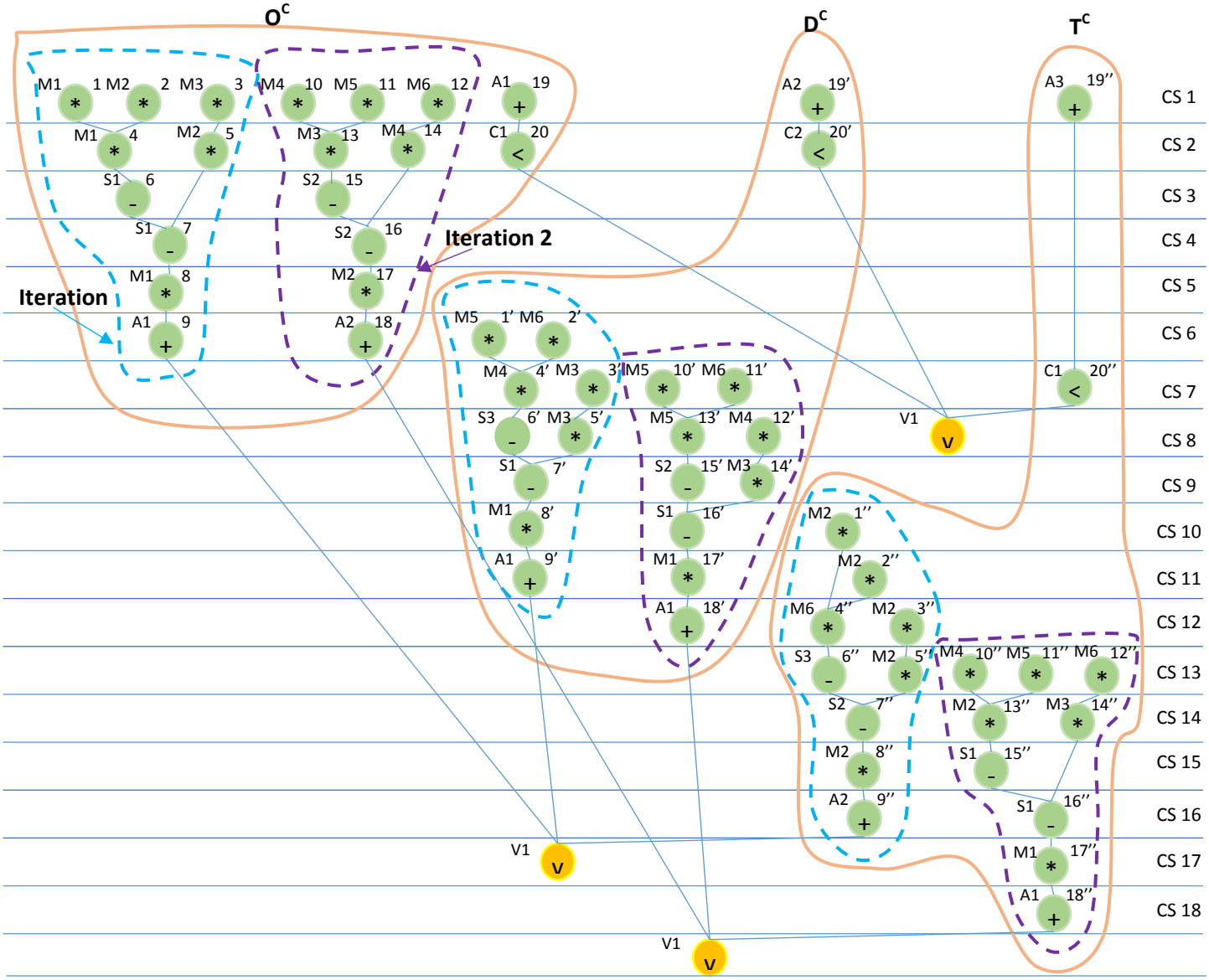


Fig. 5.5. 4-cycle TF fault tolerant SCDFG TMR of differential equation benchmark for (6M, 3A, 3S, 2C, UF=2)

blue and purple colored outlines respectively. The additional circuit comprising of an adder and a comparator is utilized for counting (incrementing) the number of iterations executed and comparing them with the maximum number of iterations (I) to be performed. This section provides a detailed description of the proposed methodology for designing kc-cycle transient fault tolerant scheduled DFG (SDFG) TMR system. The aim of the proposed methodology is to isolate the impact of transient fault in any one of the three modules (copy) of the TMR system such that remaining two modules (copies) should function correctly even in the presence of transient fault. Hence, when a voter is applied to the TMR system then voter will always vote-in the correct output. The pre-processed and unrolled CDFG thus generated is fed as input to next step of our proposed methodology.

5.4. Proposed Methodology for generating a kc-cycle transient fault tolerant design

The proposed methodology comprises of two steps as described below.

5.4.1. Creating TMR of the unrolled CDFG

The first step of kc-cycle transient fault tolerant methodology takes unrolled CDFG as input and creates a triple modular redundant (TMR) system by copying all the operations of original unrolled CDFG (O^C) as duplicate copy (D^C) and triplicate copy (T^C) as shown in fig. 5.3. The TMR system thus generated is fed into our proposed methodology for generating kc-cycle transient fault tolerant scheduled TMR system as discussed in the following sub-section.

5.4.2. Methodology for generating kc-cycle transient fault tolerant scheduled TMR system

The proposed algorithm takes D-dimensional resource configuration extracted from particle position Xi as input along with unrolled CDFG based TMR system, strength of multi-cycle transient fault (kc) and module library and produces a kc-cycle transient fault tolerant scheduled TMR system as output. The first step of proposed approach is to perform scheduling and allocation of

TMR system based on resource configuration extracted from Xi, using the proposed kc-cycle transient fault tolerant scheduling and allocation rules.

The proposed approach considers the temporal effect of transient fault as hardware conflicts. A hardware conflict arises when a hardware resource allocated to an operation of a copy is re-allocated to another operation of its cloned copies within kc-cycles. The proposed kc-cycle transient fault tolerant scheduling and allocation rules applied to resolve these hardware conflicts are:

a. Hardware resource (R) allocated to an operation of a copy can be re-allocated to an operation of the same copy within k_c control steps (cycles).
b. Shift operation of a copy if no hardware resource can be allocated without conflicts. Thus, allocations are made based on the following:
   i. Resource 'R' allocated to an operation of O^C ($v \in O^C$) can be re-allocated to an operation of D^C ($v' \in D^C$) or an operation of T^C ($v'' \in T^C$) only after a distance of k_c control steps (cycles).
      i.e. $t(v')-t(v) > k_c$, and
      $t(v'')-t(v) > k_c$
   ii. Resource 'R' allocated to an operation of D^C ($v' \in D^C$) can be re-allocated to an operation of O^C ($v \in O^C$) or an operation of T^C ($v'' \in T^C$) only after a distance of k_c control steps (cycles).
      i.e. $t(v)-t(v') > k_c$, and
      $t(v'')-t(v') > k_c$
   iii. Resource 'R' allocated to an operation of T^C ($v'' \in T^C$) can be re-allocated to an operation of O^C ($v \in O^C$) or an operation of D^C ($v' \in D^C$) only after a distance of k_c control steps (cycles).
      i.e. $t(v)-t(v'') > k_c$, and
      $t(v')-t(v'') > k_c$

c.

   i. There should be at least control steps (cycles) delay between execution of two consecutive sequential loops such that there are no conflicts:

      i.e. $T^S_{Seq2} - T^E_{Seq1} > k_C$,

ii. There should be at least control steps (cycles) delay between execution of two consecutive parallel loops such that there are no conflicts:

i.e. $T^{S}_{par2} - T^{E}_{par1} > k_C$,

iii. There should be at least control steps (cycles) delay between start of the execution of sequential loop1 and completion of parallel loop2 such that there are no conflicts:

i.e. $T^{S}_{Seq1} - T^{E}_{par2} > k_C$,

Proposed scheduling and allocation rules ensure fault isolation within a single copy i.e., a single particle strike causing transient fault in a copy (O^C, D^C or T^C) of the TMR system will not affect the remaining two copies. Hence, even in the presence of (temporal effect of) transient fault due to a single particle strike, two copies will always produce correct output. Hence, voter applied to output of TMR system will ensure correct output is always produced as final output of the TMR system. The proposed rules are elaborated in upcoming section 5.4.3. The delay of the kc-cycle transient fault tolerant design thus generated is evaluated (as discussed below) and stored for future utilization during cost evaluation.

Proposed Latency model: The latency of kc-cycle transient fault tolerant TMR (L^{TMR}) is given by following equation

$$L^{TMR} = (I\%UF) * L_{seq} + \left(\frac{I}{UF}\right)^{quotient} * L_{par} \quad \textbf{(5.2)}$$

Where, $(I\%UF)$ indicates the number of sequential loops, and $\left(\frac{I}{UF}\right)^{quotient}$ denotes number of parallel loops, L_{seq}, L_{par} denotes latency of sequential body and parallel body respectively. The L_{seq}, L_{par} are calculated as summation of 'delay of each control step of the kc-cycle fault tolerant schedule' and 'delay of strength of kc-cycle transient fault' as shown by eq. (3).

$$L_{seq/par} = \{\sum_{cs=1}^{N} Max(D(op_i),..,D(op_n),D(op_{i'}),..,D(op_{n'}),D(op_{i''}),..,D(op_{n''})\} + k_c \quad \textbf{(5.3)}$$

Where delay of a control step is evaluated as maximum value among 'delay of all the operations belonging to any copy of the TMR system'. where '$D(op_i)$',

'$D(op_{i'})$', '$D(op_{i''})$' represents delay of operation belonging to original copy, duplicate copy and triplicate copy respectively. Further, $1 \leq i \leq n$; $1' \leq i' \leq n'$; $1'' \leq i'' \leq n''$, where, i, i' and i'' = operations of original copy, duplicate copy and triplicate copy respectively. *n, n' and n''* = maximum number of nodes of original, duplicate and triplicate copy respectively; N = maximum number of control steps (cs) of the scheduled CDFG; k_c denotes the delay of k_c-cycles. Addition of k_c in the eq. (5.3), ensures k_c-cycle difference between execution of consecutive sequential/parallel loops. Hence, ensuring fault doesn't propagate within two consecutively scheduled sequential and parallel bodies. The upcoming sub-section will describe the proposed methodology with the help of an example.

5.4.3. Demonstrative example of proposed methodology for generating a kc-cycle transient fault tolerant design for control intensive DSP applications.

This section illustrates proposed kc-cycle transient fault tolerant scheduling and allocation rules with the help of an example of differential equation benchmark. For the demonstrative purpose, the realistic delay value of one control step is taken as 1000 ps for designing an application specific processor with frequency 1Ghz. Additionally, for demonstrative purpose strength of transient fault is assumed to be (kc =) 2 control steps (equivalent to 2000 ps) as adopted from [39,41,40]. Further, the values of area and delay of hardware resources are based on 15nm technology open source NanGate library [30]. However, note that the proposed approach is applicable for any other kc values.

Fig. 5.3 shows a basic TMR system of unrolled differential equations benchmark. The proposed kc-cycle transient fault tolerant scheduling and allocation rules are applied on the TMR system to obtain a 2-cycle transient fault tolerant schedule based on D-dimensional resource constrains extracted from particle position Xi = {6M, 3A, 3S, 2C, UF=2} as {6M, 3A, 3S, 2C} where UF=2 is already utilized during creation of unrolled CDFG.

The proposed rule 'a' permits a hardware resource allocated in pervious control steps to an operation of a copy to be re-allocated within kc cycles to

another operation of same copy. This is because fault affected hardware will perform operations of same copy within kc cycles, hence fault will remain isolated in the same copy and will not propagate to other copies. Further, it results in better hardware resource utilization leading to reduction in delay of the scheduled CDFG. Thus, fault isolation within the same copy is ensured as long as rules *b and c* are also satisfied. For example, rule 'a' permits hardware M1 allocated to opn 1 (of O^C) to be re-allocated to opn 4 of the same copy within kc-control steps/cycles.

As per proposed rule *'b'*, opn 1' & 2'of D^C have been shifted to CS4 since no allocation was possible due to hardware conflicts. Further, as per rule *b i.,* hardware resource M5 allocated to opn 11 of O^C at CS1 can only be re-allocated to opn 1' (of D^C) at CS4 after 2 cycles (control steps). Similarly, as per rule *b ii.,* hardware resource M5 allocated to opn 13' of D^C at CS6 is re-allocated to opn 10'' (of T^C) at CS9 only after 2 cycles (control steps). Further, according to rule *b.iii.,* M5 allocated to operation 10'' (of T^C) could only be re-allocated to an operation of O^C or D^C after 2 cycles in case re-allocation of M5 was needed.

5.5. Proposed Methodology for generating a km-unit transient fault tolerant design

The proposed methodology for generating a km-unit transient fault tolerant design takes kc-cycle fault tolerant TMR system along with strength of spatial effect of transient fault (km) as input and generates kc-cycles & km-unit fault tolerant floorplan as output.

The proposed methodology considers spatial effect of transient fault in term of hardware conflicts. A hardware conflict due to spatial effect occurs when a hardware resource allocated to an operation of a copy is placed within km-unit distance to any hardware resource allocated to an operation of remaining two

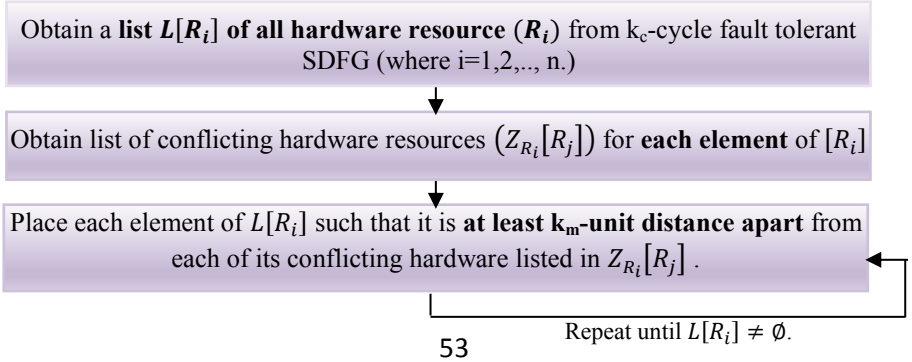


Fig. 5.6 Proposed k_m-unit transient fault tolerant floorplanning rules

copies within kc-CS (cycles). In such a scenario if two hardware resources allocated to different copies are placed less than km-unit to each other then, fault may propagate from one copy to another due to spatial effect of transient fault. Hence, more than one copy will generate incorrect output leading to incorrect output by voter. Therefore, resolving hardware conflicts due to spatial effect of transient fault is important to provide complete tolerance against transient faults.

As shown in fig. 5.5, the first step of proposed methodology is to obtain list of all hardware resources [Ri] present in kc-cycle fault tolerant design. In the next step, a list of conflicting hardware ($Z_{R_i}[R_j]$) due to spatial effect of transient fault is generated for all the resources present in the list [Ri]. Subsequently, the hardware resources are placed during floorplanning such that each resource Ri is placed at least km-unit distance apart from its conflicting resources Rj. These steps are repeated till all the resources are placed. The floorplan thus obtained is kc-cycle and km-unit transient fault tolerant floorplan.

5.5.1 Demonstrative example of proposed methodology for generating a km-unit transient fault tolerant floorplan

This section illustrates proposed km-unit transient fault tolerant methodology with the help of an example of differential equation benchmark. In the initial step, list of all hardware resource is obtained from 2-cycle transient fault tolerant TMR system (discussed earlier in section 5.4) as L[R] = {M1, M2, … , M6, A1, A2, A3, S1, S2, S3, C1, C2}. Subsequently, for each of the hardware resources, list of conflicting hardware is created. for instance, consider hardware resource M1, the M1 allocated to an operation 1 of O^C at CS 1 will be in conflict with all the hardware allocated to any operation of D^C or T^C within kc = 2 cycles. Thus, for M1 scheduled at CS1 allocated to opn 1 of O^C, the conflicting hardware in terms of spatial effect are A2, C2 (allocated to opn 19' and 20' of D^C at CS 1 and 2 respectively) and A3 (allocated to opn 19'' of T^C at CS1). Similarly, for M1 scheduled at CS2, the conflicting hardware in terms of spatial effect are C2, M5 and M6. Similarly, for M1 scheduled at CS5, the conflicting hardware are: M5, M6, M4, M3, C1, S3, S1 and S2. Further, for M1 scheduled at CS10 the conflicting hardware are M2,

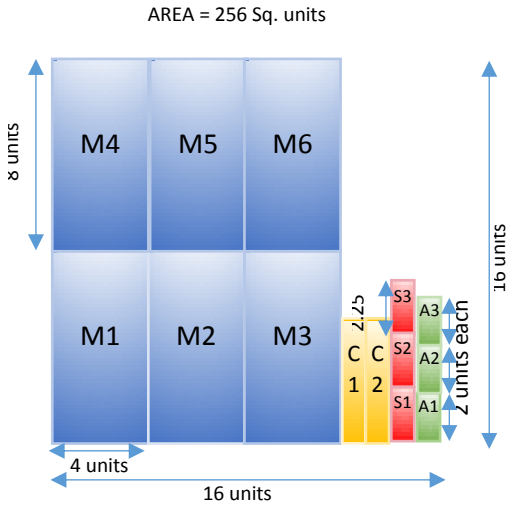


Fig. 5.7. non-tolerant Floorplan of differential equation benchmark

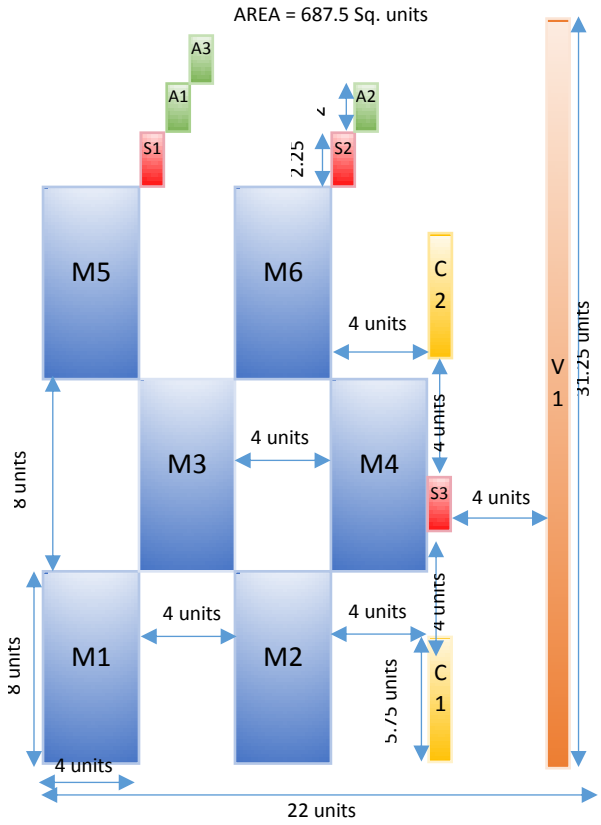


Fig. 5.8. kc=4 and km=4 fault-tolerant floorplan of differential equation benchmark

M6, S3, M4, M5, S2 and M3. Likewise, for M1 scheduled at CS11 the conflicting hardware are M2, M6, S3, M4, M5, S2, M3, S1 for all occurrences of M1 is obtained and a set of all those conflict hardware as shown below is termed as list of conflicting hardware with respect to M1:

$$Z_{M1}[R_j] = \{M2, M3, M4, M5, M6, C1, C2, S1, S2, S3, A2, A3\}$$

Similarly, the list of all the conflicting hardware with A2 is

$$Z_{A2}[R_j] = \{M1, M2, M3, M4, M5, M6, S1, S2, S3, A1, A3, C1\}.$$

Therefore, as evident from the above lists, A2 has conflict with M1 and vice-versa. Hence, A2 cannot be placed in neighborhood of M1. In similar manner, in the third and final step of the proposed km-unit transient fault tolerant floorplanning approach. The conflicting hardware are placed at least km-unit (=2) bidirectional distance apart from each other as shown in fig.5.6. Likewise, voter is also placed at km-distance apart from each hardware resource of the TMR system to avoid fault propagation from hardware resources to voter and vice-versa.

On the contrary, in case of spatially non-tolerant floorplan all the hardware resources are compactly placed as shown in fig.5.5. Hence, transient fault due to particle strike with strengths kc=2 (and km=2) affecting M1 during execution of operation 8 in CS5 will affect both M2 and M4 due to spatial effect and hence will affect operation 4', 12'. Hence, fault will propagate from original copy (O^C) to duplicate copy (D^C). Thus, voter will not be able to vote-in correct output in case of non-tolerant floorplan. Therefore, although such a floorplan has lesser area compared to proposed approach, it is vulnerable to spatial effect of transient fault. The main crux of the proposed approach is to provide tolerance against temporal as well as spatial effect of the transient fault. Hence, small area overhead could be inconsequential. However, considering the criticality of mission-critical systems, the proposed approach reduces the impact of area overhead by exploring low-cost design solution with the help of PSO-DSE framework.

5.6. Proposed PSO-DSE framework for generating low-cost kc-cycle and km-unit transient fault tolerant design

This section provides a detailed description of optimization based on PSO-DSE framework. The PSO-DSE framework comprises of four major steps as follows:

5.6.1 Particle encoding and swarm initialization

In the initial step of the PSO-DSE framework, particles of the swarm (Pi) are encoded as Xi = {NR_1, NR_2, …, NR_D, UF} where Xi denotes position of i^{th} particle in the design space, NR_D represents the number of resources of type R_D in the D^{th} dimension of the design space, UF is the pre-processed unrolling factor. Each particle of the swarm represents number of hardware resources (along with unrolling factor) utilized for generating transient fault tolerant design solutions. Subsequently, particles are initialized in the design space. The first three particles (P1, P2 and P3) are initialized at positions:

X_1={min(R_1), min(R_2), … , min(R_D), min(UF)}

X_2={max(R_1), max(R_2), … , max(R_D), max(UF)}

X_3={[min(R_1) + max(R_1)]/2, … , [min(R_D) + max(R_D)]/2,
[min(UF) + max(UF)]/2}

Representing minimum, maximum, and middle positions of the design space. Hence, ensuring good coverage of design space. Subsequently, the remaining particles (Pi) are initialized at positions:

X_i={[min(R_1) + max(R_1)]/2 $\pm \alpha$, … , [min(R_D) + max(R_D)]/2 $\pm \alpha$,
[min(UF) + max(UF)]/2 $\pm \alpha$}

Where, min(R_D) and max(R_D) denotes minimum and maximum resource in D^{th} dimension respectively. Similarly, min(UF) and max(UF) denotes minimum and maximum pre-processed unrolling factor respectively. α is a random integer between minimum and maximum value of D^{th} dimensional resource or unrolling factor.

5.6.2 Fitness / cost evaluation

Each particle's position in the design space contains the number of hardware resources in D^{th} dimension and unrolling factor. From each position, resource configuration is extracted and utilized for generating kc-cycle and km-unit transient fault tolerant design solution. The fitness of the generated design solution is evaluated using following cost/fitness function.

$$C_f(X_i) = \varphi_1 \frac{L^{FT}}{L_{\max}{}^{FT}} + \varphi_2 \frac{A^{FT}}{A_{\max}{}^{FT}} \qquad \textbf{(5.4)}$$

where $C_f(X_i)$ represents the cost/fitness of fault tolerant design solution based on the (resource configuration) particle position X_i, $\Phi 1$ and $\Phi 2$ are weightage of schedule latency and area of floorplan respectively. L^{FT} is the latency of transient fault tolerant design, L_{max}^{FT} is the maximum latency of transient fault tolerant design solution in the design space (derived using minimum number of hardware resources), A^{FT} is the enveloping floorplan chip area of the fault tolerant design solution, A_{max}^{FT} is the maximum floorplan area of the transient fault tolerant design (derived using maximum number of hardware resources).

5.6.3 Updating local best and global best

In each iteration of the PSO-DSE framework, particle 'P' of the swarm explores some position 'Xi' in the design space. The local best denotes least cost (best fit) position 'Xi' explored by an individual particle 'P' of the swarm till the current iteration. Whereas, global best represents the best fit design solution explored by the entire particle population till the current iteration.

In each iteration, local best of a particle 'P' is updated if a lower cost design solution compared to current local best is explored by particle 'P' in current iteration. Similarly, in each iteration global best of entire particle swarm is updated, if a lower cost design solution compared to previous global best is explored by particle swarm in current iteration.

5.6.4 Updating Velocity and particle's position

After the local best and global best are updated, the velocity of a particle is updated using eq. 5.5.

$$V_{d_i}^{+} = \omega V_{d_i} + b_1 r_1 [R_{d_{\mathrm{lb_i}}} - R_{d_i}] + b_2 r_2 [R_{d_{gb}} - R_{d_i}] \qquad \textbf{(5.5)}$$

Subsequently, the position of a particle is updated using eq. 3.

$$Rd_i^{+} = Rd_i + V_{d_i}^{+} \qquad \textbf{(5.6)}$$

Where $V_{d_i}^{+}$,V_{d_i} ,ω, $R_{d_{\mathrm{lb_i}}}$, $R_{d_{gb}}$, R_{d_i} ,b_1, b_2 ,r_1 and r_2 are as defined in nomenclature of this thesis ([32, 33]).

Subsequently, for the new particle positions, k_c-cycle and k_m-unit transient fault tolerant designs are generated and finesses are evaluated. This process continues till one of the termination criteria is satisfied:

3. The global best is not updated for last 10 iterations.
4. The user-defined maximum number of iterations have been executed.

The PSO-DSE process generates optimal low-cost k_c-cycle and k_m-unit transient fault tolerant design solution upon termination.

5.7. Summary

The paper presented a novel methodology that achieves fault tolerance against multi-cycle temporal and multi-unit spatial effect of transient fault in loop-based control intensive DSP IP cores generated using high level synthesis. Further, the proposed approach generates low-cost design solution for loop based CDFG applications with the help of integrated PSO-DSE framework.

Chapter 6

Methodology for generating a low-cost, highly secure, functionally obfuscated DSP IP core

This chapter presents the proposed methodology for generating low-cost functionally obfuscated DSP IP core. The chapter is organized in four sections. In the first section the problem is introduced. In the second section threat model is presented. The third and fourth section describe the proposed solution with the help of a demonstrative example. The fifth and the last section will summarize the chapter.

6.1. Introduction

As discussed in the introductory chapters, the continuous technology scaling has led to various reliability and security concerns. Further, rapid technology scaling and increasing cost of maintaining advanced fabrication facility has led to the monopoly of few advanced fabrication facilities. Majority of design houses lacks an in-house fabrication facility and must send their designs to third-party fabrication facility. This dependency of design houses on advanced fabrication facilities has enhanced security vulnerabilities such as IP Piracy, IP overbuilding, reverse engineering etc. [21, 34, 35]. Hence, methodologies are required for providing protection against these security vulnerabilities/threats.

The proposed approach provides protection against some of these threats using logic locking (a.k.a. functional obfuscation/locking). Logic locking is a technique that inserts locking units (logic gates such as AND/ OR/ XNOR etc.) such that correct output cannot be obtained until a correct key is applied to the locked circuit. A malicious attacker would be motivated to identify the correct key with the help of attacks based on reverse engineering [21, 35]. The proposed approach provides protection by enhancing the complexity of the reverse engineering. The proposed approach presents novel locking units termed as ‘IP locking blocks (ILBs)’. The proposed ILBs incorporates some novel properties to enhance its robustness against state-of-art attacks. Further, the proposed approach integrates PSO-DSE framework for generating a low-cost logically

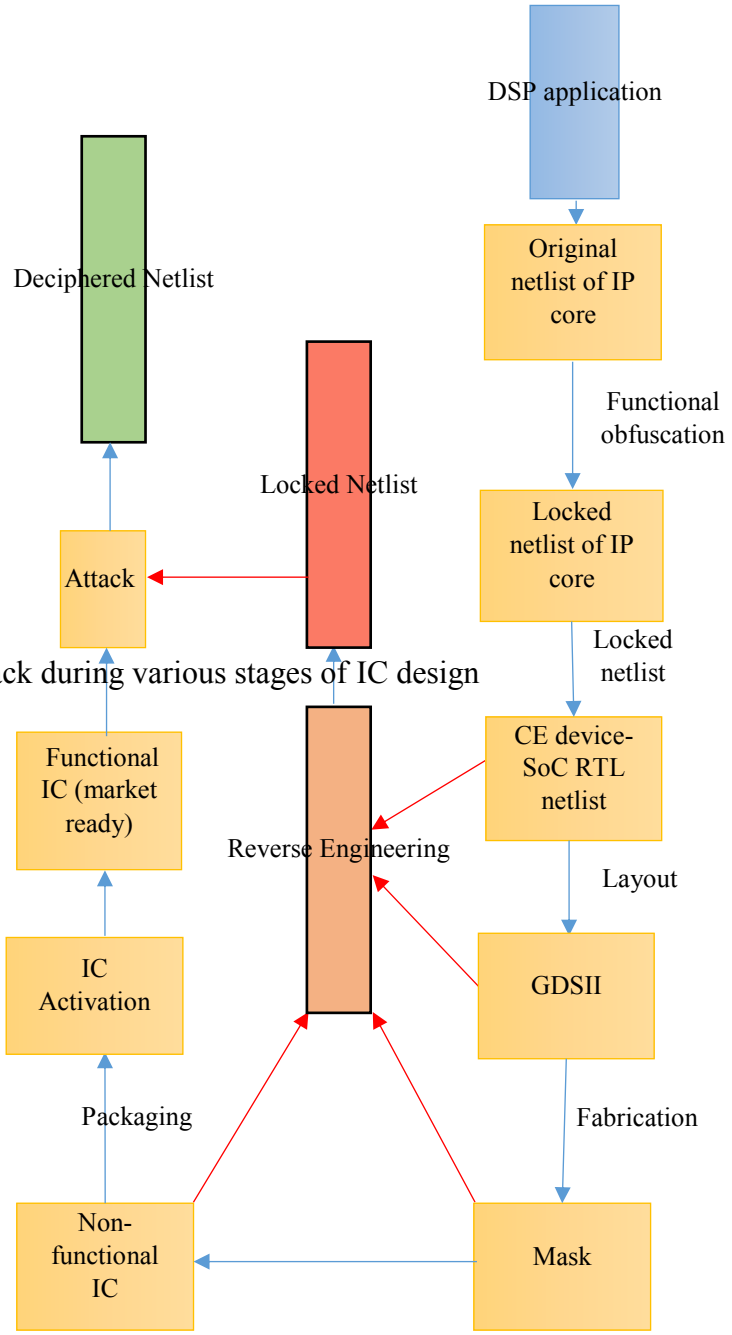


Fig. 6.1 Possibility of Reverse engineering attack during various stages of IC design

locked DSP IP core. This is because DSP circuits have several alternative design solutions and selection of an optimal (or low-cost) alternative requires integration of a design space exploration framework such as PSO-DSE. In case, if an optimization framework such as PSO-DSE is not incorporated while designing DSP IP cores, then the generated design may incur huge area, power, and delay overheads.

The proposed approach presents a novel methodology for generating a '*low cost highly secure, functionally obfuscated DSP IP core through robust locking'*

6.2. Threat model

Fig. 6.1 shows the typical IC design flow. The IP core designer will take DSP application as input and perform functional obfuscation (functional locking) to generate locked netlist of IP core. These IP cores will be integrated in SoC designs and a layout of SoC is created in the form of GDS-II file which is further processed as shown in fig.6.2. A malicious attacker could perform reverse engineering on layout, mask, non-functional IC to obtain the locked netlist. Further, he could perform attack such as key sensitization attack to obtain the unlocked (deciphered netlist). The primary motive of an attacker is to determine the secret key, so that he/she could unlock the circuit, manufacture the IC and sell them illegally. Additionally, an attacker can understand the design if correct key-bits are known and hence could insert hard to detect trojans at *safe places* [21, 22]. To accomplish these attacks, an attacker is assumed to possess the following:

(a) Locked netlist: obtained through theft or reverse engineering of layout or mask.
(b) A functional IC: brought from open market.

6.3. Proposed approach

This section briefly describes major components of proposed methodology.

6.3.1. Problem formulation

Given a DSP application in the form of data flow graph (DFG) or control data flow graph (CDFG) along with module library, IP core locking blocks (ILBs),

PSO control parameters as inputs, generate a low-cost highly secure functionally obfuscated DSP IP core.

6.3.2. Overview of proposed methodology

As shown in fig 6.2, the proposed methodology comprises of two major components namely PSO-DSE and IP functional locking. The first step of the proposed approach is to initialize the particle swarm [32]. For each particle position, a gate level datapath structure is created. Subsequently, proposed IP locking blocks are inserted in the gate level structure. Further, fitness and security (strength of obfuscation) of the obfuscated design for each particle's position is evaluated. Based on the particle's fitness PSO-DSE parameters are updated. This process is repeated till one of the PSO-DSE termination criteria is met. The solution thus generated is low-cost functionally obfuscated DSP IP core. The functionally obfuscated design thus obtained will be highly robust against reverse engineering based attacks. The particle swarm optimization-based design space exploration (PSO-DSE) block is primarily responsible for exploring low-cost design solution.

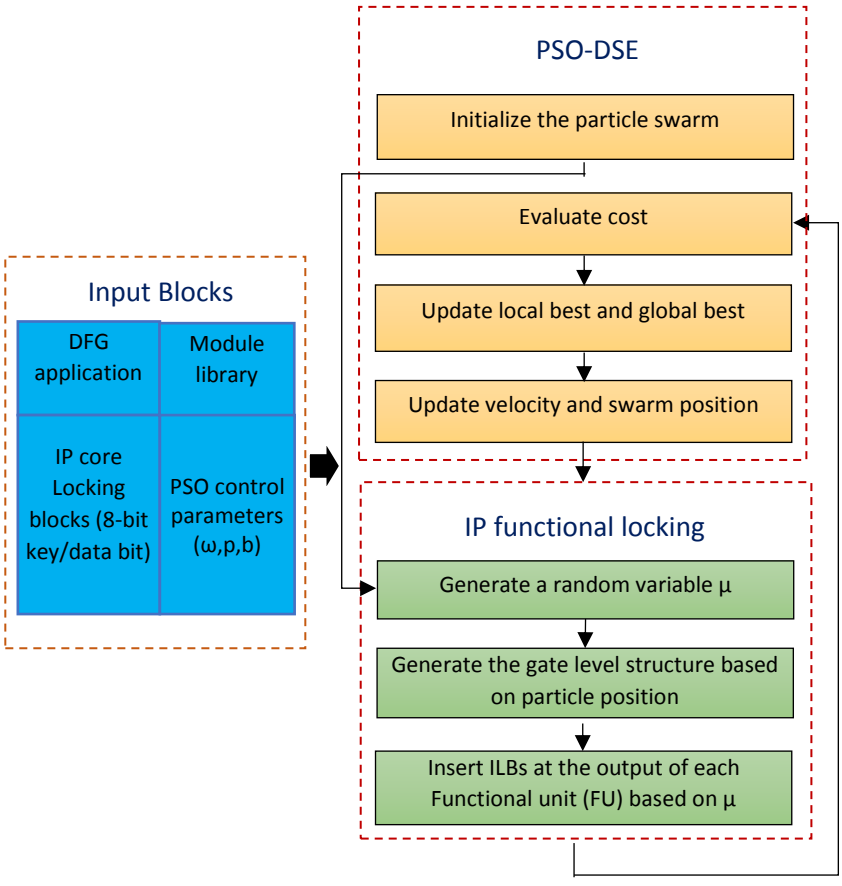


Fig. 6.2. Details of proposed functional obfuscation methodology

The upcoming section describes our proposed IP locking blocks and discuss their properties responsible for enhancing strength of obfuscation.

6.3.3. Proposed IP core locking blocks

This section discusses properties of proposed ILBs shown in fig. 6.3. Each ILB provides same robustness against RE and key sensitization attacks. However, they activate for different key bits. Further, Each ILB has different structure that causes different implications on hardware power and delay. These implications are considered and incorporated in PSO-DSE framework with the help of modified particle encoding. The modified design space represents particle positions as Xi = {NR_1, NR_2, …, NR_D, μ} for DFG applications and Xi = {NR_1, NR_2, …, NR_D, UF, μ} for CDFG applications. Where μ is a random integer. The proposed methodology is applicable to both DFG as well as CDFG applications. However, to avoid confusion, the proposed approach will be presented in context of DFG applications.

The proposed ILBs incorporate robust security features such as multi-pairwise security, prohibition of key gate isolation etc. These security features enhance robustness against reverse engineering and key sensitization attacks as discussed below:

- *Multi-pairwise security*: This security feature is responsible for providing protection against key sensitization attack. Key sensitization is an attempt of an attacker to identify and apply input pattern combination that sensitizes key-bits to primary output pins [21, 22]. The attacker can identify single input pattern or a combination of input patterns for sensitizing key-bits and apply them to observe correct key bits at the output pins of functional IC. Key-bits K1 & K2 are said to be pairwise secure if an attacker cannot sensitize K1 without knowing/controlling key bit K2 and vice-versa [21]. Our proposed ILBs are multi-pairwise secured, i.e., any of the 8 key-bits cannot be sensitized without knowing/controlling other 7 key-bits. Therefore, an attacker must apply brute-force attack to determine the correct key. Thereby proposed ILB's multi-pairwise security property enhances robustness of functional obfuscation methodology and increases complexity of reverse engineering in comparison to other locking units present in literature.

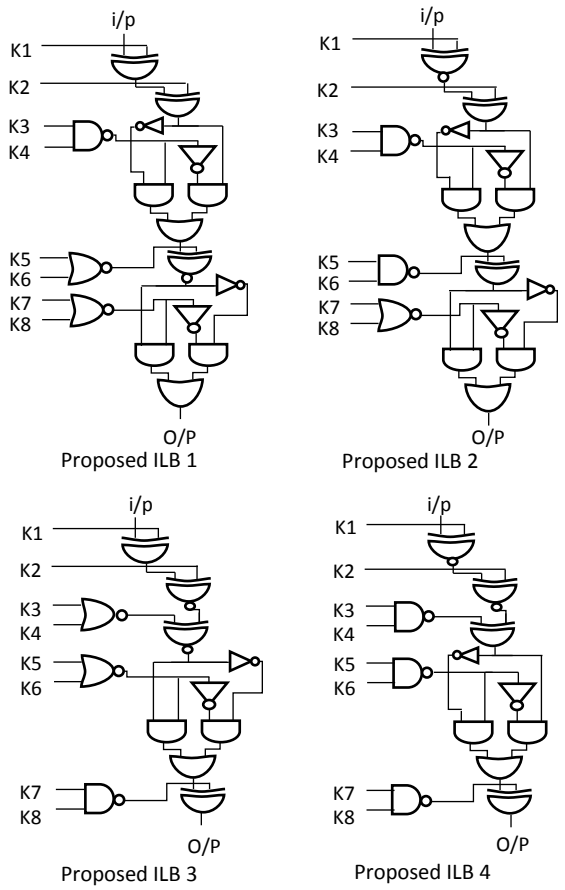


Fig. 6.3 Proposed IP core locking Blocks

- *Prohibiting key gate isolation*: Isolated key gates can be easily sensitized using key sensitization attacks as shown in [21]. A key K_{iso} is said to isolated if there is no path between K_{iso} and remaining keys of the locked design and vice-versa. Hence, such keys are highly vulnerable to sensitization attacks and therefore must be avoided. The proposed ILBs have multiple path between key bits and none of the key-bits are isolated hence proposed ILBs have higher resiliency against key sensitization attack.
- *Protection against run of key gates*: A few combinations of run-of key gates may reduce the effort of an attacker to identify correct key by increasing number of valid keys [21]. Further, an attacker could replace run of key gates with a single gate. This is not feasible in case of proposed ILBs as key gates of proposed ILBs are intertwined for 8-key bits. Hence, it is very difficult to identify run of key gates in the structure of proposed ILBs.

- *Non-mutable key gates*: An attacker try to identify a *'non-key'* primary input between the path connecting two key bits K1 and K2 such that by controlling this input, effect of K1 can be stopped from reaching K2 and simultaneously K2 can be sensitized to primary output. Such a key gate K1 is termed as mutable key gate. The proposed ILBs have intertwined paths between its 8 key-bits. Hence, it is infeasible to sensitize a particular key bit without controlling remaining 7 key bits. Further, the effect of 7 key bits cannot be muted by controlling a single input. Thus, proposed ILBs are robust against muting based key sensitization attacks presented in [21].

6.3.4. Insertion technique of proposed ILBs

As discussed earlier, the particle positions are encoded as Xi = {NR1, NR2, …, NRD, μ} where μ is a random number between 1 and T_{ILB}; where, μ symbolizes user specified repetition pattern of ILB insertion. T_{ILB} is the total number of different ILB structures available for selection. Once a gate level structure is generated with respect to each particle position, the proposed ILBs are inserted at the output of each functional unit (FU), each data output bit is locked using an ILB. The same ILB is inserted 'μ' times. After 'μ' repetitions new ILB is selected from T_{ILB} and inserted 'μ' times. The process is repeated till all the output bits of FUs are locked using proposed IP functional locking blocks (ILBs).

An illustrative example of 4-bit FIR locked datapath generated for particle position {1A, 1M, μ=2} is shown in fig.6.3. Initially, a gate level structure of FIR benchmark is generated based on resource configuration (1Adder, 1Multiplier). Subsequently, as μ=2, the proposed ILB1 is inserted at first two output bits of adder functional unit. Further, after 'μ=2' repetitions, ILB2 is selected and inserted at next two output data bits. The process is repeated till all the output bits of each FU is locked.

6.3.5. Security due to insertion of proposed ILBs

The security enhancement due to insertion of proposed ILBs is given by following equation

$$K_S = 2 \text{ ^ } (b * m * f) \qquad \textbf{(6.1)}$$

Where K_S symbolizes the key-space (Strength of Obfuscation), b = key-bits per ILB, m = number of ILBs inserted per functional unit, f = number of functional unit in the datapath. For example, consider the security evaluation of 4-bit FIR benchmark shown in fig.6.3. The number of output bits of each FU is 4. Therefore, number of ILBs inserted per functional unit is (m=) 4. Further, as each ILB structure has 8 key-bit therefore b=8. Additionally, as the FIR datapath is generated for resource configuration (1 adder, 1 multiplier). Hence, number of functional units in the datapath is (*f=)* 2. Therefore, the strength of obfuscation of 4-bit FIR datapath is $K_S = 2 \wedge (8*4*2) = 1.8\ e+19$.

The upcoming section analyze security of proposed methodology from an attacker's perspective.

6.3.6. Security analysis of proposed methodology

An attacker is assumed to have following tools/facilities to unlock the locked design:

- Access to an advanced fabrication facility.
- A locked gate-level netlist obtained through theft or reverse engineering the layout or mask of the locked design.
- Functional IC bought from open market.

An attacker who has access to these tools will try to determine the number of key bits through reverse engineering. Once an attacker determines the correct set of key-bits. He/she will try to apply key sensitization attack to determine the value of key-bit that matches with a valid key. As the proposed methodology is resilient to several state-of-art attacks (see section 6.3.3 and 6.3.7). Hence, an attacker is forced to apply brute force attack to identify the valid key. For demonstrative example of FIR datapath having 64 key-bits, an attacker has to apply 2 ^ 64 different combination of key-bits to determine the correct key. Hence, if 1 billion combinations of key-bits could be applied in 1 second [21], it would require 10^21 years to determine the valid key using brute force attack.

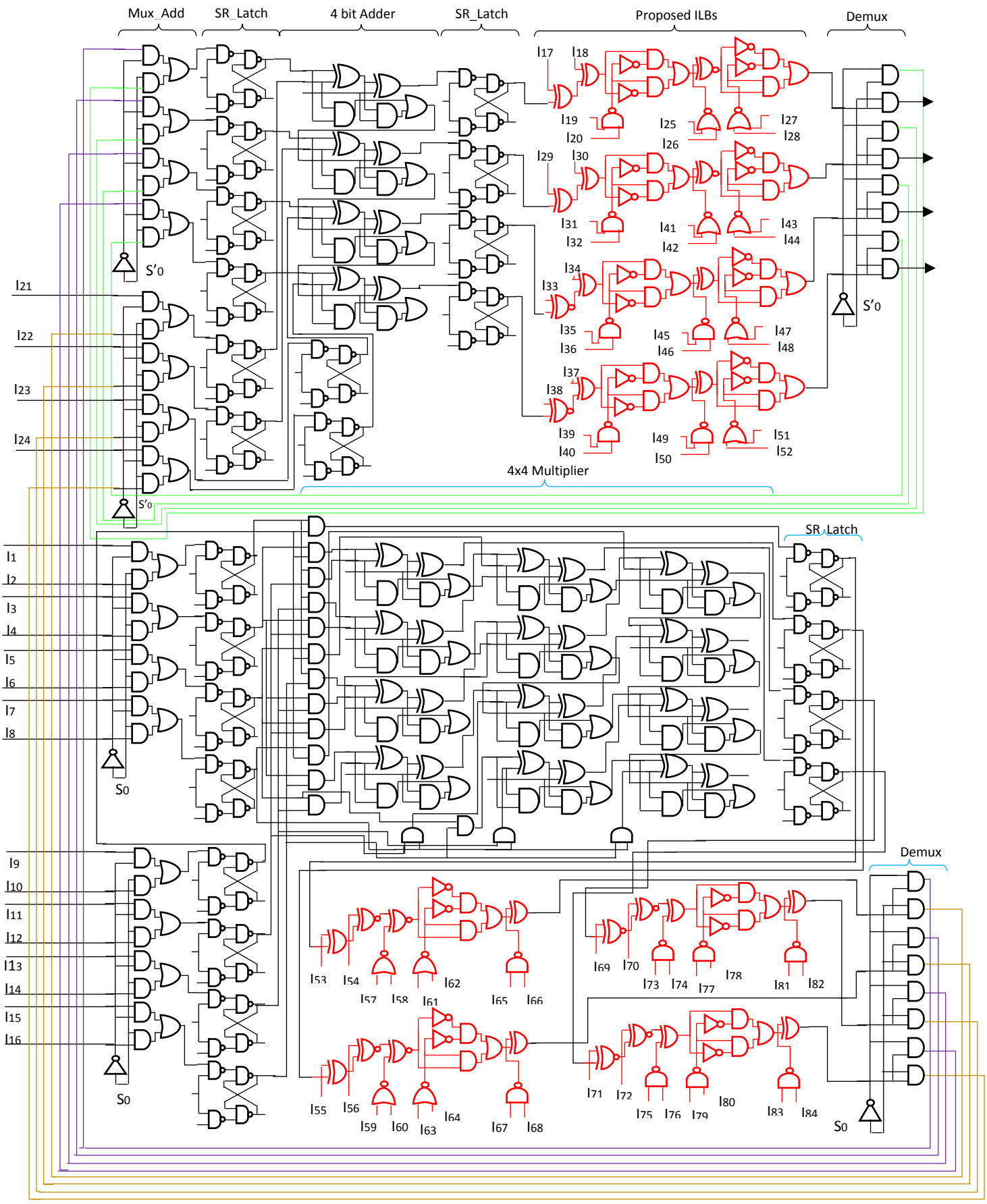


Fig. 6.4 Obfuscated (locked) gate-level 4-bit FIR for (1A, 1M, μ=2) locked with 64-bit key

6.3.7. Resiliency of proposed methodology against various state-of-art attacks

This section discuss the resiliency of proposed approach against key-sensitization attack [21, 22], IP piracy attack [36, 37], and Trojan insertion attack [38].

(i) *Key sensitization attack based on isolated key-bits*: A isolated key bit can be easily sensitized. Hence, to avoid its sensitization isolation must be avoided. A key-bit k^{iso} is said to be isolated if there is no path between k^{iso} and any of the remaining key-bits utilized for locking the circuit. As discussed earlier, our proposed ILB are intertwined structure of 8 key-bits interdependent on each other. Hence, key sensitization due to isolated key-bits is not feasible in our proposed ILB structures.

(ii) *Key sensitization attack based on run of key-gates*: A back-to back connection of key gates is termed as run of key gates [21]. The run of key gates can increase the valid (correct) key in the key space. Thereby, reducing the effort to identify a valid key through brute force attack. In run-of-key based attack, an attacker tries to identify and replace a run of key gates with a single key gate and identify the input value of replaced key gate. Based on this value, the correct key bits are determined. The proposed ILBs are intertwined connection of gates among 8 key inputs. Hence, complexity to identify and replace run of key gates is increased compared to XOR/XNOR based run of key gates.

(iii) *Key sensitization attack based on mutable key-gates*: An attacker attempts to mute the impact of a key bit ($k^{mutable}$) from reaching another key-bit ($k^{sensitizable}$), such that while $k^{mutable}$ is muted, the key-bit $k^{sensitizable}$ could be sensitized to the primary output. The muting is performed by controlling the path between two key bits by controlling few primary inputs. Such an attack is not feasible through our proposed ILB structures as the proposed ILBs doesn't have any such controllable (by primary inputs) path between its 8 key bits. Furthermore, proposed ILB's multi-pairwise security feature ensures a key bit cannot be

sensitized without controlling the remaining 7 key-bits. Hence, proposed ILBs are resilient to mutable key-gates based sensitization attacks.

(iv) *IP piracy and trojan insertion attacks*: An attacker or a pirate must understand the correct functionality of the IP core, so that a pirate can identify the appropriate buyer for re-selling the IP core and market (explain) the IP properly. Further, an attacker targeting trojan insertion must understand the correct functionality so that the trojan(s) could be inserted at *safe places.* Thereby reducing chances of detection. The proposed functional obfuscation methodology based on ILBs, enhances the effort of an attacker to identify the correct key as it is resilient to many state-of-art attacks discussed above.

6.4. Proposed PSO-DSE framework for generating low-cost functionally obfuscated DSP IP core.

This section provides a detailed description of PSO-DSE framework. The PSO-DSE framework comprises of four major steps as follows:

6.4.1 Particle encoding and swarm initialization

The particles of the swarm (Pi) are encoded as Xi = {NR_1, NR_2, …, NR_D, μ}, where Xi denotes position of i^{th} particle in the design space, NR_D represents the number of resources of type R_D in the D^{th} dimension of the design space, μ is a random integer between 1 and T_{ILB} ($1 \leq \mu \leq T_{ILB}$). Each particle represents number of hardware resources (along with μ) utilized for generating functionally obfuscated IP cores. Subsequently, particles swarm is initialized. The first three particles (P1, P2 and P3) are initialized at positions:

X_1={min(R_1), min(R_2), … , min(R_D), μ}

X_2={max(R_1), max(R_2), … , max(R_D), μ}

X_3={[min(R_1) + max(R_1)]/2, … , [min(R_D) + max(R_D)]/2, μ}

Representing minimum, maximum, and middle positions of the design space [32, 33]. Hence, ensuring good coverage of design space. Subsequently, the remaining particles (Pi) are initialized as:

X_i={[min(R_1) + max(R_1)]/2 ± α, … , [min(R_D) + max(R_D)]/2 ± α,μ}

Where, min(R_D) and max(R_D) denotes minimum and maximum resource in D^{th} dimension respectively. α is a random integer between minimum and maximum value of D^{th} dimensional resource.

6.4.2 Fitness / cost evaluation

For each particle's position, a gate level structure is created based on the number of hardware resources in D^{th} dimension. Subsequently, ILBs are inserted based on μ. The fitness of the obfuscated IP core thus generated is evaluated using following cost/fitness function.

$$C_f(X_i) = \phi_1 \frac{P^{OB}}{P_{max}{}^{OB}} + \phi_2 \frac{D^{OB}}{D_{max}{}^{OB}} \quad \textbf{(6.2)}$$

where $C_f(X_i)$ represents the cost/fitness of the obfuscated IP core, based on the (resource configuration) particle position X_i. Φ1 and Φ2 are weightage of power and delay of obfuscated IP core respectively. P^{OB} and D^{OB} are the power and delay of the IP core based on particle position Xi. $P_{max}{}^{OB}$ and $D_{max}{}^{OB}$ are the maximum power and maximum delay of the functionally obfuscated IP core's design space.

6.4.3 Updating local best and global best

The local best and global best are updated as explained in PSO-DSE framework of previous chapters as well as in [32, 33].

6.4.4 Updating Velocity and particle's position

The velocity and particle's position are updated as explained in PSO-DSE framework of previous chapter. The PSO-DSE process generates low-cost, highly secure, functionally obfuscated design solution upon termination.

6.5. Summary

The proposed approach presents a novel methodology for generating a low-cost highly secured functionally obfuscated DSP IP core. Further, the proposed methodology introduces a novel locking unit termed as IP locking block (ILB). This chapter presented the security enhancing properties of the ILB. Subsequently, the security of the proposed approach is evaluated and demonstrated with the help of an example of FIR benchmark.

Chapter 7

Methodology for analyzing the aging effect of NBTI stress on performance of DSP IP core

This chapter provides a detailed description of the proposed approach to analyze the impact of negative bias temperature instability (NBTI) stress on performance of DSP IP core. The given methodology can be utilized to detect presence of accelerated aging attack on an IP core. In the first section we will introduce the problem. In the second section we will present a brief overview of the proposed solution. The third section will describe the major blocks of the proposed solution with the help of a demonstrative example. The fourth will conclude the chapter.

7.1. Introduction

As discussed in previous chapters, technology scaling has raised several reliability and security concerns. One such reliability concern is negative bias temperature instability [39-42]. NBTI occurs when a negative bias is applied between gate and source terminal of a PMOS transistor at an elevated temperature resulting in instability of transistor's parameters such as threshold voltage (Vth), transconductance(gm), etc. The continuous application of NBTI stress causes degradation in delay (performance) of the transistor. A malicious attacker may exploit this phenomenon to accelerate the aging process of a PMOS transistor due to NBTI stress [15]. Different input vector activates (stresses) different PMOS transistors in a circuit thereby degrading performance of different transistors [43, 44]. An attacker would like to determine the input vector causing maximum degradation of critical path of a circuit thereby causing maximum acceleration in performance degradation (aging) of the device. On the other hand, a designer would like to determine these input vectors and apply input vectors causing minimum performance degradation during the standby mode. The proposed approach presents a novel methodology for (a) estimating performance degradation of DSP IP cores subjected to NBTI stress (b) determine input vectors that causes minimum/maximum degradation. (c) presents hardware-based attack model for accelerated aging attack on DSP IP cores.

A large share of electronic products manufacturing companies focuses primarily on consumer electronic (CE) devices such as television, cameras, mobile phones etc. Majority of these electronic devices contains at least one digital signal processing (DSP) component. Further, due to arduous competition and stringent time-to-market deadlines, CE industry rely heavily on 3rd party IP core to beat the competition. This dependency of CE industry on 3rd party IP cores can be exploited by a malicious attacker in the IP design house or IP supply chain to perform several types of attacks such as trojan insertion, IP piracy, etc. One such attack is *accelerated aging attack* using NBTI stress [15]. In this type of attack, an attacker aims to modify the IP core such that the IP core remains under constant NBTI stress in the standby mode. The aim of the attacker is to ensure continuous performance degradation of the IP core (thereby of the device that integrates the compromised IP core), even when the device is not in active usage. The primary motive of the attacker is to cause device failure within warranty period [15]. Different input vectors cause different amount of NBTI stress on the circuit [43, 44]. Therefore, techniques are required to identify the impact of input vectors on DSP IP core.

The proposed approach presents a novel methodology for *'performing NBTI stress analysis of DSP IP core that can be utilized to identify the presence of accelerated aging attack on DSP IP cores'*

7.2. Proposed approach

This section provides a brief overview of our proposed methodology.

7.2.1. Problem formulation

Given a DSP application in the form of data flow graph (DFG) or control data flow graph (CDFG) along with module library, perform the NBTI stress analysis to determine the input vectors that causes maximum degradation due to continuous NBTI stress.

7.2.2. Overview of proposed methodology

The proposed work presents a novel methodology for analyzing the effect of NBTI stress on DSP IP cores. Based on the analysis the input vectors causing maximum degradation are determined and are utilized to identify the presence

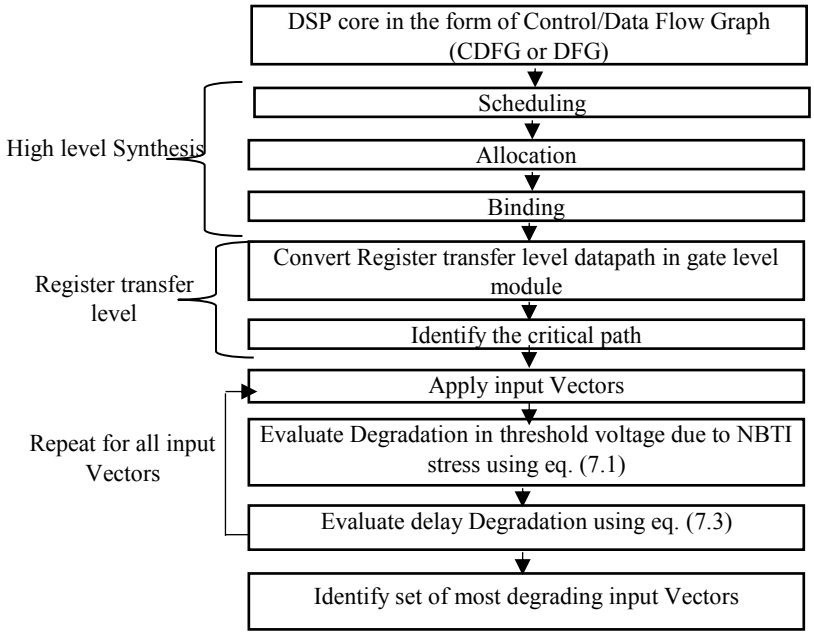


Fig. 7.1 Proposed NBTI stress analysis methodology

of accelerated aging attack on the DSP IP core. As shown in fig.7.1, The first step of the proposed approach takes DSP application in the form of DFG or CDFG as input and performs high level synthesis (scheduling, allocation and binding) to generate a register transfer level (RTL) datapath. The RTL datapath thus obtained is converted into gate level structure. Subsequently, the critical path of the gate level structure is determined. Later, input vectors are applied on the gate level structure and degradation in performance parameter (threshold voltage) is evaluated. Subsequently, the degraded threshold voltage is utilized to calculate delay degradation. The process is repeated for all input vectors and the input vector(s) causing maximum degradation are identified. Further, the presence of accelerated aging attack in the device is identified by operating the device in the standby mode for a substantial amount of time (say 15 days). If the device's performance degrades with similar rate as that of *maximum rate of degradation, then accelerated aging attack is said to be present in the device.*

The approach to evaluate the effect of NBTI stress on the DSP IP core is discussed in the upcoming section.

7.2.3. Evaluating effect of NBTI stress on DSP IP core

The various combinations of input vector are applied on the gate level structure of DSP IP core and the impact of NBTI stress on PMOS transistor's

parameters such as threshold voltage and delay are evaluated using equations 7.1, 7.2 and 7.3.

$$\Delta V_{\text{th}} = b \cdot a^n t^n \quad (7.1)$$

Where, ΔV_{th} represents change in threshold voltage due to NBTI stress. b = 3.9 x 10^{-3} V.s-1\6, n is time exponential constant=0.16, a = input signal probability, t = time in seconds.

$$V_{th}^{new} = V_{\text{th}} + \Delta V_{\text{th}} \quad (7.2)$$

Where, V_{th}^{new} represents new threshold voltage after PMOS transistor is stressed for 't' amount of time. V_{th} represents threshold voltage= 0.365V for 65nm technology scale [15]. Further, the new threshold voltage(V_{th}^{new}) of pmos thus obtained is utilized in eq. 7.3

$$T = K \frac{V}{(V - V_{th}^{new})^{\alpha}} \quad (7.3)$$

Where, T= delay of pmos transistor, K is technology based proportionality constant, V = V_{DD}. For 65nm technology scale, V= 1.2V is adopted from [15], and α=1.4, K=155 x 10^{-6} is adopted from [45].

Equation 7.1 represents change in threshold voltage, when a continuous NBTI stress is applied for a duration of 't' seconds. The change in threshold voltage is added to original threshold voltage to obtain new threshold voltage using eq.7.2. Subsequently, the new threshold voltage is utilized to evaluate degraded delay of stressed PMOS transistor using eq. 7.3. Note that the delay of NMOS transistor is evaluated using original threshold voltage instead of new threshold voltage because NBTI stress does not affect NMOS transistors. A case-study of the proposed methodology on FIR benchmark is presented in the upcoming sub-section.

7.2.4. Case-study

The FIR application can be represented as a pseudocode shown in fig.7.2(a). In the initial step of proposed approach, application's pseudocode is converted into data flow graph (DFG) and taken as input. Subsequently, high level synthesis is performed to obtain register level datapath [46]. HLS comprises of three sub-steps: Scheduling, allocation and binding. In the first sub-step, the

```
For (i = 0 ; i< 2 ; i++)
{
Y(n) + = x(n-i) * h_i
```

Fig 7.2(a) Pseudocode of FIR benchmark

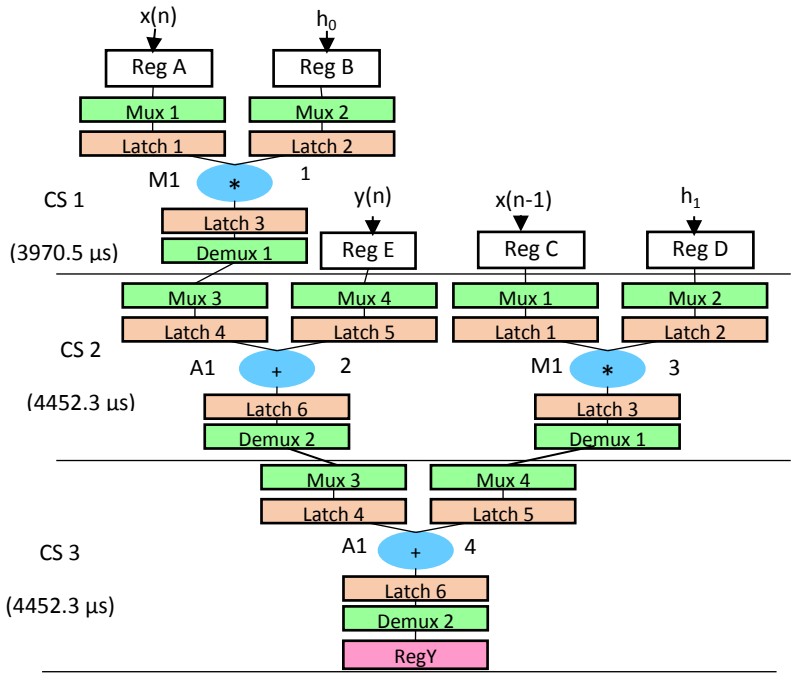


Fig.7.2(b) Scheduling and allocation diagram based on sample resource configuration (1A, 1M)

scheduling of FIR benchmark is performed based on resource configuration (1A, 1M). Subsequently, resources are allocated to each operation during allocation step of HLS. The scheduled and allocated FIR application is shown in fig.7.2(b). Subsequently, all the operations allocated to a particular hardware resources (say adder1 (A1)) are bonded together during binding step of HLS. The RTL datapath thus generated is subsequently converted into subsequent gate level modules (of NAND gates) and critical path is identified as shown by red colored line in fig.7.3. The critical path comprises of 11 gates (G1, …, G11) in the critical path of multiplier and 12 gates (G12, …, G23) in the critical path of adder sub-circuits. Subsequently, various combinations of input vector are applied to primary input of FIR datapath and correspondingly turned on PMOS/NMOS transistors of each gate of the critical path is tabulated. Table 7.1 shows the turned on PMOS/NMOS transistors on applying input vector 11101. The NBTI stress occurs on PMOS transistor of CMOS NAND gates when logic'0' is applied at its input. The *degraded* delay of stressed PMOS transistors is evaluated using equations 7.1, 7.2 and 7.3.

Table 7.1. Gate delay and pmos details corresponding to stress time 1 year for input test vector 11101 (Note : G1, …., G23 represents gates of FIR datapath)

G1	G2	G3	G4	G5	G6	G7	G8
2PMOS	2NMOS	1P 1N	2NMOS	1P 1N	2NMOS	2PMOS	2NMOS
261.8 µs	478.8 µs	266.7 µs	478.8 µs	266.7 µs	478.8 µs	261.8 µs	478.8 µs
G9	G10	G11	Total				
1P 1N	2NMOS	2PMOS	----				
261.8 µs	478.8 µs	261.8 µs	3970.5 µs				

CS 1

G12	G13	G14	G15	G16	G17	G18	G19
2PMOS	2NMOS	1P 1N	2NMOS	1P 1N	2NMOS	1P 1N	2NMOS
261.8 µs	478.8 µs	261.8 µs	478.8 µs	261.8 µs	478.8 µs	266.7 µs	478.8 µs
G20	G21	G22	G23	Total			
1P 1N	2NMOS	1P 1N	2NMOS	----			
266.7 µs	478.8 µs	266.7 µs	478.8 µs	4452.3 µs			

CS 2

G12	G13	G14	G15	G16	G17	G18	G19
2NMOS	2PMOS	2NMOS	1P 1N	2NMOS	2PMOS	1P 1N	2NMOS
478.8 µs	261.8 µs	478.8 µs	261.8 µs	478.8 µs	261.8 µs	266.7 µs	478.8 µs
G20	G21	G22	G23	Total			
1P 1N	2NMOS	2PMOS	2NMOS	Total			
266.7 µs	478.8 µs	266.7 µs	478.8 µs	4452.3 µs			

CS 3

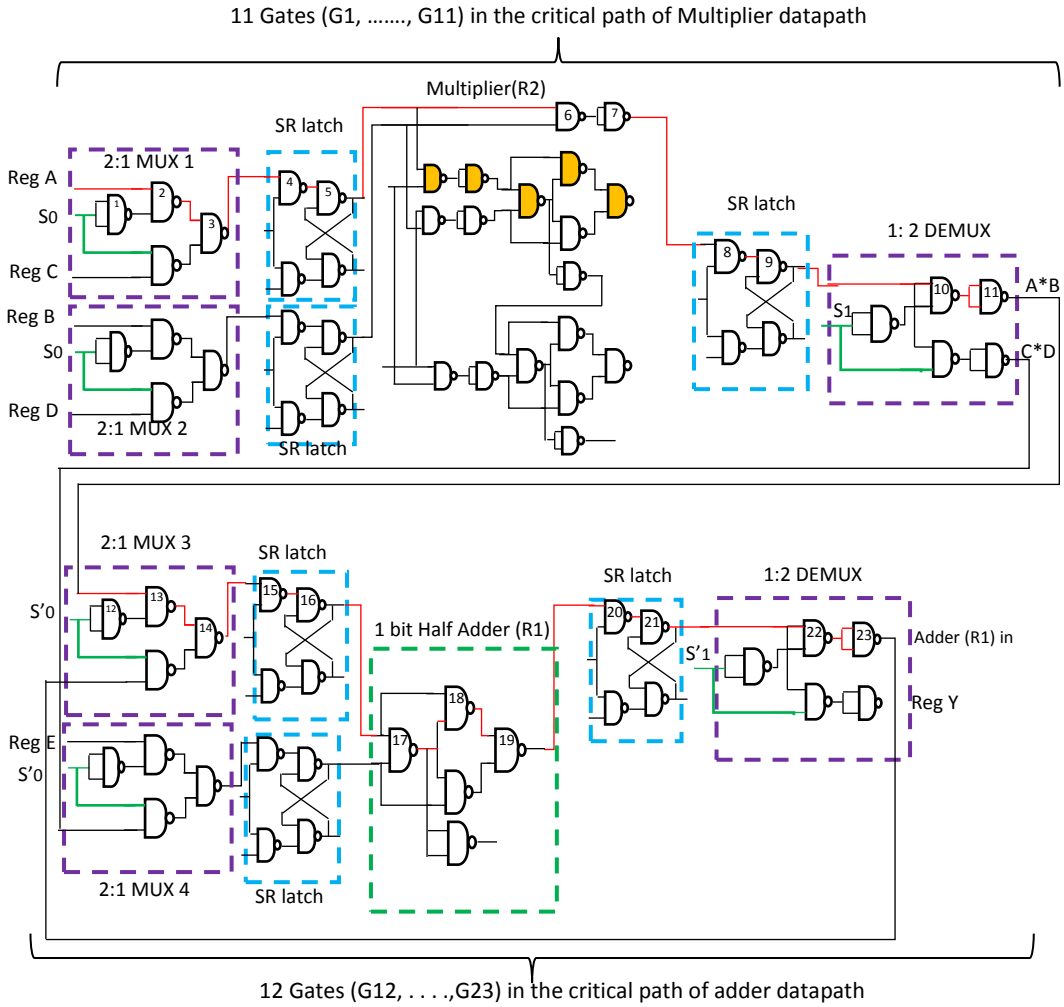


Fig. 7.3 NAND based gate level implementation of FIR datapath

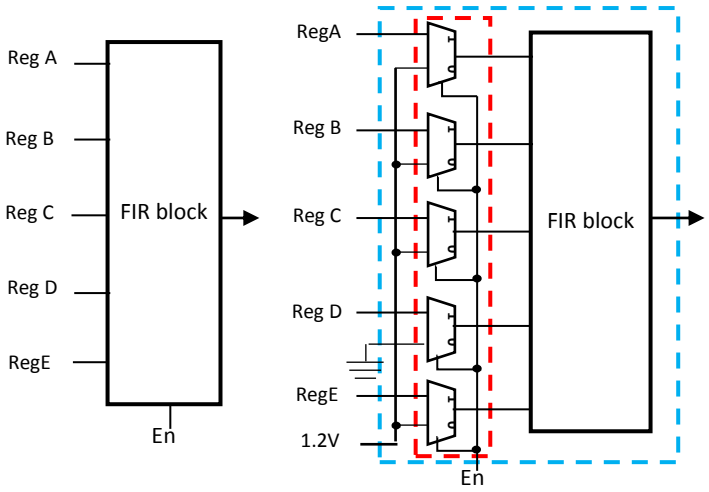


Fig.7.4(a) FIR IP core block Fig.7.4(b) Modified Hardware logic

The process is repeated for each possible combination of input vectors. Finally, the input pattern causing maximum degradation is identified. Based on the identified vector, an attacker could launch an accelerated aging attack as discussed below.

7.3. Accelerated aging attack: Modelling and detection

This section presents attack model and detection mechanism of accelerated aging attack

7.3.1 Attack model

An attacker would exploit the natural aging of PMOS transistor due to NBTI stress to accelerate the aging process. To achieve acceleration, an attacker must keep PMOS transistor in stressed (turned on) state for as long as possible. To accomplish this goal, an attacker must devise an attack that apply continuous stress when the device is in standby mode (i.e., outside natural aging due to active usage). The attack could be launched through hardware as well as software modifications as discussed below

- Hardware based attack model: As shown in fig.7.4(b), The attacker can devise a hardware modification such that the modified DSP IP core age naturally (functions correctly) when enable signal ‘EN’ is ‘1’. Moreover, aging is accelerated when ‘EN’ is ‘0’ (in standby mode) by applying most harmful vector 11101.

- Software based attack model [15]: An attacker could also identify the correct working of DSP IP core by reverse engineering the device. Subsequently, a software modification is devised such that the hardware is in continuous stress in operating system mode.

7.3.2 Detection of accelerated aging attack

As discussed in previous section, an aging attack could be modelled as hardware or software based attack. However, the detection method of both type of attack is same. A tester should keep the device activated in the standby mode or operating system mode for a substantial amount of time (say fifteen days). After 15 days the tester can test the delay of the device, if the degradation of IP core occurs roughly at the same rate as the *maximum rate* (degradation due to input vector causing maximum degradation) then presence of accelerated aging attack is confirmed. Hence, if an attack is detected, the design house should check for and remove any malicious hardware or software modifications.

7.4. Summary

The proposed approach presents a novel methodology to analyze the impact of aging due NBTI stress on DSP IP cores. The impact of NBTI stress is analyzed based on the following: (a) performance degradation of DSP IP cores subjected to NBTI stress (b) input vectors that causes minimum/maximum degradation. The proposed approach presents hardware-based attack model for accelerated aging attack on DSP IP cores.

Chapter 8

Computational forensic engineering methodology for resolving ownership conflict of DSP IP core generated using high level synthesis

This chapter provide a detailed description of the proposed approach to resolve false claim of ownership of reusable DSP IP core using computational forensic engineering (CFE). The first section introduces the problem. The second section presents a brief overview of the proposed solution. The third and fourth section describes the proposed methodology with the help of demonstrative examples. The fifth section concludes the chapter.

8.1. Introduction

As discussed in previous chapters, consumer electronic industries rely heavily on 3rd party IP (3PIP) core to beat the competition. This is because 3PIP cores helps in achieving higher productivity and reducing deign development time. However, 3PIP core are vulnerable against several threats such as abuse of IP ownership, IP piracy, false claim of ownership, etc. [36-38, 47-48] Hence, protection mechanisms are required to provide protection against these threats. The proposed methodology provides protection against one such threat known as false claim of ownership.

Although mechanisms such as patents, copyright, trademarks, etc. are provided by law to enjoy the legal ownership. However, these mechanisms are either incapable or inadequate in protecting reusable IP cores [13]. Further, in context of reusable IP cores, IP piracy is a major threat. A malicious attacker can obtain the IP by means of theft/fraud. By virtue of which he/she can also claim to be the rightful owner of the IP. In such a scenario, methodologies to resolve ownership conflict of reusable IP core is needed. One such approach is digital watermarking [13, 49]. In this approach, signature is inserted in the design without affecting the functionality of the design by the IP designer. Further, if someone else falsely claims the ownership of the IP, signature detection step is carried out to identify the rightful owner. Because signature will be known only to the rightful owner (although rarely, but an attacker can

recover signature through reverse engineering), if signature is detected in the IP core, ownership will be awarded to the rightful claimant. However, watermarking requires signature insertion while designing an IP core. In case if designer doesn't forecast the possibility of the threat or does not take appropriate measures such as signature insertion (watermarking) during the design phase. Then, ownership claims will become very hard to resolve. Moreover, watermarking is vulnerable to signature tampering attacks. Hence, methodologies are required that can resolve the ownership without depending on proactive measures such as signature insertion. In this chapter, we will present a novel methodology that does not depend on such proactive measures. Further, there is no known attack on the generic CFE, which is the baseline framework used in our proposed approach.

The proposed approach presents a novel computational forensic engineering based methodology to *'protect reusable DSP IP cores generated using high level synthesis against false/fraudulent claim of ownership'*

8.2. Computational Forensic Engineering Framework

This section provides a brief description of generic CFE framework utilized in our proposed methodology.

8.2.1. Generic CFE: Problem definition

A typical CFE problem can be formulated as: given a solution 'S' to a problem 'P' having a finite set of algorithms/tools AT_n (n is a non-zero positive integer) applicable to problem P, that can generate solution S, identify with a certain degree of confidence that the algorithm/tool AT_i has been applied to generate the solution S [50, 51].

8.2.2. Overview of generic CFE

A generic CFE approach comprises of four stages: (a) feature and data collection (b) feature extraction (c) Algorithm clustering, and (d) Validation [50]. During the execution of the first stage, features are identified that can classify the data point in one of the categories during multi-category classification. Further, features are extracted from each solution of the various algorithms, during feature extraction stage of CFE. Once the features are

extracted, the data points (algorithms/tools) are classified (clustered) in several categories during algorithm clustering stage of CFE. Finally, during the validation phase, the accuracy of the classification is checked for. If the classification is sufficiently accurate (say ≥ 95% accuracy), then the CFE approach is said to be able to classify any other data point with the same accuracy. If the classification is not sufficiently accurate, then new features should be introduced for increasing accuracy.

8.2.3. Comparison of proposed CFE vs generic CFE

In our proposed methodology, we have adopted only three stages of generic CFE as (a) IP core feature and data collection, (b) IP core feature extraction, and (b) IP validation. Note that in our proposed approach we have not adopted algorithm clustering stage as our problem is loosely related to clustering. The proposed approach classifies the claimant in just two categories: 'Rightful owner' and 'fraudulent claimants'. In practical scenarios, the number of IP vendors claiming to be the rightful owner of an IP core will be very few (typically 2 to 3) with only one rightful claimant. Hence, the ownership problem has very few data points and thus will create two clusters (classes) of size 1 and 'n-1' (typically 1 to 2) respectively. Therefore, our proposed approach does not require a separate clustering stage. Moreover, while resolving ownership conflicts, the resolution must be 100% accurate. Hence, our adoption of IP validation stage necessitates 100% accuracy. Therefore, our proposed methodology skips the optional algorithm clustering and identify the 'rightful claimant' in the IP validation stage.

8.3. Proposed approach

This section describes the proposed methodology for resolving ownership of reusable IP cores generated using HLS.

8.3.1 Key points about the proposed approach

- The proposed CFE approach for IP ownership is applicable in scenarios where 'n' IP vendors are claiming to be the rightful owner of an IP core. Each IP vendor is assumed to have its own HLS tool to generate their respective IP designs. However, if two or more IP vendors uses a common third-party HLS tool then proposed approach is not applicable.

- The proposed approach does not require source code, packaging information of HLS tools, only an executable version of HLS tools of each IP claimant is required.
- If any IP claimant refuses to provide an executable version of HLS tool or 'respectively generated RTL description in supervision of a legal entity'. Then, that specific claimant will be disqualified. As rightful owner will be willing to provide at least RTL description generated using its own HLS tool.
- The proposed approach is applicable for HLS tools that targets generation of application specific IP core (processors) of digital signal processing applications. The HLS tools that targets generation of general purpose processors does not fall in the scope of the proposed work.

8.3.2 Problem formulation

Given the IP core whose ownership is to be identified (termed as IP_{ID}) along with IPs generated from HLS tools of IP claimants (termed as IP_{CT} n, where 'n' signify the IP core generated using HLS tool of 'n^{th}' claimant) identify the rightful owner of the IP_{ID}.

8.3.3 Overview of proposed methodology

As discussed earlier, the proposed CFE based approach comprises of three major steps (a) IP core feature and data collection (b) IP core feature extraction, and (c) IP core validation. In the first step of the proposed approach the HLS tools are collected from the competing IP vendors. Subsequently, HLS tools are executed to generate IP cores with respect to each vendor's HLS tool (IP_{CT} n). Once all the IP cores are generated, IP_{CT} n are examined to identify features that can distinguish IP cores based on their originating HLS tool. A set of such features is termed as '*feature set'*. In the second step of the proposed step, feature extraction rules are devised. Based on these rules, features are extracted from IP_{CT} n and IP_{ID}. In the third and final step of the proposed approach, the ownership of IP_{ID} is awarded to vendor whose IP_{CT}'s feature set matches 100% with feature set of IP_{ID}. The upcoming section demonstrate the proposed approach with the help of a case-study

8.4. Case study

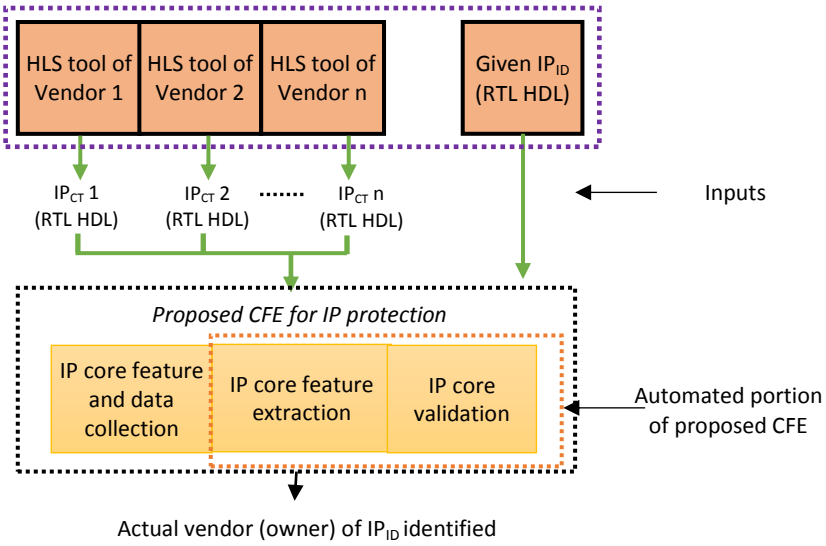


Fig 8.1 Process of resolving ownership conflict of a given IP core (IP_{ID}) using CFE

In this case study we have considered a scenario where seven claimants are legally competing for the ownership of IP_{ID} in court of law and court must award IP ownership to the rightful claimant. (Note: we have used seven claimants to demonstrate the proposed approach effectively). The case study considers industrial as well as academics HLS tools [12, 13, 17, 33, 52- 54]. In the first step of the proposed approach, the HLS tools are obtained from the respective IP vendor's company. Further, each tool is executed to generate a solution IP_{CT} n, (n = 1, …, 7). Subsequently, each IP_{CT} is studied to identify properties that can distinguish an IP core based on its parent HLS tool.

In practical scenarios, each company has their own set of proprietary algorithms/techniques that are uniquely developed by that company to advance state-of-art. These, properties are unique to that particular company thus features based on such properties are termed as 'unique feature'. Further, the proposed unique feature set include properties that are rarely found but can potentially be available in more than one advanced HLS tools. The unique features identified through our case study are: {reliability, trojan security, loop support, pipelining, chaining, multi-cycling, design objective}. Moreover, every HLS tool implements some common HLS framework. The framework can be implemented using different algorithms resulting in different properties of IP cores. These properties are examined to create a generic feature set: {Scheduling algorithm, resource type, bus width support}. A *feature set* comprising of both generic as well as unique features is created. Subsequently,

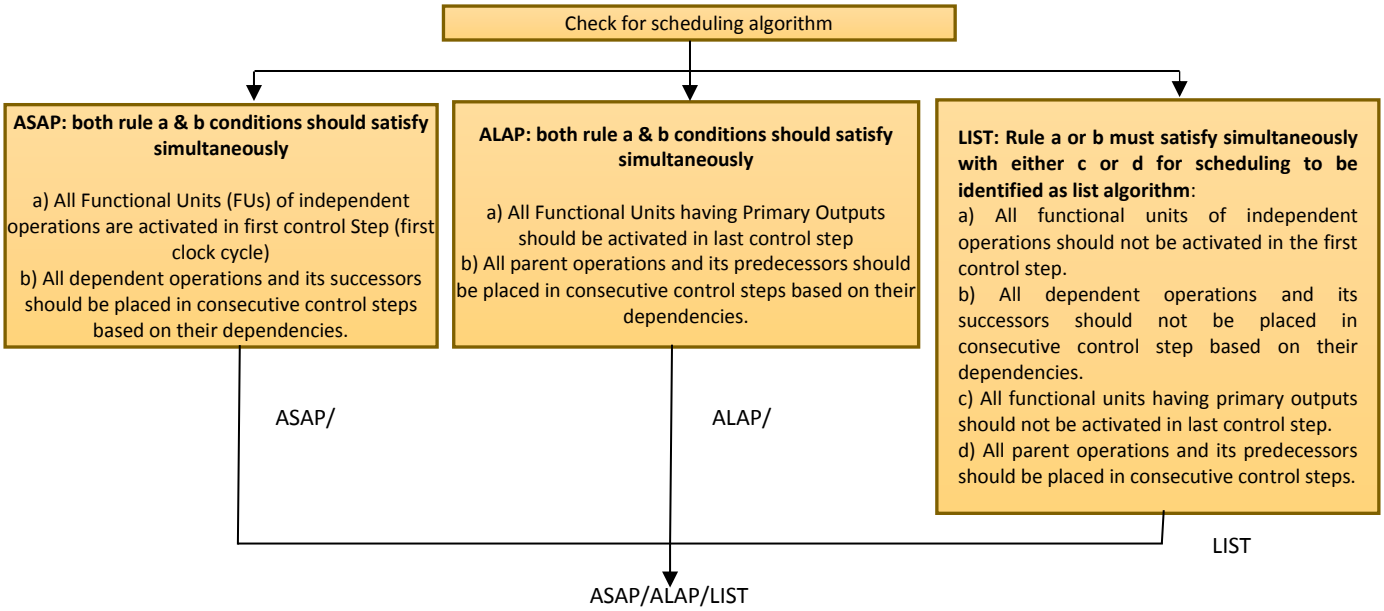


Fig. 8.2 Flow graph representing the feature extraction methodology for scheduling algorithm

in the second step of the proposed approach feature extraction rules are devised and features are extracted as discussed in upcoming sub-sections.

8.4.1 Scheduling algorithm

The feature extraction methodology to extract scheduling algorithm feature takes controller HDL file of IP core as input and identify the scheduling algorithm utilized during HLS of the IP core. The proposed technique classifies the scheduling algorithm as either ASAP scheduling, ALAP scheduling, or List scheduling (three most widely used scheduling algorithms [55-57]) (fig.8.2). The feature extraction rules to identify the scheduling algorithm used are:

- ASAP scheduling: A scheduling algorithm satisfying both the conditions (a) and (b) is ASAP scheduling.

(a) All functional units of independent operations should be activated in the first control step.

(b) All dependent operations and its successors should be placed in the consecutive control step based on their dependencies.

- ALAP scheduling: A scheduling algorithm satisfying both the conditions (c) and (d) is ALAP scheduling.

(c) All functional units having primary outputs should be activated in the last control step.

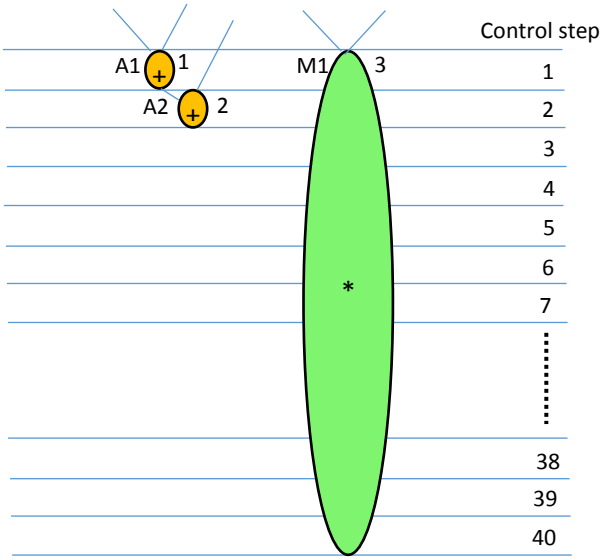


Fig. 8.3 Schedule displaying chaining of adder w.r.t. multiplier functional unit

(d) All parent operations and its predecessors should be placed in the consecutive control steps.

- LIST scheduling: A scheduling algorithm that satisfies conditions (e) or (f) along with either (g) or (h), then the scheduling algorithm is list scheduling.

(e) All functional units of independent operations should not be activated in the first control step.

(f) All dependent operations and its successors should not be placed in the consecutive control step based on their dependencies.

(g) All functional units having primary outputs should not be activated in the last control step.

(h) All parent operations and its predecessors should not be placed in the consecutive control steps.

This feature distinguishes (HLS tools utilized for creating) IP_{ID} and $IP_{CT}n$. If IP_{ID} utilizes different scheduling algorithm than $IP_{CT}n$, then HLS tool that generates $IP_{CT}n$ cannot be the rightful owner.

8.4.2 Resource configuration type

The resource configuration type feature extraction methodology takes datapath HDL file of IP core as input. Further, HDL file is examined to identify the different type of resources (functional units) utilized in the RTL datapath of

the IP core. For instance, if an IP core have adder, subtractor and multiplier resources, the resource config type feature is represented as: A, S, M.

```
component Adder
port (enable_R1 : in    std_logic;
      Data_out7 : in    std_logic_vector (15 downto 0);
      Data_out8 : in    std_logic_vector (15 downto 0);
      Data_in9  : out   std_logic_vector (15 downto 0));
end component;
```

8.4.3 Chaining

Chaining is an optimization technique that targets reduction of schedule delay. The concept of chaining can be understood with the help of an exemplary schedule shown in fig.8.3. In this example, two addition operations (1 & 2) are scheduled during a single execution of multiplication operation (3). If there was no chaining, operation 2 would have been scheduled at control step 41. Hence, overall delay of without chaining would be 41 control steps. The rule to identify the presence of chaining feature can be stated as: *if more than one operation of functional unit of type 'i' (FUi), is executed within a single execution of FUj; then, chaining feature is present in the IP core*. The chaining feature extraction rule is algorithmically represented in fig.8.4.

The chaining feature extraction algorithm takes controller HDL file of IP core as input and identifies the presence or absence of chaining feature in the given IP core. In Fig.8.4, 'n' represents the total number of functional units presents in the IP core. $CS_S(FU_i)$ *and* $CS_E(FU_i)$ represents the starting and ending control steps of i^{th} functional unit respectively. The starting and ending control steps of a FU can be determined from the controller HDL file. For instance, consider the controller shown in fig.8.5, the first multiplication operation starts its execution in control step 1 (MUL_EN_1<='1') and ends in control step 40 (MUL_EN_1<='0').Hence, $CS_S(multiplier) = 1$ and $CS_E(multiplier) = 40$. As shown in fig. 8.4, 'i' and 'j' are loop variables. The first loop runs for all FUs. The second loop allow all the FU_i such that $i \neq j$. Further, the first if allows comparison of i^{th} FU with only those j^{th} FUs that have started their execution after execution of i^{th} FU is started and FUs that have ended their execution before execution of i^{th} FU is ended. If the number of all such FUs is ≥ 2 then chaining feature is present in the IP core.

8.4.4 Bus width support

The Bus width support feature extraction algorithm takes datapath HDL file of IP core as input. Subsequently, top level entity HDL code is examined to identify the bus width of register components. A portion of HDL code is shown below:

```
component registerTp
   port ( tp   : in    std_logic_vector (7 downto 0);
          regtp: out   std_logic_vector (7 downto 0);
          strobe: in    std_logic);
end component;
```

As shown in the HDL code, register components are identified with the help of component's name. Further, the size of register components is determined as the highest size of the variable using statements such as `std_logic_vector (7 downto 0)`. Where, 7 down to 0 indicates variable size as 8 bits. Similarly, size of all the variables is evaluated and the largest variable's size is taken as size of the register. Similarly, largest size among all the register components present in an IP is taken as the bus width supported by the architecture of an IP core.

8.4.5 Data pipelining

The pipelining technique intends to reduce the delay of the overall design of the IP core. The data pipelining feature extraction algorithm takes datapath HDL file of IP core as input and identify the presence of pipelining feature as per the following equation:

Algorithm (Input: controller HDL of IP: Output: detection of chaining)

for (i=1 to n)
{
  for (j=1 to n && j != i)
  {
  if ($CS_S(FU_i) \leq CS_S(FU_j)$ && $CS_E(FU_i) \geq CS_E(FU_j)$)
    {
      if ($CS_E(FU_i) - CS_S(FU_i) \geq (CS_E(FU_j) - CS_S(FU_j))_1 + (CS_E(FU_j) - CS_S(FU_j))_2 + \ldots + (CS_E(FU_j) - CS_S(FU_j))_m$)
        {
          Chaining feature detected in IP core!
        }
    }
  }
}

Fig. 8.4 Proposed algorithm to detect chaining in an IP

$$(CS_E(N)_1 - CS_S(N)_1 + 1) > (CS_E(N)_2 - CS_E(N)_1 + 1) \quad \textbf{(8.1)}$$

Where, $CS_E(N)_1$ and $CS_E(N)_2$ denotes the ending control steps of data set 1 and 2. Similarly, $CS_S(N)_1$ denotes starting control step od data set 1. Further, $(CS_E(N)_1 - CS_S(N)_1 + 1)$ represents the execution time of data set 1. Likewise, $(CS_E(N)_2 - CS_E(N)_1 + 1)$ represents time difference between ending control step of data set 1 and ending control step of data set 2. Hence, in case when IP core does not incorporate pipelining feature. Both the right-hand side and left-hand side of eq. (8.1) will be equal. However, if pipelining is present in the IP core eq. (8.1) will be satisfied. For instance, consider schedule of an IP core shown in fig.8.6. The output of data set 1 and data set 2 are available in register Y at control step 42 and 82 respectively. Hence, The L.H.S of eq. 1 can be written as (42-1+1) = 42. Similarly, R.H.S. can be written as (82-42+1) = 41. Hence, eq. (8.1) is satisfied when pipelining is present in an IP core.

8.4.6 Multi-cycling

The multi-cycling feature extraction algorithm takes controller HDL file of IP core as input. Subsequently, on examining HDL code if there is a functional

```
entity control_unit is
port(
     clock, reset: in std_logic;
          :
          :
          :
          ADD_EN_1, ADD_EN_2, MUL_EN_1 : out
std_logic;
          REG_Y : out std_logic;
          :
          :
          :
          );
end control_unit;


architecture Behavioral of control_unit is
signal CS: INTEGER RANGE 0 TO 19;
signal count: INTEGER RANGE 0 TO 10;
signal busy : std_logic;
 begin
  process(clock,reset)
   begin

        if (clock'event and clock='1') then
           if(reset='0')then
             if CS =0 then
                clk<='1';
                        REG_IP_A_EN <='1';
                        REG_IP_B_EN <='1';
                        REG_IP_C_EN <='1';
                        REG_IP_D_EN <='1';
                        REG_IP_E_EN <='1';
             end if;
```

Fig. 8.5(a) Portion of a HDL code

unit whose execution time span more than 1 control step, then multi-cycling feature is said to be present in the IP core. In other words, if a functional unit's operation ends at control step greater than the starting control step (eq.8.2), then multi-

```
                CS <= CS +1;
          end if;


--------------------------------CONTROL STEP 1-------------------------

if CS=1 then
        if count=first_count
                        ADD_EN_1<='1';
                        MUL_EN_1<='1'; //start
of multiplication operation//
                        count <= count+1;
     end if;
        :
        :
        If count = last_count
                        ADD_EN_1 <= '0';
                        count <= '0';
        end if;
                        CS <= CS+1;
end if;


--------------------------------CONTROL STEP 2-------------------------

if CS=2 then
        if count= first_count
                        ADD_EN_1<='1';
                        count <= count+1;
     end if;
        :
        :
        :
        If count = last_count
                        ADD_EN_1 <= '0';
                        count <= '0';
        end if;
                        CS <= CS+1;
end if;
:

--------------------------------CONTROL STEP 40-------------------------

if CS=40 then
        if count= first_count
                        count <= count+1;
                        REG_IP_A_EN <='0';
                        REG_IP_B_EN <='0';
                        REG_IP_C_EN <='0';
                        REG_IP_D_EN <='0';
                        REG_IP_E_EN <='0';
     end if;
        :
        :
        :
        If count = last_count
                        MUL_EN_1 <= '0'; //end
of multiplication operation//
                        REG_IP_A_EN <='1';
                        REG_IP_B_EN <='1';
                        REG_IP_C_EN <='1';
                        REG_IP_D_EN <='1';
                        REG_IP_E_EN <='1';
                        count <= '0';
        end if;
                        CS <= CS+1;
end if;


--------------------------------CONTROL STEP 41-------------------------

if CS=41 then
        if count= first_count
                        count <= count+1;
                        ADD_EN_1 <= '1';
                        ADD_EN_2 <= '1';
                        MUL_EN_1 <= '1';
```

Fig. 8.5(b) Portion of a HDL code

```
                        REG_Y <='0';
     end if;
        :
        :
        :
        If count = last_count
                        ADD_EN_1 <= '0';
                        ADD_EN_2 <= '0';
                        count <= '0';
        end if;
                        CS <= CS+1;
end if;


--------------------------CONTROL STEP 42-------------------------

if CS=42 then
        if count= first_count
                        REG_Y <='1';//output of
data set 1 available//
                        count <= count+1;
                        ADD_EN_1 <= '1';
     end if;
        :
        :
        :
        If count = last_count
                        ADD_EN_1 <= '0';
                        count <= '0';
        end if;
                        CS<=CS+1;
end if;


--------------------------CONTROL STEP 80-------------------------

if CS=80 then
        if count= first_count
                        count <= count+1;
                        REG_IP_A_EN <='0';
                        REG_IP_B_EN <='0';
                        REG_IP_C_EN <='0';
                        REG_IP_D_EN <='0';
                        REG_IP_E_EN <='0';
     end if;
        :
        :
        :
        If count = last_count
                        MUL_EN_1 <= '0';
                        REG_IP_A_EN <='1';
                        REG_IP_B_EN <='1';
                        REG_IP_C_EN <='1';
                        REG_IP_D_EN <='1';
                        REG_IP_E_EN <='1';
                        count <= '0';
        end if;
                        CS<=CS+1;
end if;


--------------------------CONTROL STEP 81-------------------------

if CS=81 then
        if count= first_count
                        count <= count+1;
                        ADD_EN_1 <= '1';
                        ADD_EN_2 <= '1';
                        MUL_EN_1 <= '1';
                        REG_Y <='0';
     end if;
        :
        :
        :
        If count = last_count
                        ADD_EN_1 <= '0';
                        ADD_EN_2 <= '0';
                        count <= '0';
```

Fig. 8.5(c) Portion of a HDL code

```
            end if;
                                        CS <= CS+1;
end if;

---------------------------------CONTROL STEP 82--------------------------
if CS=82 then
            if count= first_count
                                        REG_Y <='1';//output of dataset 2
available//
                                        count <= count+1;
                                        ADD_EN_1 <= '1';
    end if;
            :
            :
            :

            If count = last_count
                                        ADD_EN_1 <= '0';
                                        count <= '0';
            end if;
                                        CS<=CS+1;
end if;

:
:
----------------------CONTROL STEP 4001(for 100 data set) -------
--if (clock'event and clock='1') then
   elsif(reset='1')then
                                        count<=0;
            end if;
end if;
--end if;
end process;
--count1 <=count;
end Behavioral;
```

Fig. 8.5(d) Portion of a HDL code

cycling is present in the IP core:

$$CS_E(FU_i) > CS_S(FU_i) \quad \textbf{(8.2)}$$

8.4.7 Design Objective

The design objective feature extraction methodology takes executable HLS tool's interface as input. By examining the user interface, various design objectives / constraints supported by that particular HLS tool such as area, power, delay, etc. can be identified.

8.4.8 Reliability

Reliability is an advanced feature and typically found in sophisticated HLS tools. Reliability can be incorporated in the IP core in various ways such as security/tolerance against permanent faults [58], intermittent fault [59], or transient fault [60], etc. In our proposed approach, we have considered recent reliability handling techniques that uses dual modular redundancy (DMR) such as [17], [54]. Note, there are other techniques to generate reliable IP core

using HLS. However, the proposed approach has considered only recent DMR based techniques.

The reliability feature extraction methodology takes datapath HDL (RTL code) of IP core as input. Subsequently, the top level entity HDL code of the IP core is examined to identify the presence of DMR. If a top level entity HDL contains a comparator component that takes two input signals coming from output register of module 1 (*output register signal 1*), and output register of module 2 (*output register signal 2*), and its output signal is the final output of the IP core. Then such a comparator component indicates the presence of DMR structure, thereby indicating presence of reliability feature in the IP core. An exemplary comparator's port map is: *port map (output register signal 1, output register signal 2, comparator output signal)*.

8.4.9 Loop support

The loop support feature extraction methodology takes input application file of the executable HLS tool as input. The input file considered in this case-study can be a control intensive application (in the form of control data graph (CDFG)) or a data intensive application (in the form of data flow graph). The CDFG application typically contains the maximum iterations value. However, as DFG applications doesn't contain any iteration information. Hence, this property of input application can help distinguish HLS tools that supports CDFG application from those who don't. The feature is termed as loop support feature. This feature tries to remove HLS tools that does not support loop based CDFG applications. For instance, if IP_{ID} is generated for some CDFG application such as FIR, then all the HLS tools that does not support loop based CDFGs will be eliminated.

8.4.10 Trojan security

Similar to the reliability, trojan security is also one of the advanced features used in highly sophisticated HLS tools. Trojan security can be understood as detection of hardware trojans in an IP core. The typical approach to identify hardware trojans utilizes hardware resources from at least two different vendors and a DMR system is designed [53].

The trojan security feature extraction methodology takes datapath HDL file and module library as input. Subsequently, the top level entity datapath is examined to identify a comparator component that takes two inputs, one each from the primary output of module 1 (as *output signal 1*) and module 2 (as *output signal 2*). Moreover, the final output of the IP core is the output of the comparator (*comparator output signal*) then dual modular redundancy is detected. Additionally, input module library of the HLS tool is examined to identify whether modules from more than 1 (at least 2) vendors are present or not? If, DMR as well as presence of hardware resources from multiple vendors are detected then, HLS tool supports trojan security feature.

The upcoming subsection describes the third and the final step of the proposed methodology.

IP validation: Once all the features are extracted, the feature set of IP_{ID} is compared with feature set of every competing HLS tool and ownership is awarded to the IP vendor whose *feature set* matches exactly (100%) with the *feature set* of the IP_{ID}. The following equation is utilized to evaluate the match percentage (m) between feature sets of IP_{ID} and IP_{CT}n

$$m = \frac{Number\ of\ matching\ features}{Total\ \ number\ of\ features\ in\ feature\ set} * 100 \qquad \textbf{(8.3)}$$

In a very rare case, the feature set of more than one HLS tool will match exactly with feature set of IP_{ID}. In such a scenario, number of features can be increased for achieving better results. However, note that such a case is *very rare*, as proposed methodology incorporates *unique features* along with generic features. Further, in case if none of the competing HLS tool's feature set matches 100% with feature set of IP_{ID} then ownership will not be awarded to any of the competing vendors.

8.5. Summary

The proposed approach presents a novel computational forensic engineering based methodology for resolving false claim of ownership of DSP IP cores. Further, the proposed methodology introduces a novel feature-set comprising of ten features. Feature extraction rules for extracting these features were presented. Based on these rules, feature-sets of IP_{ID} and IP_{CT}n were obtained

and matched. Finally, the IP ownership was award to the claimant whose IP_{CT}'s feature-set matches exactly with the feature set of IP_{ID}.

The proposed approach is compared with watermarking based approaches for resolving ownership conflicts. The proposed approach is found to be more reliable as it incurs zero-overhead (due to lack of signature-insertion step) and has no known attack in comparison with watermarking based approaches (as they vulnerable to reverse engineering based attack such as signature tampering) [5].

Chapter 9

Experimental Results and Analysis

This chapter discusses the experimental results and analyses of the proposed methodologies presented in this thesis.

9.1. Results and analysis: Methodology for generating a DSP IP core that is simultaneously secure/resilient against multi-cycle temporal and multi-unit spatial effect of transient fault.

This section discusses the experimental results of the proposed methodology presented in chapter 3 of this thesis. The proposed approach is implemented in java and executed on Intel core i5 3210M processor with 3MB cache, 4GB DDR3 primary memory and frequency of 2.5GHz. The proposed methodology is applied on DSP IP benchmarks of [61]. Note that the proposed approach is the first work in the literature which simultaneously provides resiliency against multi-cycle (k_c) and multi-unit (k_m) transient fault affected due to single radiation strike at behavioral/architecture level. The proposed approach simultaneously achieves temporal and spatial resiliency through a novel unification of high level synthesis and physical level design. All prior work that handled multiple transient fault were at lower levels such as gate-level or transistor level. Nevertheless, comparisons to baseline duplication (non-security DMR designs) and normal designs (no duplication & security constraints) for chip area, delay and power has been reported in Tables 9.1, 9.2, and 9.3. The results are compared on the basis of following design metrics

a) Chip area of the multi-unit (km) transient fault resilient floorplan.
b) Delay of the multi-cycle (kc) transient fault resilient DMR schedule.
c) Power of the transient fault resilient design.

9.1.1 Area comparison

Table 9.1 shows the area comparison of proposed fault resilient design with non-resilient design. It is easily evident that the proposed approach incurs a modest area overhead in comparison with non-resilient design. This is because imposing k_m-unit MTF resiliency constraint affects the placement of modules within the floorplan. For example consider DCT benchmark with resource

constraint X_i = (7M, 4A), the floorplan which does not follow our k_m-unit MTF resiliency constraint, results in a chip area of 556 sq.units. On the contrary, the floorplan which abides by our k_m-unit MTF resiliency constraint results in a chip area of 590.75sq.units. Thus, an area overhead of 34.75 sq. units due to imposing resiliency constraint is visible. The results are compared for large value of kc (=10) and km (=4), as large values are likely to produce high overhead. However, as evident from the results, the proposed approach incurs a nominal overhead even for significantly large strength of transient fault.

9.1.2 Delay comparison

The delay comparison of the proposed approach with non-resilient design is reported in table 9.2. The designs generated for large kc-cycle transient fault resiliency constraint (such as kc = 10) results in delay overhead compared to both non-transient fault resilient schedules (with and without duplication). This is because large resiliency constraint value creates more chances of hardware conflicts, therefore to avoid transient fault hazards operations must be pushed in lower control step (thereby increasing delay overhead).

9.1.3 Power comparison

Table 9.2. Results comparison of proposed 10-cycles, 4-units transient fault resilient designs with non-transient fault resilient in terms of chip area and corresponding overhead

Table 9.1. Results comparison of proposed 2-cycle, 2-unit transient fault resilient design with non-transient fault resilient in terms of chip area and corresponding overhead

Benchmark	User Resource Constraint	Chip area in sq. units (Non-transient fault resilient DMR design)	Chip area in sq. units (k_m-unit transient fault resilient design)	Chip area overhead in sq.units	Benchmark	User Resource Constraint	Chip area in sq. units (Non-transient fault resilient DMR design)	Chip area in sq. units (k_m-unit transient fault resilient design)	Chip area overhead in sq. units
ARF	4A, 4M	556	556	0.00	EWF	4A,2M	607.25	654.75	47.5
	3A, 3M	428	556	128		3A,2M	465	654.5	189.5
	2A, 2M	321	321	0.00		2A,2M	465	561	96
BPF	3A, 4M	556	556	0.00	FFT	8A,4M	396	445.5	49.5
	3A, 3M	316	428	112		8A,3M	376	423	47
	3A, 2M	401.25	428	26.75		8A,2M	262.5	374.5	112
DCT	8A, 4M	590.75	695	104.25	FIR	8A,8M	556	556	0.00
	7A,4M	556	590.75	34.75		7A,7M	516	556	40
	6A,4M	516	556	40		6A,6M	516	556	40

		Delay in ns			Delay overhead		Chip area	Chip area	
EWF	3A,2M	0.97	1.364	1.366	0.39	0.002	465	654.5	189.5
	2A,2M	1.03	1.752	1.944	0.72	0.192	465	561	96
FFT	8A,4M	0.39	0.46	0.46	0.07	0.000	396	562.5	49.5
	8A,3M	0.46	0.65	0.658	0.19	0.008	376	454.75	47
	8A,2M	0.46	0.85	0.856	0.39	0.006	262.5	428	112
FIR	8A,8M	0.57	0.64	0.644	0.07	0.004	556	764.5	0.00
	7A,7M	0.58	0.64	0.646	0.06	0.006	516	625.5	40
	6A,6M	0.58	0.64	0.646	0.06	0.006	516	625.5	40
JPEG IDCT	24A,24M	0.520	0.59	0.916	0.396	0.326	1816	1972	156
	20A,20M	0.522	0.654	0.98	0.458	0.326	1560	1880	320

The power comparison of the proposed approach with non-resilient design is reported in table 9.3. A small overhead is observed for some designs of the proposed approach due to imposing of simultaneous multi-cycle & multi-fault resiliency constraints. This is because, imposing the constraints may cause increase in register /multiplexer count (due to possibility of a different schedule/binding) in some cases, resulting in slightly higher power magnitude. The power value reported includes total power due to functional units (hardware), steering logic (multiplexer, demultiplexer, interconnects) and storage elements. The results shows that with minimal power overhead

sometimes (while no power overhead for most cases), the proposed approach generates DSP IP cores that are simultaneous resilient against multi-cycle and multi-unit transient fault.

Table 9.3. Power comparison results of proposed 10-cycle, 4-unit multiple transient fault resilient designs and non-transient fault resilient DMR designs

Benchmark	User Resource Constraint	Power in µW (Non-transient fault resilient design)	Power in µW (10-cycle, 4-unit transient fault resilient DMR design)	Power overhead in µW
ARF	2A 2M	9.605	10.117	0.512
	3A 3M	9.022	9.278	0.256
	4A 4M	8.840	8.840	0.00
BPF	3A 2M	8.110	8.110	0.00
	3A 3M	8.162	9.058	0.896
	3A 4M	8.572	8.956	0.384
DCT	6A 4M	14.598	14.598	0.00
	7A 4M	13.821	14.077	0.256
	8A 4M	12.579	12.579	0.00
EWF	2A 2M	9.394	9.522	0.128
	3A 2M	11.109	11.493	0.384
	4A 2M	10.911	10.911	0.00
FFT	8A 2M	8.486	8.486	0.00
	8A 3M	10.308	10.308	0.00
	8A 4M	9.511	9.511	0.00
FIR	6A 6M	8.322	8.322	0.00
	7A 7M	8.478	8.478	0.00
	8A 8M	8.928	8.928	0.00
JPEG IDCT	20A 20M	39.398	39.398	0.00
	24A 24M	36.875	36.875	0.00

9.2. Results and analysis: Methodology for generating a DSP IP core that is simultaneously tolerant against multi-cycle temporal and multi-unit spatial effect of transient fault.

The methodologies for generating DSP IP core tolerant against multi-cycle and multi-unit transient fault has been discussed in chapter 4 for data intensive applications and in chapter 5 for loop based control intensive applications. This section presents results and analysis of both these methodologies. The proposed methodologies are implemented in java and executed on Intel core i5 3210M processor with 3MB cache, 4GB DDR3 primary memory and frequency of 2.5GHz. The proposed methodologies are implemented on data intensive applications such as BPF, DCT, DWT as well as loop-based control intensive applications such as Differential equations, FFT, FIR, and Test_case of express benchmark suite [61]. The experimental results thus obtained are analyzed based on following metrics

a) Fitness/cost of the explored kc-cycles, km-units tolerant design solution.
b) Power consumption of the explored kc-cycles, km-units tolerant design solution.
c) Rectangular chip area of the km-units fault tolerant floorplan.
d) Delay of the kc-cycles fault tolerant scheduled C/DFG TMR

As discussed earlier in the chapter 2, there is no work in the literature that simultaneously provide tolerance against multi-cycle and multi-unit transient fault. A prior work that closely relates to the proposed approaches is [12]. The results of comparison of proposed approach with [12] are tabulated in table 9.4, 9.5, 9.6 and 9.7 respectively. The comparison of the proposed approach with [12] is performed for multi-cycle k_c=4 (equivalent to 400ps) [12,24] & multi-unit k_m=4 (equivalent to 3072nm) [62,63,64] transient fault impact. However, note that the proposed methodologies are applicable for any value of kc and km.

As reported in table 9.4, the proposed approach always generates low cost (better fitness) tolerant design solution compared to [12]. This is due to integrated PSO-DSE framework that explores low-cost transient fault tolerant

design. On the other hand, [12] is not capable to obtain a low-cost design solution due to lack of optimization framework in the tolerance algorithm, besides being deficient in providing tolerance against spatial effects of transient fault. Additrionally, [12] is not capable of performing pre-processing of unrolling factor (especially filters UF with large sequential loops) and exploring a combination of loop UF for control intensive applications. Thus [12] provides tolerance without appropriate unrolling and produces expensive fault tolerant solution. Further, due to lack of design space exploration framework, the design solution of [12] never produces low cost results. For comparison purpose, the design solution for [12] is based on the particle encoding with mid-hardware configuration. For example, as shown in table 9.4, for DCT benchmark, the proposed approach has explored an low cost solution having design cost of 0.37, while [12] yielded a high-cost solution with a design cost of 0.49. Thus, relative cost improvement of 0.12 is achieved. Similarly, cost improvements for other benchmarks are reported in table 9.4. An average cost reduction of ~30 % is achieved for benchmarks tabulated in table 9.4.

As evident from table 9.5, a significant reduction in power consumption of proposed approach has been obtained with respect to [12]. The power reported in table 9.5 is evaluated based on the following power model.

Power Model: For a given functional resource, the power consumption (adopted from [17]) can be given as:

$$P_T^{FT-TMR} = \sum_{i=1}^{Max} (K(FU_i) \cdot p(FU_i)) \qquad \textbf{(9.1)}$$

Where, $p(FU_i)$ is the power consumed by FU_i (as per 15nm technology scale open cell NanGate Library [31]); $K(FU_i)$ is the number of instances of FU_i used in the FT-TMR design and 'Max' indicates the index of the last FU type used in the FT-TMR design.

The proposed approach implements PSO based DSE for generating fault tolerant solution based on appropriate combination of loop unrolling factor and hardware resources compared to [12] which does not perform any optimization to handle overhead. Thus, proposed approach results in

significantly lesser power consumption. For example, as shown in table 9.5, for DCT benchmark, the proposed approach has explored a fittest design solution having power of 2.49 uW, while [12] yielded a design cost 5.05uW. Thus, relative power reduction of 2.56uW is achieved. Similarly, power reductions for other benchmarks are reported in table 9.5. An average power reduction of ~57 % is achieved for benchmarks tabulated in table 9.5.

Table 9.6 and 9.7 shows the area and delay value of the obtained design solutions for the standard benchmarks. As represented in table 9.6, area of proposed approaches is lesser than the area of [12] (for all the benchmarks) as design solution explored through proposed approach obtains lesser number of hardware resources and unrolling factor compared to [12], which does not explore appropriate combination of unrolling factor and hardware as well as does not perform preprocessing of unfit unrolling factors. Further, as shown in table 9.7 significantly larger number of resources are utilized in [12], hence due to higher parallelization, delay of [12] may sometimes be lesser compared to proposed approach. Nonetheless, the overall design cost and power of [12] is significantly higher than proposed approach due to lack of provision of optimization technique during tolerance design.

Table 9.4. Cost comparison of proposed method with [12] for k_c=4 & k_m=4

Benchmark	Design Solution of [12]	Design Cost of [12]	Design Solution of proposed approach	Design Cost of proposed approach	Reduction in Design Cost %	Benchmark	Design Solution of [12]	Cost of [12]	Design Solution of proposed approach	Design Cost of proposed approach	Reduction in cost %
BPF	5A, 6M	0.53	3A, 2M	0.37	**30.18 %**	**DIFF_EQ**	12A, 12S, 36M, 2C, UF=8	0.30	2A, 2S, 6M, 2C, UF=4	0.18	**40 %**
DCT	12A,6M	0.49	5A, 3M	0.37	**24.48 %**	**FFT**	26A,12S, 24M,2C, UF=8	0.32	4A, 5S, 4M, 2C, UF=4	0.20	**37.5 %**
DWT	6A, 8M	0.57	3A, 2M	0.42	**26.31 %**	**FIR**	2A, 12M, 2C, UF=8	0.41	2A, 3M, 2C, UF=4	0.28	**31.7 %**
						TEST_CASE	14A,12M, 2C, UF=8	0.38	4A, 5M, 2C, UF=4	0.30	**21 %**

Table 9.5. Comparison of power of proposed method with [12] for k_c=4 & k_m=4

Benchmark	Design Solution of [12]	Power of [12] (in µW)	Design Solution of proposed approach	Proposed power (in µW)	Reduction in power %	Benchmark	Design Solution of [12]	Power of [12] (in µW)	Design Solution of proposed approach	Proposed power (in µW)	Reduction in power %
BPF	5A, 6M	4.84	3A, 2M	2.95	**39.04 %**	**DIFF_EQ**	12A, 12S, 36M, 2C, UF=8	23.60	2A, 2S, 6M, 2C, UF=4	4.20	**82. 20 %**
DCT	12A,6M	5.05	5A, 3M	2.49	**50.69 %**	**FFT**	26A, 12S, 24M, 2C, UF=8	19.37	4A, 5S, 4M, 2C, UF=4	4.38	**77.38 %**
DWT	6A, 8M	4.86	3A, 2M	1.97	**59.46 %**	**FIR**	2A, 12M, 2C, UF=8	6.92	2A, 3M, 2C, UF=4	2.72	**60.69 %**
						TEST_CASE	14A,12M, 2C, UF=8	8.22	4A, 5M, 2C, UF=4	5.61	**31.75 %**

Table 9.6. Comparison of area of proposed method with [12] for k_c=4 & k_m=4 (Note : 1 unit = 768nm)

Benchmark	Design Solution of [12]	Area of [12] (in Sq. units)	Design Solution of proposed approach	Area of proposed approach (in Sq. units)	Benchmark	Design Solution of [12]	Area of [12] (in Sq. units)	Design Solution of proposed approach	Area of proposed approach (in Sq. units)
BPF	5A, 6M	500.0	3A, 2M	406.25	**DIFF_EQ**	12A, 12S, 36M, 2C, UF=8	1640.5	2A, 2S, 6M, 2C, UF=4	593.75
DCT	12A, 6M	531.25	5A, 3M	437.5	**FFT**	26A, 12S, 24M, 2C, UF=8	1247.75	4A, 5S, 4M, 2C, UF=4	593.75
DWT	6A, 8M	531.25	3A, 2M	406.25	**FIR**	2A, 12M, 2C, UF=8	625.0	2A, 3M, 2C, UF=4	468.75
					TEST_CASE	14A, 12M, 2C, UF=8	687.5	4A, 5M, 2C, UF=4	593.75

Table 9.7. Comparison of delay of proposed method with [12] for k_c=4 & k_m=4

Benchmark	Design Solution of [12]	Delay of [12] (in ns)	Design Solution of proposed approach	Delay of proposed approach (in ns)	Benchmark	Design Solution of [12]	Delay of [12] (in ns)	Design Solution of proposed approach	Delay of proposed approach (in ns)
BPF	5A, 6M	2.1	3A, 2M	3.1	**DIFF_EQ**	12A, 12S, 36M, 2C, UF=8	1.7	2A, 2S, 6M, 2C, UF=4	5.8
DCT	12A, 6M	1.9	5A, 3M	3.0	**FFT**	26A, 12S, 24M, 2C, UF=8	4.1	4A, 5S, 4M, 2C, UF=4	8.7
DWT	6A, 8M	1.6	3A, 2M	2.5	**FIR**	2A, 12M, 2C, UF=8	2.5	2A, 3M, 2C, UF=4	3.8
					TEST_CASE	14A, 12M, 2C, UF=8	1.8	4A, 5M, 2C, UF=4	3.8

9.3. Results and analysis: Methodology for generating a low-cost, highly secure, functionally obfuscated DSP IP core

This section discusses the experimental results of the proposed methodology presented in chapter 6 of this thesis. The proposed approach and methodology presented in [21] have been implemented in java and executed on Intel Core i5 3210M CPU with 4GB DDR3 primary memory and processor frequency of 2.5 GHz. The proposed methodology generates a low-cost, low-power, highly secured functionally obfuscated IP core. The power and delay values are based on 15 nm NanGate library [31]. The proposed approach and [21] are tested on Express Benchmark suite [61]. The results obtained are analyzed based in terms of the following parameters:

a. Comparison of strength of obfuscation of proposed approach with [21] from an attacker's perspective.

b. Power comparison of proposed approach with [21].

The strength of obfuscation parameter represents the complexity for an attacker to reverse engineer the design netlist. The strength of obfuscation of the proposed approach and [21] are reported in table 9.8. This is an optimistic estimate, since for each key guess input output pattern of the circuit is also verified. For [21] since each key gate is encoded with 1 bit, therefore number of key gates is equal to number of encoded key bits. For example as shown in table 9.8, the number of key bits for JPEG IDCT is 432, therefore, number of key gates added is 432. The proposed approach is able to provide an enhancement in the strength of obfuscation compared to [21]. For example, in case of JPEG IDCT benchmark, the attacker has to apply 3.83 e+404 brute-force input combinations to decipher the netlist. Similarly, for [21] the brute-force effort is 1.1 e+130. The strength of obfuscation enhancement through proposed approach is 3.46 e+274 times of [21].

As reported in table 9.9, the leakage power consumption of the proposed obfuscation approach is less than the [21]. This is because proposed obfuscation technique integrates PSO-DSE framework for exploration of low-cost obfuscated design solution. Therefore, the design solution explored by the proposed approach consumes less power compared to [21]. An average

reduction of 9.94 % in static power consumption of proposed approach is observed compared to [21]. The obfuscated cost of the proposed approach and [21] are reported in table 9.10. An average cost reduction of 6.35% is obtained through proposed obfuscation approach. As discussed earlier, the low-cost solution is obtained since the proposed approach integrates PSO-DSE framework. Thus, even though there is marginal delay overhead due to ILBs, however it gets optimized during overall design delay reduction through PSO-DSE. Altogether, the proposed approach on comparison with [21] yielded a power reduction of ~ 10 %, design cost reduction of ~ 6.5 % and security enhancement (strength of obfuscation) of at least 4.29 e+9 times.

Table 9.8. Strength of obfuscation comparison of proposed functionally obfuscated approach w.r.t. [21]

DSP Core Benchmarks [19]		No. of key-bits encoded for proposed obfuscation (r)	Strength of obfuscation of proposed approach	No. of key-bits encoded for [21] (r)	Strength of obfuscation of [21]	Strength of obfuscation enhancement of proposed approach (by factor of)
Name	Size					
IIR	**9919**	192	6.28 e+57	96	7.92 e+28	7.92 e+28
Mesa Horner	**10842**	192	6.28 e+57	80	1.2 e+24	5.19 e+33
DWT	**10958**	128	3.40 e+38	96	7.92 e+28	4.29 e+ 9
ARF	**14833**	256	1.15 e+77	112	5.19 e+33	2.23 e+43
FIR	**16047**	320	2.13 e+96	144	2.23 e+43	9.57 e+52
JPEG IDCT	**42710**	1344	3.83 e+404	432	1.10 e+130	3.46 e+274
Mesa Interpolate	**48853**	832	2.86 e+250	464	4.76 e+139	6.01 e+110

Table 9.9. Power comparison of proposed functionally obfuscated approach w.r.t. [21]

Benchmark	**Explored proposed functionally obfuscated Design Solution**	**Gate count of netlist (proposed approach)**	**Power of proposed approach (in µW)**	**Design Solution of [21]**	**Gate count of netlist [21]**	**Power of [21] (in µW)**	**Gate Reduction (in %)**	**Power Reduction (in %)**
IIR	1A, 2M, µ=4	6444	20.146	2A, 4M	7649	24.850	15.75 %	18.92 %
Mesa Horner	1A, 2M, µ=4	6641	26.080	2A, 4M	7780	28.986	14.64 %	10.02 %
DWT	1A, 1M, µ=1	5745	25.586	3A, 3M	7324	31.365	21.55 %	18.42 %
ARF	2A, 2M, µ=3	7741	39.234	3A, 4M	8495	43.967	8.87 %	10.76 %
FIR	3A, 2M, µ=4	8112	41.864	4A, 5M	9436	45.274	14.03 %	7.53 %
JPEG IDCT	11A,10M,µ=2	23370	172.523	12A,15M	23998	178.843	2.61 %	3.53 %
Mesa Interpolate	8A, 5M, µ=4	18061	132.924	13A,16M	24932	155.673	27.55 %	14.61 %

Table 9.10 Cost comparison of proposed functionally obfuscated approach w.r.t. [21]

Benchmark	**Proposed functionally obfuscated Design Solution**	**Cost of proposed approach**	**Design Solution of [21]**	**Cost of [21]**	**Cost Reduction (in %)**
IIR	1A, 2M, µ=4	0.6810	2A, 4M	0.7427	8.30 %
Mesa Horner	1A, 2M, µ=4	0.6526	2A, 4M	0.6820	4.31 %
DWT	1A, 1M, µ=1	0.7549	3A, 3M	0.7708	2.06 %
ARF	2A, 2M, µ=3	0.5259	3A, 4M	0.5281	0.41 %
FIR	3A, 2M, µ=4	0.5638	4A, 5M	0.5853	3.67 %
JPEG IDCT	11A,10M,µ=2	0.3629	12A,15M	0.4455	18.54 %
Mesa Interpolate	8A, 5M, µ=4	0.3093	13A,16M	0.3573	13.43 %

9.4. Results and analysis: Methodology for analyzing the aging effect of NBTI stress on performance of DSP IP core

This section discusses the experimental results of the proposed methodology presented in chapter 7 of this thesis. The proposed investigation is performed on Altera cyclone II FPGA board EP2C20F484C7. The respective software program Quartus II version 7.2 run on Intel® Xeon® CPU with 4GB RAM at 3.10 GHz. Fig. 9.1 shows the datapath diagram of Nand based gate level implementation with its respective pin assignments. The gate level implementations have been analyzed based on following criteria

a) Change in Threshold Voltage Vs. Stress Time

b) Delay Degradation Vs. Stress Time

c) Delay degradation due to NBTI Stress and No-Stress for most threatful input vector.

d) Delay degradation due to NBTI Stress and No-Stress for different samples of input vector.

9.4.1. Change in Threshold Voltage Vs. Stress Time

NBTI stress affects several parameters of a device including threshold voltage, drain current, transconductance etc. In our experiments we have focused on the effect of NBTI stress on threshold voltage of the pmos. More the NBTI stress time, more is the increase in threshold voltage (as discussed in eqn. (7.1) & (7.2)). This has been shown by varying the stress time for evaluating the effect on threshold voltage. Fig. 9.2(a) shows the change in threshold voltage observed after applying NBTI stress for 1, 2 & 3 years respectively on ARF IP core for distinct values of stress probability. Stress probability as defined in [65] is the fraction of the time the pmos transistor is under stress (it represents the workload of the device). The value of stress probability is considered as number of control steps in which a gate is under NBTI stress out of total number of control steps.

9.4.2. Delay degradation Vs. Stress Time

Delay of the gate gets affected with change in threshold voltage (as shown in eqn. (7.3)). Thus, when threshold voltage of the pmos increases due to NBTI

stress, delay of the gate (corresponding to that pmos) also increases. This causes performance degradation of the entire datapath. However, it also depends on the input vector applied at the gates. This is because not all input vectors are capable of turning ON all (or majority of) the pmos in the critical path. Depending on the input applied, the number of pmos turned ON in the critical path changes. Thus, it is important to analyze the effect of each input vector on the critical path of the datapath, as critical path determines the delay of the circuit. Following process is performed to evaluate the delay of the gate level datapath for each input vector. First, for a specific test vector, the number of pmos in the critical path being turned ON is determined, followed by determination of ΔV_{Th} corresponding to a specific stress time (t). Once ΔV_{Th} is calculated, then the new threshold voltage (V_{Th}^{New}) corresponding to the pmos is calculated (using eqn. (7.2)). Subsequently, the V_{Th}^{New} is used to evaluate its gate delay (using eqn. (7.3)). In case a test vector is applied that does not turn a pmos of a gate ON, then the original threshold voltage corresponding to the nmos is used to evaluate delay of the gate. If a test vector affects both pmos and nmos of a gate, then the delay corresponding to the pmos is considered (as it is larger). Note: On applying a test vector if number of nmos being turned ON increases then total delay increases. This is because nmos transistors are in series in NAND gate representation. However, if number of pmos transistor being turned increases then delay doesn't increase as significantly as pmos transistors are connected in parallel in NAND gate representation. Fig. 9.2 (b) shows delay of the gate level datapath corresponding to each test (input) vector applied. As observed, the *red colored ones (1010,1000,0010,0000)* are most threatful as they all incur same maximum performance degradation. The *green colored ones (0011,1011,0111)* produces least delay degradation. Similar results were observed for other benchmarks. Table 9.11 shows delay after 1 year of continuous NBTI stress is applied on IIR core through each of the possible input vector combination. Similarly, the delay of ARF IP core is reported in table 9.12.

9.4.3. Delay degradation due to NBTI Stress and No-Stress for most threatful input vector (for varying Stress time)

Fig. 9.2(c) shows the delay of the gate level datapath of ARF under NBTI stress and no-stress for most threatful input vector say '0000' (i.e., the input vector which causes maximum delay degradation as obtained in previous section). In other words, we analyze in this section how much degradation occurs when NBTI stress is applied due to specific input vector in contrast to when no-NBTI stress occurs. No-stress here indicates a theoretical condition when NBTI stress does not affect the pmos of the gate (i.e. its threshold voltage and corresponding delay). Three possible cases have been investigated for stress time (1 year, 2 year and 3 year) on datapath. As expected, with increase in stress time, the delay of the datapath has increased (due to increase in threshold voltage of corresponding pmos of the gate). However, there is no effect on delay when no NBTI stress is considered as threshold voltage remains same. This trend of Fig.9.2(c) is likely to remain same as increase in stress time will always increase the threshold voltage.

9.4.4. Delay degradation due to NBTI Stress and No-Stress for most threatful input vector (for varying Stress time)

In this section we investigate the effect of different samples of input vector on the delay of the datapath for both NBTI stress and no-NBTI stress condition. We have selected three samples viz. 0000(causing maximum delay degradation), 0011 (causing minimum delay degradation) and 1101 (causing median delay degradation) for this analysis. Fig. 9.2(d) shows the impact on delay of the datapath for the chosen sample vectors for NBTI stress and no-stress condition. Similar trends are observed for all the tested benchmarks.

Table 9.11 Delay after applying 1 year of continuous NBTI stress of IIR Benchmark

Input Vectors	Individual Control Steps						Total Delay
	CS 1	CS 2	CS 3	CS 4	CS 5	CS 6	
0000	4731.9	4201.7	5993.8	6427.6	6427.6	6427.6	34210.2
0001	4731.9	4201.7	5993.8	6427.6	6427.6	6427.6	34210.2
0010	4729.6	3980.9	5993.8	6427.6	6427.6	6427.6	33987.1
0011	4725.1	3973.2	5756.4	6192.8	6413.9	6415.6	33477.0
0100	4731.9	4201.7	5993.8	6427.6	6427.6	6427.6	34210.2
0101	4731.9	4201.7	5993.8	6427.6	6427.6	6427.6	34210.2
0110	4729.6	3980.9	5993.8	6427.6	6427.6	6427.6	33987.1
0111	4725.1	3973.2	5756.4	6192.8	6413.9	6415.6	33477.0
1000	4727.4	4194.0	5993.8	6427.6	6427.6	6427.6	34198.0
1001	4727.4	4194.0	5993.8	6427.6	6427.6	6427.6	34198.0
1010	4731.9	3984.8	5993.8	6427.6	6427.6	6427.6	33998.3
1011	4727.4	3973.2	5756.4	6192.8	6413.9	6415.6	33479.3
1100	4713.4	4182.4	5967.2	6411.0	6412.7	6195.7	33882.4
1101	4713.4	4182.4	5967.2	6411.0	6412.7	6195.7	33882.4
1110	4713.4	3979.9	5967.2	6411.0	6412.7	6195.7	33679.9
1111	4726.5	3980.9	6193.7	6203.2	6420.1	6203.1	33727.5

Table 9.12 Delay after applying 1 year of continuous NBTI stress on ARF benchmark

Input Vector	CS1	CS2	CS3	CS4	CS5	CS6	CS7	CS8	CS9	CS10	CS11	CS12	CS13	CS14	CS15	CS16	CS17	CS18	C19	TOTAL
0000	8446	7925	8446	9390	8655	9390	9390	7925	9390	8659	7925	9390	9390	8659	7925	9390	9390	8655	8655	167004
0001	8429	7689	8429	8911	8176	8911	8176	7689	8911	8436	7918	8911	8911	8436	7918	8911	8911	8176	8176	160034
0010	8446	7925	8446	9390	8655	9390	9390	7925	9390	8659	7925	9390	9390	8659	7925	9390	9390	8655	8655	167004
0011	8429	7666	8429	8670	7937	8671	8671	7666	8675	8432	7913	8889	8889	8432	7913	8889	8889	7677	7677	158425
0100	8419	7438	8419	9390	8655	9390	9390	7438	9390	8173	7912	9390	9390	8173	7438	9390	9390	8655	8655	164502
0101	8428	7696	8428	8911	8176	8911	8911	7696	8911	8427	7911	8911	8688	8427	8123	8911	8911	8176	8176	160739
0110	8419	7438	8419	9390	8655	9390	9390	7438	9390	8173	7912	9390	9390	8173	7438	9390	9390	8655	8655	164502
0111	8418	7675	8418	8887	7930	8890	8663	7675	8668	8424	7903	8676	8453	8424	7903	8676	8676	8204	8204	158776
1000	8446	7925	8446	9390	8655	9390	9390	7925	9390	8659	7925	9390	9390	8659	7925	9390	9390	8655	8655	167004
1001	8429	7689	8429	8911	8176	8911	8176	7689	8911	8436	7918	8911	8911	8436	7918	8911	8911	8176	8176	160034
1010	8446	7925	8446	9390	8655	9390	9390	7925	9390	8659	7925	9390	9390	8659	7925	9390	9390	8655	8655	167004
1011	8429	7666	8429	8670	7937	8671	8671	7666	8675	8432	7913	8889	8889	8432	7913	8889	8889	7677	7677	158425
1100	8390	7911	8390	9350	8423	9350	9157	7911	9157	8646	7905	9378	9378	8646	7911	9378	9378	8419	8419	165504
1101	8409	7426	8409	8952	8217	9335	8952	8952	8952	8414	8414	8952	8952	8414	8107	8952	8952	8217	8217	163205
1110	8390	7911	8390	9350	8423	9350	9157	7911	9157	8646	7905	9378	9378	8646	7911	9378	9378	8419	8419	165504
1111	8415.	7683	8415	9367	8631	9367	9365	7899	9367	8411	7898	9367	9367	8411	8108	9367	9367	8632	8632	166079

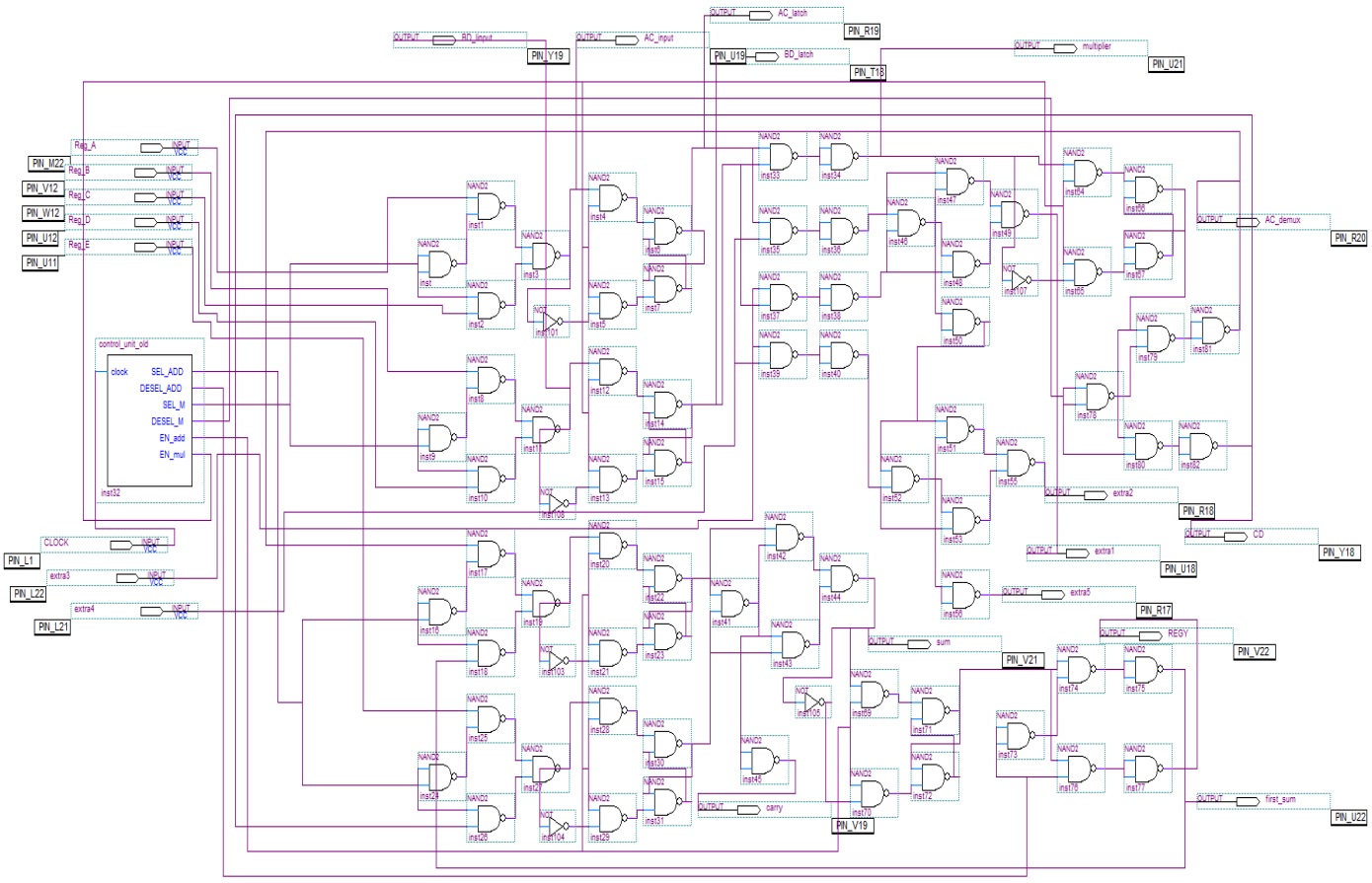
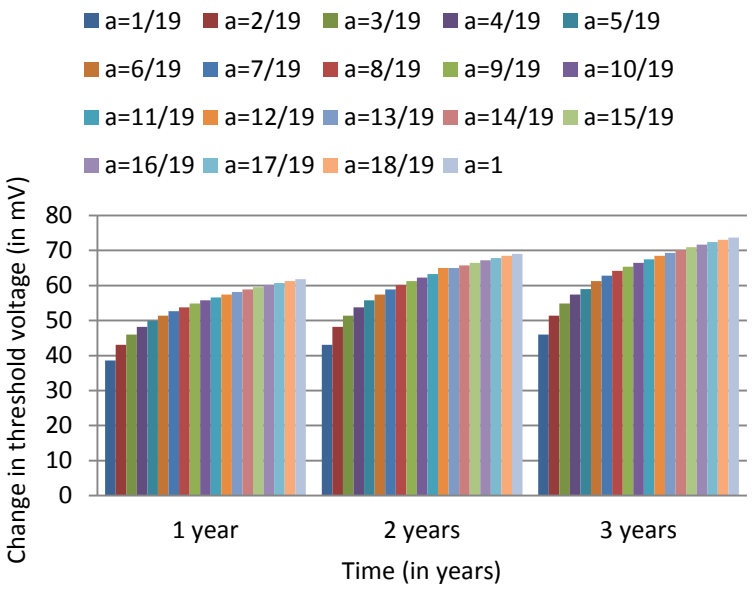
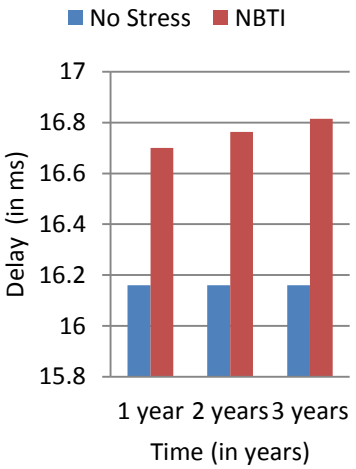


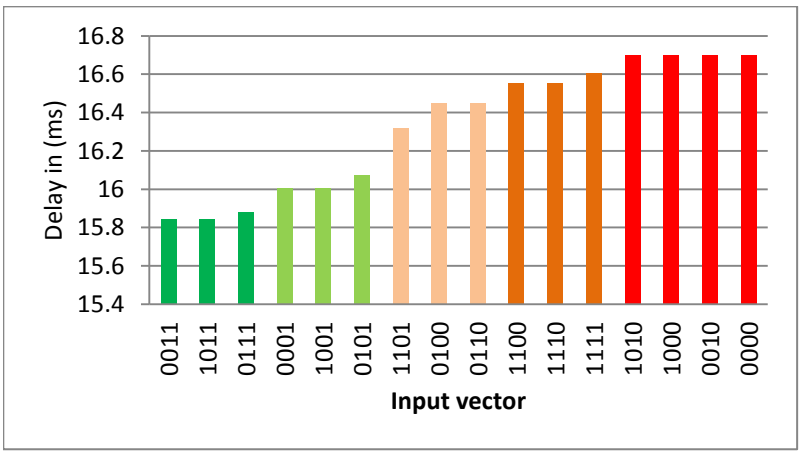
Fig.9.1 Nand based gate level implementation of FIR datapath on FPGA board



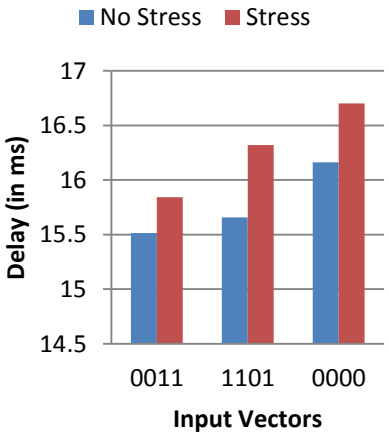
a



c



b



d

Fig. 9.2 Effect of NBTI stress on ARF Benchmark
(a) Change in threshold voltage with stress time,(b) Delay of the datapath corresponding to each input vector applied, (c) Stress Vs No-Stress for 0000, (d) Delay wrt Stress Vs. no-Stress

9.5. Results and analysis: Computational forensic engineering for resolving ownership conflict of DSP IP core generated using high level synthesis

The proposed approach and [13] were both implemented in java and run on Intel Core-i5-460M CPU with 3MB L3 cache memory; 4GB DDR3 memory at 2.5 GHz. The proposed approach containing 10 unique highly specialized design features in the 'feature set' (encompassing feature types of objectives, application type, data bit type, performance and datapath structure) have been investigated and tested on three major types of digital application specific IP cores. For example, benchmarks ARF, BPF & DCT are data intensive type application specific IPs; FFT & FIR are control intensive (loop based) type application specific IPs and JPEG IDCT is condition based data intensive type application specific IP cores [61]. Therefore the 'feature set' of the proposed approach is enough and applicable on all type of digital application specific IP cores. However, as mentioned in chapter 8, the proposed approach does not apply to IP cores of general purpose applications. It is only applicable for any type of application specific IP cores such as from signal processing and multimedia. The HLS tools selected for generating results for the proposed approach, are diverse in nature. For comprehensive analysis we have chosen seven academic/industrial tools (i.e. n = 7, from IPCT 1 to IPCT 7) with varying design objectives, varying DSE frameworks and varying properties as listed below:

1. Hybrid PSO-GA based HLS tool [52].
2. Fault tolerant based HLS tool [12].
3. Fault secure based HLS tool [17].
4. Watermarking based HLS tool [13].
5. Trojan security based HLS tool [53].
6. PSO based HLS tool [33].
7. BFOA based HLS tool [54].

Testing proposed CFE for ownership resolution for n = 7 is sufficient as the seven HLS tools are quite diverse and unique in nature. The same HLS tools are suitable for different IP cores as long as they are digital application specific IP by nature. Other HLS tools available in the literature mostly contain similar properties, frameworks or design objectives. Thus, addition of more HLS tools for testing may incur redundancy. However, the current seven HLS tools chosen for testing also comprises of HLS tools of similar characteristics. For example, HLS tool 1 (IPCT 1), HLS tool 5 (IPCT 5) and HLS tool 7 (IPCT 7) have several characteristics common in them. As shown in Table 9.13 for ARF benchmark, these three tools share eight common characteristics, but still the proposed approach was capable of identifying the legal owner successfully. Table 9.13 shows that HLS tool 5 (IPCT 5) has 100 % matching with given IP_{ID}. Additionally, as our results confirm that ten features in the feature set is sufficient to resolve IP ownership conflict for HLS tools. This is because all ten features in the set are unique though diverse and cover all the key aspects of HLS tools ranging from objectives (area, delay, power, Trojan security, fault reliability), application type (loop based/non-loop based), data bit type (data width), performance (scheduling type, chaining, multi-cycling, pipelining) and datapath structure (resource type used). Tables 9.13 to 9.16 shows the feature-set of proposed CFE approach generated with respect to each competing HLS tool for various benchmarks. The results indicate the matching percentage of feature-set of each competing HLS tools (corresponding to each IP vendor) with feature-set of IP_{ID}. The HLS tool whose IP feature-set matches 100 % with the feature-set of IP_{ID} is considered as the rightful owner. For example in table 9.13, the feature set decided for both IP to be identified (IP_{ID}) and competing IP tool vendors (IP_{CT} n) are: (scheduling algorithm, resource type, chaining, bus width support, pipelining, multi-cycling, design objective, reliability, Trojan security, loop support). The proposed feature extraction step determines the details of features for IPID & IPCT1… IPCT7. For ARF benchmark in table 9.13, for instance in IP_{ID}, the presence of ‘chaining’ feature was detected (thus denoted as ‘Yes’) and information of scheduling algorithm is indicated as ‘LIST’. Similarly, details of remaining features after extraction is also indicated. As evident the feature extraction of all IP core from each competing HLS tool is extracted. However,

feature set of only IP core generated by HLS tool 5 ($IP_{CT}5$) matches completely with IP_{ID}. Similarly, results for other benchmarks have been shown in table 9.13 to 9.16. Table 9.15 shows a case in which the IP_{ID}'s feature-set doesn't match with any of the feature-set of the competing IP tools i.e. matching percentage is less than 100%. Therefore, in such a scenario the legal ownership of the IP_{ID} cannot be awarded to any of the claimants. Further, there is possibility more than one competing IP vendor tool to have 100 % matching percentage. However, in our experiment we didn't encounter a scenario. Nevertheless, in such a scenario, further analysis through CFE is needed through addition of more features in the current feature-set (i.e. beyond the features in the current set). Table 9.17 shows the feature extraction time of each of the features of the feature set by proposed CFE approach. In other words, the features of the feature set are illustrated in increasing order of time complexity. This shows that the extraction time taken for 'loop support' is least, while for 'scheduling algorithm' is highest. Further, this also shows that all the ten features of the feature set are extracted within acceptable runtime (in order of only few milli-seconds).

Additionally, the possibility of false positive and false negative does not arise in the proposed results as the rightful IP owner is systematically determined through several digital forensic evidences acquired during/after high level synthesis design process. This is an inherent property of computational forensic engineering performed on high level synthesis based IP cores.

Table 9.18 shows the advantages of proposed CFE approach for IP core protection over watermarking based IP protection approach [13] in terms of storage overhead (i.e. number of registers required in final design). As evident from table 9.18, for watermarking approach [13], significant storage registers are required in final IP design. This is because signature insertion is done at the register allocation step of architectural synthesis. The presence of this signature is evaluated during signature detection stage for IP protection (by resolving false claim of vendor ownership). On the contrary, the proposed approach as shown in table 9.18 does not require embedding any vendor signature thus resulting into zero register during implementation (i.e. no design hardware overhead). The proposed CFE approach provides

greater/stronger reliability and protection as it is almost non-vulnerable to any threats due to no existence of reverse engineering step as well as vendor signature like in case of watermark based approaches.

Table 9.13. Feature-set of IP_{ID} and IP_{CT} for ARF benchmark
(Note: IP_{CT} n = IP core generated by competing HLS tool by vendor 'n')

Benchmark: ARF (28 nodes)											
IP features	**Schedule algorithm**	**Resource type**	**chaining**	**Bus width support**	**Data pipelining**	**Multi-cycling**	**Design objective**	**Fault Reliability**	**Loop support**	**Trojan Security**	**Match percent**
IP_{ID}	LIST	A, M	Yes	32 bit	No	Yes	Area - Execution time / Power-Execution time	No	DFG	No	NA
IP_{CT} 1 (Hybrid PSO-GA HLS [52])	LIST	A, M	Yes	16 bit	No	Yes	Area-Power-Latency	No	DFG	No	80
IP_{CT} 2 [Fault secure HLS [17])	LIST	A, M, C	No	16 bit	No	Yes	Area - Execution time / Power-Execution time	Yes	DFG	No	60
IP_{CT} 3 (Watermark-HLS [13])	LIST	A, M, C, S	No	32 bit	No	No	Area - Execution time / Power-Execution time	No	DFG	No	70
IP_{CT} 4 (Trojan Secure-HLS [53])	LIST	A, M, C, S	No	16 bit	No	No	Area - Execution time / Power-Execution time	No	DFG	Yes	50
IP_{CT} 5 (BFOA-HLS [54])	LIST	A, M	Yes	32 bit	No	Yes	Area - Execution time / Power-Execution time	No	DFG	No	100
IP_{CT} 6 (Fault Tolerant-HLS [12])	LIST	A, M, C, S	No	16 bit	No	No	Area - Latency	Yes	DFG	No	40
IP_{CT} 7 (PSO-HLS [33])	LIST	A, M	Yes	8 bit	Yes	Yes	Area - Execution time / Power-Execution time	No	DFG	No	80

Table 9.14. Feature-set of IP_{ID} and IP_{CT} for FFT benchmark

Benchmark: FFT (36 nodes)											
IP features	**Scheduling algorithm**	**Resource type**	**chaining**	**Bus width support**	**Data pipelining**	**Multi-cycling**	**Design objective**	**Fault Reliability**	**Loop support**	**Trojan Security**	**Match percent**
IP_{ID}	LIST	A, M, C	No	16 bit	No	Yes	Area - Execution time / Power- Execution time	Yes	Yes (CDFG & DFG)	No	NA
IP_{CT} 1 (Hybrid PSO-GA HLS [52])	LIST	A, M	Yes	16 bit	No	Yes	Area-Power-Latency	No	No (DFG ONLY)	No	50
IP_{CT} 2 [Fault secure HLS [17])	LIST	A, M, C	No	16 bit	No	Yes	Area - Execution time / Power- Execution time	Yes	Yes (CDFG & DFG)	No	100
IP_{CT} 3 (Watermark-HLS [13])	LIST	A, M, C, S	No	32 bit	No	No	Area - Execution time / Power- Execution time	No	Yes (CDFG & DFG)	No	60
IP_{CT} 4 (Trojan Secure-HLS [53])	LIST	A, M, C, S	No	16 bit	No	No	Area - Execution time / Power- Execution time	No	Yes (CDFG & DFG)	Yes	60
IP_{CT} 5 (BFOA-HLS [54])	LIST	A, M	Yes	32 bit	No	Yes	Area - Execution time / Power- Execution time	No	No (DFG ONLY)	No	50
IP_{CT} 6 (Fault Tolerant-HLS [12])	LIST	A, M, C, S	No	16 bit	No	No	Area - Latency	Yes	No (DFG ONLY)	No	60
IP_{CT} 7 (PSO-HLS [33])	LIST	A, M, C	Yes	8 bit	Yes	Yes	Area - Execution time / Power- Execution time	No	Yes (CDFG & DFG)	No	60

Table 9.15. feature-set of IP_{ID} and IP_{CT} for FIR benchmark

Benchmark: FIR (23 nodes)											
IP features	**Scheduling algorithm**	**Resource type**	**chaining**	**Bus width support**	**Data pipelining**	**Multi-cycling**	**Design objective**	**Reliability**	**Loop support**	**Trojan Security**	**Match percent**
IP_{ID}	LIST	A, M,C	Yes	8 bit	No	Yes	Area - Execution time / Power-Execution time	No	Yes (CDFG & DFG)	Yes	NA
IP_{CT} 1 (Hybrid PSO-GA HLS [52])	LIST	A, M	Yes	16 bit	No	Yes	Area-Power-Latency	No	No (DFG ONLY)	No	50
IP_{CT} 2 [Fault secure HLS [17])	LIST	A, M, C	No	16 bit	No	Yes	Area - Execution time / Power-Execution time	Yes	Yes (CDFG & DFG)	No	60
IP_{CT} 3 (Watermark-HLS [13])	LIST	A, M, C, S	No	32 bit	No	No	Area - Execution time / Power-Execution time	No	Yes (CDFG & DFG)	No	50
IP_{CT} 4 (Trojan Secure-HLS [53])	LIST	A, M, C, S	No	16 bit	No	No	Area - Execution time / Power-Execution time	No	Yes (CDFG & DFG)	Yes	60
IP_{CT} 5 (BFOA-HLS [54])	LIST	A, M	Yes	32 bit	No	Yes	Area - Execution time / Power-Execution time	No	No (DFG ONLY)	No	60
IP_{CT} 6 (Fault Tolerant-HLS [12])	LIST	A, M, C, S	No	16 bit	No	No	Area - Latency	Yes	No (DFG ONLY)	No	20
IP_{CT} 7 (PSO-HLS [33])	LIST	A, M, C	Yes	8 bit	No	Yes	Area - Execution time / Power-Execution time	No	Yes (CDFG & DFG)	No	90

Table 9.16. feature-set of IP_{ID} and IP_{CT} for JPEG_IDCT benchmark

Benchmark: JPEG_IDCT (112 nodes)											
IP features	**Schedule algorithm**	**Resource type**	**chaining**	**Bus width support**	**Data pipeline**	**Multi-cycling**	**Design objective**	**Reliability**	**Loop support**	**Trojan Security**	**Match percent**
IP_{ID}	LIST	A, M	Yes	8 bit	Yes	Yes	Area - Execution time / Power-Execution time	No	DFG	No	NA
IP_{CT} 1 (Hybrid PSO-GA HLS [52])	LIST	A, M	Yes	16 bit	No	Yes	Area-Power-Latency	No	DFG	No	70
IP_{CT} 2 [Fault secure HLS [17])	LIST	A, M, C	No	16 bit	No	Yes	Area - Execution time / Power-Execution time	Yes	DFG	No	50
IP_{CT} 3 (Watermark-HLS [13])	LIST	A, M, C, S	No	32 bit	No	No	Area - Execution time / Power-Execution time	No	DFG	No	50
IP_{CT} 4 (Trojan Secure-HLS [53])	LIST	A, M, C, S	No	16 bit	No	No	Area - Execution time / Power-Execution time	No	DFG	Yes	40
IP_{CT} 5 (BFOA-HLS [54])	LIST	A, M	Yes	32 bit	No	Yes	Area - Execution time / Power-Execution time	No	DFG	No	80
IP_{CT} 6 (Fault Tolerant-HLS [12])	LIST	A, M, C, S	No	16 bit	No	No	Area - Latency	Yes	DFG	No	30
IP_{CT} 7 (PSO-HLS [33])	LIST	A, M	Yes	8 bit	Yes	Yes	Area - Execution time / Power-Execution time	No	DFG	No	100

Table 9.17. Average time consumed (ms) for feature extraction through proposed CFE approach

Benchmarks	Loop support	Design objective	Resource type	Bus width support	Multi-cycling	Fault Reliability	Trojan Security	chaining	Data pipelining	Scheduling algorithm
ARF	0.3	1.2	3.1	7.2	23.5	46.3	48.7	80.5	74.6	374.5
BPF	0.7	1.5	4.9	9.3	19.1	52.2	51.3	70.2	54.8	256.7
DCT	0.8	2.4	5.7	12.8	19.6	49.8	57.8	68.7	88.5	231.1
FFT	0.9	2.8	4.7	10.3	28.5	68.1	52.0	89.5	88.8	407.0
FIR	0.6	4.7	5.9	10.7	13.6	35.9	72.9	76.8	69.2	240.1
JPEG_IDCT	1.3	10.9	18.3	48.7	89.5	153.3	203.7	283.8	452.3	1903.0

Table 9.18. Advantages of proposed CFE approach over watermarking [13] for IP protection during HLS

Benchmark	Watermarking IP protection HLS approach [13]	Proposed CFE based IP protection HLS approach
	Storage registers	Storage registers
ARF	11	0
BPF	11	0
DCT	11	0
FFT	10	0
FIR	11	0
JPEG_IDCT	25	0

Chapter 10

Conclusion and Future work

10.1. Conclusion

This thesis has presented novel methodologies for generating reliable and secure IP cores. The following objectives were accomplished

- Proposed a methodology that integrates 'high level synthesis' framework with 'physical design' framework for generating a DSP IP core that is simultaneously secure/resilient against multi-cycle temporal and multi-unit spatial effect of transient fault. The transient fault resiliency is achieved with a nominal design overhead.
- Proposed a methodology for generating a DSP IP core that is *simultaneously tolerant* against multi-cycle temporal and multi-unit spatial effect of transient fault for *data intensive applications*. The proposed approach is the first technique in the literature that considers simultaneous tolerance against temporal and spatial effect of single event transient. The proposed approach presents novel transient fault tolerance-aware floor-planning rules. Further, it integrates PSO-DSE framework for exploring low-cost design solution.
- Proposed a methodology for generating a DSP IP core that is *simultaneously tolerant* against multi-cycle temporal and multi-unit spatial effect of transient fault for *control intensive applications*. The proposed approach achieves a design cost improvement of ~27% along with power reduction of ~61% compared to the state-of-the-art.
- Proposed a methodology for generating a low-cost, highly secure, functionally obfuscated DSP IP core. The proposed methodology presents a novel IP functional locking block termed as ILB. The proposed ILBs inherits security properties that enhances strength of obfuscation of the IP cores. Further, Security comparison of proposed approach with state-of-the-art, shows a minimum security enhancement of 4.29 e+9 times for the tested benchmarks.

- Proposed a methodology for analyzing the aging effect of NBTI stress on performance of DSP IP core. It presents performance comparison of stressed v/s not-stressed states of IP cores. Further, it presents a technique to identify input vector that causes maximum performance degradation due to NBTI stress on DSP IP core. The proposed approach can be utilized to detect the presence of accelerated aging attack on IP core.
- Proposed a novel computational forensic engineering methodology for resolving ownership conflict of DSP IP core generated using high level synthesis. The proposed approach presents a set of ten novel features that can distinguish an IP core from another IP core generated using different high level synthesis tools. Further, the proposed approach presents feature extraction rules/algorithms for each of the ten features of the feature-set. The comparison of proposed approach with state-of-the-art (watermarking based) approach for resolving ownership conflicts shows that the proposed approach incurs zero-overhead and zero-performance degradation.

10.2. Future work

In future, various reliability aware methodologies for resolving reliability concerns such as electromigration, intermittent faults, etc. can be devised for DSP cores using high level synthesis framework. In a similar manner, low energy/power security aware methodologies can be devised for ensuring protection against hardware Trojan, IP piracy, IP overbuilding, etc. using high level synthesis.

REFERENCES

[1] Mack, C. A. (2011). Fifty years of Moore's law. *IEEE Transactions on semiconductor manufacturing*, *24*(2), 202-207.

[2] Consumer Technology Association (CTA) report 2018 https://lsc-pagepro.mydigitalpublication.com/publication/?i=495372&ver=html5

[3] Sengupta, A. (2016). Evolution of the IP Design Process in the Semiconductor/EDA Industry [Hardware Matters]. *IEEE Consumer Electronics Magazine*, *5*(2), 123-126.

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

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

[6] Sengupta, A. (2016). Soft IP Core Design Resiliency Against Terrestrial Transient Faults for CE Products [Hardware Matters]. *IEEE Consumer Electronics Magazine*, *5*(4), 129-131.

[7] Gajski, D. D., Wu, A. C. H., *et al* (2000). Embedded tutorial: essential issues for IP reuse. In *Proceedings of the 2000 Asia and South Pacific Design Automation Conference* (pp. 37-42). ACM.

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

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

[10] Omana, M., Papasso, G., Rossi, D., & Metra, C. (2003, July). A model for transient fault propagation in combinatorial logic. In *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*(pp. 111-115). IEEE.

[11] Andjelkovic, M., Krstic, M., Kraemer, R., Veeravalli, V. S., & Steininger, A. (2017, November). A Critical Charge Model for Estimating the SET and SEU Sensitivity: A Muller C-Element Case

Study. In *Asian Test Symposium (ATS), 2017 IEEE 26th*(pp. 82-87). IEEE.

[12] Inoue, T., Henmi, H., Yoshikawa, Y., & Ichihara, H. (2011, July). High-level synthesis for multi-cycle transient fault tolerant datapaths. In *On-Line Testing Symposium (IOLTS), 2011 IEEE 17th International* (pp. 13-18). IEEE.

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

[14] Amrouch, H., Krishnamurthy, P., Patel, N., et al (2017, October). Emerging (un-) reliability based security threats and mitigations for embedded systems: special session. In *Proceedings of the 2017 International Conference on Compilers, Architectures and Synthesis for Embedded Systems Companion* (p. 17). ACM.

[15] Sinanoglu, O., Karimi, N., Rajendran, J., Karri, R., Jin, Y., Huang, K., & Makris, Y. (2013, May). Reconciling the IC test and security dichotomy. In *Test Symposium (ETS), 2013 18th IEEE European* (pp. 1-6). IEEE.

[16] Wu, K., & Karri, R. (2001, November). Algorithm level re-computing: a register transfer level concurrent error detection technique. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design* (pp. 537-543). IEEE Press.

[17] Sengupta, A., & Sedaghat, R. (2015). Swarm intelligence driven design space exploration of optimal k-cycle transient fault secured datapath during high level synthesis based on user power–delay budget. *Microelectronics Reliability*, *55*(6), 990-1004.

[18] Wu, K., & Karri, R. (2004). Fault secure datapath synthesis using hybrid time and hardware redundancy. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *23*(10), 1476-1485.

[19] Rusu, C., Bougerol, A., Anghel, L., Weulerse, C., Buard, N., Benhammadi, S., ... & Gaillard, R. (2007, July). Multiple event transient induced by nuclear reactions in CMOS logic cells. In *On-Line Testing Symposium, 2007. IOLTS 07. 13th IEEE International* (pp. 137-145). IEEE.

[20] Miskov-Zivanov, N., & Marculescu, D. (2010). Multiple transient faults in combinational and sequential circuits: A systematic approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *29*(10), 1614-1627.

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

[22] Rajendran, J., Pino, Y., Sinanoglu, O., & Karri, R. (2012, June). Security analysis of logic obfuscation. In *Proceedings of the 49th Annual Design Automation Conference* (pp. 83-89). ACM.

[23] Roy, D., & Sengupta, A. (2017). Low overhead symmetrical protection of reusable IP core using robust fingerprinting and watermarking during high level synthesis. *Future Generation Computer Systems*, *71*, 89-101.

[24] Lisboa, C. A., Erigson, M. I., & Carro, L. (2007, May). System level approaches for mitigation of long duration transient faults in future technologies. In *Test Symposium, 2007. ETS'07. 12th IEEE European* (pp. 165-172). IEEE.

[25] Heijmen, T. (1994). Radiation-induced soft errors in digital circuits. *energy*, *7*, 19.

[26] Rossi, D., Omana, M., Toma, F., & Metra, C. (2005, October). Multiple transient faults in logic: An issue for next generation ICs?. In *Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on* (pp. 352-360). IEEE.

[27] Martin, R. C., Ghoniem, N. M., Song, Y., & Cable, J. S. (1987). The size effect of ion charge tracks on single event multiple-bit upset. *IEEE Transactions on Nuclear Science*, *34*(6), 1305-1309.

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

[29] Dubrova, E. (2013). *Fault-tolerant design* (pp. 55-65). New York: Springer.

[30] Kshirsagar, R. V., & Patrikar, R. M. (2009). Design of a novel fault-tolerant voter circuit for TMR implementation to improve reliability in digital circuits. *Microelectronics Reliability*, *49*(12), 1573-1577.

[31] Martins, M., Matos, J. M., Ribas, R. P., et al (2015, March). Open cell library in 15nm FreePDK technology. In *Proceedings of the 2015 Symposium on International Symposium on Physical Design*(pp. 171-178). ACM.

[32] Sengupta, A., & Mishra, V. K. (2014). Automated exploration of datapath and unrolling factor during power–performance tradeoff in architectural synthesis using multi-dimensional PSO algorithm. *Expert Systems with Applications*, *41*(10), 4691-4703.

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

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

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

[36] Koushanfar, F. (2011, May). Integrated circuits metering for piracy protection and digital rights management: An overview. In *Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI* (pp. 449-454). ACM.

[37] Alkabani, Y., Koushanfar, F., & Potkonjak, M. (2007, November). Remote activation of ICs for piracy prevention and digital right management. In *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design* (pp. 674-677). IEEE Press.

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

[39] Mahapatra, S., Goel, N., Desai, S., Gupta, S., Jose, B., Mukhopadhyay, S., ... & Alam, M. A. (2013). A comparative study of different physics-based NBTI models. *IEEE Transactions on Electron Devices*, *60*(3), 901-916.

[40] Grasser, T., Rott, K., Reisinger, H., Waltl, M., Schanovsky, F., & Kaczer, B. (2014). NBTI in nanoscale MOSFETs—The ultimate modeling benchmark. *IEEE Transactions on Electron Devices*, *61*(11), 3586-3593.

[41] Mahapatra, S., Huard, V., Kerber, A., Reddy, V., Kalpat, S., & Haggag, A. (2014, June). Universality of NBTI-From devices to circuits and products. In *Reliability Physics Symposium, 2014 IEEE International* (pp. 3B-1). IEEE.

[42] Gös, W. (2011). *Hole trapping and the negative bias temperature instability*. (PhD Thesis), Technische universität wien, Austria, December 2011

[43] Wang, Y., Chen, X., Wang, W., Balakrishnan, V., Cao, Y., Xie, Y., & Yang, H. (2009, March). On the efficacy of input vector control to mitigate NBTI effects and leakage power. In *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design* (pp. 19-26). IEEE.

[44] Firouzi, F., Kiamehr, S., & Tahoori, M. B. (2011, May). A linear programming approach for minimum NBTI vector selection. In *Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI* (pp. 253-258). ACM.

[45] Gonzalez, R., Gordon, B. M., & Horowitz, M. A. (1997). Supply and threshold voltage scaling for low power CMOS. *IEEE Journal of Solid-State Circuits*, *32*(8), 1210-1216.

[46] Sengupta, A., Sedaghat, R., & Zeng, Z. (2010). 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.

[47] Reece, T., & Robinson, W. H. (2016). Detection of hardware trojans in third-party intellectual property using untrusted modules. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *35*(3), 357-366.

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

[49] Roy, D., & Sengupta, A. (2017). Low overhead symmetrical protection of reusable IP core using robust fingerprinting and watermarking during high level synthesis. *Future Generation Computer Systems*, *71*, 89-101.

[50] Wong, J. L., Kirovski, D., & Potkonjak, M. (2004). Computational forensic techniques for intellectual property protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *23*(6), 987-994.

[51] Franke, K., & Srihari, S. N. (2008, August). Computational forensics: An overview. In *International Workshop on Computational Forensics* (pp. 1-10). Springer, Berlin, Heidelberg.

[52] Ram, D. H., Bhuvaneswari, M. C., & Logesh, S. M. (2011, July). A novel evolutionary technique for multi-objective power, area and delay optimization in high level synthesis of datapaths. In *A Novel Evolutionary Technique for Multi-objective Power, Area and Delay Optimization in High Level Synthesis of Datapaths*. IEEE.

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

[54] Sengupta, A., & Bhadauria, S. (2015). Bacterial foraging driven exploration of multi cycle fault tolerant datapath based on power-performance tradeoff in high level synthesis. *Expert Systems with Applications*, *42*(10), 4719-4732.

[55] Coussy, P., Gajski, D. D., Meredith, M., & Takach, A. (2009). An introduction to high-level synthesis. *IEEE Design & Test of Computers*, *26*(4), 8-17.

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

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

[58] Yu, S. Y., & McCluskey, E. J. (2001). Permanent fault repair for FPGAs with limited redundant area. In *Defect and Fault Tolerance in VLSI Systems, 2001. Proceedings. 2001 IEEE International Symposium on* (pp. 125-133). IEEE.

[59] Constantinescu, C. (2008, January). Intermittent faults and effects on reliability of integrated circuits. In *Reliability and Maintainability Symposium, 2008. RAMS 2008. Annual* (pp. 370-374). IEEE.

[60] Gomaa, M. A., & Vijaykumar, T. N. (2005, June). Opportunistic transient-fault detection. In *Computer Architecture, 2005. ISCA'05. Proceedings. 32nd International Symposium on* (pp. 172-183). IEEE.

[61] Express benchmark suite, University of California San Diego, 2016, https://www.ece.ucsb.edu/EXPRESS/benchmark/

[62] Baumann, R. (2013, July). Landmarks in terrestrial single event effects. In *IEEE NSREC Short Course*.

[63] Li, X. (2005). Tolerating Radiation-Induced Transient Faults in Modern Processors DISSERTATION. In *PhD Diss*. University of California Irvine.

[64] Gaillard, R. (2011). Single event effects: Mechanisms and classification. In *Soft Errors in Modern Electronic Systems* (pp. 27-54). Springer, Boston, MA.

[65] Wang, W., Wei, Z., Yang, S., & Cao, Y. (2007, November). An efficient method to identify critical gates under circuit aging. In *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design* (pp. 735-740). IEEE Press.

[66] Violante, M., Meinhardt, C., Reis, R., & Reorda, M. S. (2011). A low-cost solution for deploying processor cores in harsh environments. *IEEE Transactions on Industrial Electronics*, *58*(7), 2617-2626.

[67] Sarkar, S., & Shinde, S. (2005, September). Effective IP reuse for high quality SOC design. In *SOC Conference, 2005. Proceedings. IEEE International* (pp. 217-224). IEEE.

[68] Xie, Y., & Hung, W. L. (2006). Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (MPSoC) design. *Journal of VLSI signal processing systems for signal, image and video technology*, *45*(3), 177-189.

[69] Saleh, R., Wilton, S., Mirabbasi, S., Hu, A., Greenstreet, M., Lemieux, G., ... & Ivanov, A. (2006). System-on-chip: Reuse and integration. *Proceedings of the IEEE*, *94*(6), 1050-1069.

[70] Sait, S. M., & Youssef, H. (1999). *VLSI physical design automation: theory and practice* (Vol. 6). World Scientific Publishing Company.