LOW COST FAULT RELIABILITY AND TROJAN SECURITY AWARE HIGH LEVEL SYNTHESIS FOR APPLICATION SPECIFIC DATAPATH PROCESSORS

Ph.D. Thesis

By

SAUMYA BHADAURIA

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JANUARY 2016

LOW COST FAULT RELIABILITY AND TROJAN SECURITY AWARE HIGH LEVEL SYNTHESIS FOR APPLICATION SPECIFIC DATAPATH PROCESSORS

A THESIS

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

by

SAUMYA BHADAURIA

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
JANUARY 2016

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **LOW COST FAULT RELIABILITY AND TROJAN SECURITY AWARE HIGH LEVEL SYNTHESIS FOR APPLICATION SPECIFIC DATAPATH PROCESSORS** in the partial fulfillment of the requirements for the award of the degree of **DOCTOR OF PHILOSOPHY** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING, INDIAN INSTITUTE OF TECHNOLOGY INDORE**, is an authentic record of my own work carried out during the time period from July 2013 to January 2016 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
SAUMYA BHADAURIA

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

(Dr. Anirban Sengupta)

SAUMYA BHADAURIA has successfully given his/her Ph.D. Oral Examination held on

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

Convener, DPGC
Date:

Signature of PSPC Member #1
Date:

Signature of PSPC Member #1
Date:

Signature of External Examiner
Date:

ACKNOWLEDGEMENTS

First of all, I would like to thank my parents and my brother, Ishan, for their strong belief in me and for always being a constant source of motivation and guidance. This thesis would not have been possible without them.

I would like to express my gratitude to my supervisor Dr. Anirban Sengupta for providing me the opportunity to do work under his supervision, without his guidance and directions to solve the challenging tasks this research work could not be completed. I have learned a lot from him.

Besides my advisor, I also owe a mention to Dr. Abhishek Srivastava, Dr. Surya Prakash and Dr. Shanmugam Dhinakaran for their valuable feedbacks on my research work throughout these years. I am grateful to Dr. Kapil Ahuja, Head, CSE Discipline for all the help and cooperation.

Special thanks to Navneet, Rajendra and Vipul for their moral support and valuable opinions at times. It has been a great pleasure to work with many talented colleagues during my years of work at IIT Indore. I would like to thank all my past and present colleagues and all my friends for their help and support.

At last, I would like to thank IIT Indore, DST, CSIR to help me financially and providing me an opportunity to showcase my research at international grounds.

DEDICATED TO MY FAMILY

ABSTRACT

With changing trends in technology and to effectively compete in the market, designers are focussing on attempts to optimize Very Large Scale Integration (VLSI) digital systems. Attempts to devise design systems with higher performance, accuracy and efficiency along with lower overall cost are being made. In order to achieve this, High Level Synthesis (HLS) / architectural synthesis has come into force. However, there is a paradigm shift in the area of HLS as more and more designs are suffering from reliability and hardware security issues. These are expected to become the key focus due to massive scaling in nanometre technology and globalization involved in the VLSI design process. This thesis proposes methodologies for generating low cost security solutions for both transient fault and hardware Trojan with respect to data intensive and control intensive applications during design of application specific datapath processor at behavioural level. This thesis solves five different types of problems in generating reliable//hardware secured designs: **a)** Problem of design Space Exploration (DSE) during power-performance trade-off for data intensive applications that produces high quality design solutions. In addition, a novel Bacterial Foraging Optimization (BFO) driven DSE methodology is proposed which explores the design points in the design space. A novel chemotaxis, replication and elimination-dispersal algorithm is proposed which generates the design points. **b)** Problem of exploration of low cost optimal *k*-cycle transient fault secured datapath during power-performance trade-off for data intensive applications. A novel fault security algorithm for handling single and multi-cycle transient faults is proposed. A novel multi-cycle Single Event Transient (SET) fault security aware multi objective DSE methodology that explores an optimal combination of transient fault secured (Double Modular Redundant) DMR datapath configuration has been proposed. Moreover, a novel scheme for selecting appropriate edges for inserting cuts in the scheduled Data Flow Graphs (DFG) minimizing delay overhead associated with transient fault security, a novel execution time model for estimating the execution time of a transient fault secured/Trojan secured design during DSE process, a novel fitness function, used for design quality assessment in DSE process has been proposed. **c)** Problem of exploration of low cost optimal *k*-cycle transient fault secured datapath during area-delay trade-off for control intensive applications. a novel multi-cycle SET fault security aware multi objective DSE methodology that explores

an optimal combination of transient fault secured DMR datapath configuration and loop Unrolling Factor (UF) for Control Data Flow Graphs (CDFG) has been proposed. Moreover, a novel estimation model for computation of execution delay of a loop unrolled CDFG (based on a resource configuration explored) without tediously unrolling the entire CDFG for the specified loop value has been proposed. **d)** Problem of exploration of low cost optimal *k*-cycle transient fault tolerant datapath based on power-performance tradeoff for data intensive applications. In relation to this, a novel multi-cycle transient fault tolerant algorithm that has capability to isolate original and duplicate units in a DMR with respect to the transient fault has been proposed. Moreover, a novel equivalent circuit that works with DMR systems performs the function of extracting the correct output from the DMR design has been proposed. **e)** Problem of exploration of low cost optimal Trojan secured datapath during behavioural synthesis for data intensive applications has been tackled. A novel encoding scheme for representing bacterium in the design space (comprising of candidate datapath resource configuration and vendor allocation information for hardware Trojan secured datapath) has been proposed. Moreover, a novel exploration process of an efficient vendor allocation procedure that assists in yielding a low cost hardware Trojan secured datapath within user constraints has been proposed.

LIST OF PUBLICATIONS

International Journals (5)

1. Saumya Bhadauria, Anirban Sengupta "Adaptive bacterial foraging optimization driven Design Space Exploration: Exploring area-performance tradeoff during HLS", ***Elsevier Journal on Applied Mathematics and Computations***, Volume 269, pp. 265–278, Oct 2015. ***(5 yr Impact Factor ~1.686)***

2. Anirban Sengupta, Saumya Bhadauria, "Automated Design Space Exploration of Multi-Cycle Transient Fault Detectable Datapath based on Multi-Objective User Constraints for Application Specific Computing", ***Elsevier Journal on Advances in Engineering Software***, Volume 82, pp. 14- 24, April 2015. ***(5 yr Impact Factor ~1.5)***

3. Anirban Sengupta, Saumya Bhadauria, "Bacterial Foraging Driven Exploration of Multi Cycle Fault Tolerant Datapath based on Power-Performance Tradeoff in High Level Synthesis", ***Elsevier Journal on Expert Systems With Applications***, Volume 42, pp. 4719 - 4732, Jan 2015. ***(5yr Impact Factor = 2.339)***.

4. Saumya Bhadauria, Anirban Sengupta, "Multi-Cycle Single Event Transient Fault Security Aware MO-DSE for Single loop CDFGs in HLS", ***IEEE VLSI Circuits & Systems Letters***, Volume 1, Issue 2, Oct 2015.

5. Anirban Sengupta, Saumya Bhadauria, "Exploration of Multi-Objective Tradeoff During High Level Synthesis Using Bacterial Chemotaxis and Dispersal", ***Elsevier Journal on Procedia Computer Science***, Volume 35, Issue C, pp. 63 -72, Sep 2014.

Peer Reviewed Conferences (7)

6. Anirban Sengupta**,** Saumya Bhadauria, "Untrusted Third Party Digital IP cores: Power-Delay Trade-off Driven Exploration of Hardware Trojan Secured Datapath during High Level Synthesis", Proceedings of ***25th IEEE/ACM Great Lake Symposium on VLSI (GLSVLSI)***, *Pennsylvania*, pp. 167-172, May 2015. *(DOUBLE BLIND REVIEW).*

7. Anirban Sengupta, Saumya Bhadauria, "Automated Design Space Exploration of Transient Fault Detectable Datapath Based on User Specified Power and Delay Constraints", Proceedings of ***33rd VLSI - Design Automation & Test (VLSI - DAT),*** *Taiwan*, pp. 1-4, April 2015. (DOUBLE BLIND REVIEW) *Note- Amongst top 10 EDA/VLSI conferences*

8. Anirban Sengupta, Saumya Bhadauria, "User Power-Delay Budget Driven PSO Based Design Space Exploration of Optimal *k*-cycle Transient Fault Secured Datapath during High Level Synthesis", Proceedings of ***16th IEEE International Symposium on Quality Electronic Design (ISQED 2015),*** California, USA, pp. 289 - 292, March 2015. (DOUBLE BLIND REVIEW) *Note- Amongst top 10 EDA/VLSI conferences*

9. Saumya Bhadauria, Anirban Sengupta, "A High Level Synthesis Approach for Exploring Low Cost *k*-cycle Transient Fault Secured Solution", ***21st Asia South Pacific-Design Automation Conference (ASP-DAC),*** Jan 2016, Accepted.

10. Anirban Sengupta, Saumya Bhadauria, "Automated Exploration of Datapath in High Level Synthesis using Temperature Dependent Bacterial Foraging Optimization Algorithm", Proceedings of ***27th IEEE Canadian Conference on Electrical and Computer Engineering***, ***Toronto***, pp. 68 -73, May 2014.

11. Anirban Sengupta, Saumya Bhadauria, "Error Masking of Transient Faults: Exploration of a Fault Tolerant Datapath Based on User Specified Power and Delay Budget", Proceedings of ***13th IEEE International Conference on Information Technology***, pp. 345 – 350, Dec 2014. (DOUBLE BLIND REVIEW)

12. Saumya Bhadauria, Anirban Sengupta, "Secure Information Processing during System level: Exploration of an Optimized Trojan Secured Datapath for CDFGs during HLS based on User Constraints", Proceedings of **1st** ***IEEE iNIS 2015*** Special Session, Accepted, 2015.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

p	Population Size
D	Total number of resource types
R_x	Candidate resource combination for optimal solution
X_i	Resource combination
X_i^{Last}	Last resource set of i^{th} bacterium solution
X_i^{New}	New configuration of the particle/bacterium
P_T	Power consumed by a resource configuration
P_S	Static power consumed by a resource configuration
P_D	Dynamic power consumed by a resource configuration
P_{cons}	Power constraints specified by the user
$P_{\max}$	Maximum power consumed by a resource configuration in design space
P_{min}	Minimum power consumed by a resource configuration in design space
P_T^{DMR}	Power consumed by a fault secured DMR system
P_S^{DMR}	Static power consumed by a fault secured DMR system
P_D^{DMR}	Dynamic power consumed by a fault secured DMR system
$P_{\max}^{DMR}$	Maximum power consumed by a fault secured DMR system
$P_{\min}^{DMR}$	Minimum power consumed by a fault secured DMR system
p_c	Power dissipated per area unit (e.g. transistors)
A_{cons}	Area constraints specified by the user
A_T^{DMR}	Area consumed by a fault secured DMR system
$A_{\max}^{DMR}$	Maximum area consumed by a fault secured DMR system
L	Latency of a resource configuration
T_c	Initiation interval or cycle time of a resource configuration
T_E	Execution time consumed by a resource configuration
T_{cons}	Execution time constraints specified by the user
$T_{\max}$	Maximum execution time consumed by a resource configuration in design space
T_{min}	Minimum execution time consumed by a resource configuration in design space
T_E^{DMR}	Execution time consumed by a fault secured DMR system

$T_{\max}^{DMR}$	Maximum execution time consumed by a fault secured DMR system
$T_{\min}^{DMR}$	Minimum execution time consumed by a fault secured DMR system
$K(R_d)$	Represents the area occupied by resource R_d
$N(R_d)$	number of instances of resource type R_d
$N(R_d)^{New}$	New value of the d^{th} resource type
$N(R_d)^{\max}$	Maximum value of d^{th} resource type
$N(R_d)^{\min}$	Minimum value of d^{th} resource type
$t_{\min}$	Minimum temperature
$t_{\max}$	Maximum temperature
$Temp$	Initial temperature of exploration process
Δt	T_w / N_c
T_w	Temperature window
j	Current iteration step/chemotactic step
k	Replication steps counter
l	Elimination dispersal step counter
N_c	Maximum number of chemotactic steps
N_{re}	Maximum number of replication steps
N_{ed}	Maximum number of elimination dispersal steps
$\Delta(i)$	Tumble Vector
$C(i)$	Step size
$C(i)^{New}$	New step size generated
$C(i)^{Last}$	Last step size used
Rep [j–])	Array to check whether replication performed or not
Ed [j–]))	Array to check whether elimination dispersal performed or not
x	Step # at which replication has to be performed
y	Step # at which elimination dispersal has to be performed
$C_f(X_i)$	Cost of bacterium with resource set X_i fitness of particle/bacterium
$C_{f_{lbi}}(X_i)$	Local best fitness of particle 'X_i'
E_{FU}	Energy consumed by the major FU's
E_{mux} and E_{demux}	Energy consumed by the multiplexers and de-multiplexers
ϕ_1 and ϕ_2	User defined weights
k_c	strength of the fault

Z	Termination criteria
U^{OG}	Original unit
U^{DP}	Duplicate unit
c.s	Control steps
X_{gb}	Global best position
X_{lbi}	Local best position for an i^{th} particle
P_m	Mutation Probability
$R_{d_i}^{+}$	New resource value of particle X_i in d^{th} dimension
R_{d_i}	Previous resource value of particle X_i in d^{th} dimension
$R_{d_{lbi}}$	Resource value of X_{lbi} in d^{th} dimension
$R_{d_{gb}}$	Resource value of X_{gb} in d^{th} dimension
$V_{d_i}^{+}$	New velocity of particle X_i in d^{th} dimension
UF	Unrolling factor
UF_N	N^{th} unrolling factor
C_{body}^{DMR}	Number of CS required to execute loop body of $CDFG^{DMR}$
C_{first}^{DMR}	Number of CS required to execute first iteration of the $CDFG^{DMR}$
I	Maximum number of iteration (loop count)
Δ	Delay of one CS in nanoseconds
£	Number of iteration
D_c	Dependency information
$D(opn)$	Delay of operation 'n'
A_v	Vendor allocation procedure type

ACRONYMS

IC	Integrated Circuits
VLSI	Very Large Scale Integration
HLS	High Level Synthesis
HDL	Hardware Description Languages
DFG	Data Flow Graph
RTL	Register Transfer Level
CDFG	Control Data Flow Graph
DSE	Design Space Exploration
ASAP	As Soon As Possible
ALAP	As Late As Possible
FU	Functional Units
GA	Genetic Algorithm
WSPSO	Weighted Sum Particle Swarm Optimization
CED	Concurrent Error Detection
TMR	Triple Modular Redundant
SET	Single Event Transient
SoC	System-On-Chip
IP	Intellectual Property
3PIP	Third Party Intellectual Property
DMR	Double Modular Redundant
BFOA	Bacterial Foraging Optimization Algorithm
QoR	Quality Of Result
ED	Elimination-Dispersal
PSO	Particle Swarm Optimization
SEU	Single Event Upset
SDFG	Scheduled Data Flow Graph
EA	Evolutionary Algorithm

TFH	Transient Fault Hazards
LET	Linear Energy Transfer
MCFT	Multi Cycle Fault Tolerance
VHDL	VHSIC Hardware Description Language
FPGA	Field Programmable Gate Array
ARF	Auto Regressive Filter
BPF	Band Pass Filter
DCT	Discrete Cosine Transformation
IDCT	Inverse Discrete Cosine Transformation
DWT	Discrete Wavelet Transform
FIR	Finite Impulse Response
WDF	Wave Digital Filter
EWF	Elliptic Wave Filter
FFT	Fast Fourier Transformation
MO-DSE	Multi Objective Design Space Exploration

Chapter 1

Introduction

1.1 Preamble

With the explosion of technology, the 20th century era witnessed a drastic change in the lifestyle. The key inventions on Integrated Circuits (ICs) have led to high speed microprocessors and memories. With the advent of such breakthroughs, there have been equally important developments which have brought steady growth in digital systems. In early 60s, *Moore*, predicted the exponential growth of the number of transistors on an integrated circuit. This in turn provided higher functionalities within a single unit at low cost, leading to higher complexity while designing and verification.

As the complexity of systems increases, there arises need for automation at higher abstraction levels where functionalities and tradeoffs are easier to understand. Automation assures a shorter design cycle. Also, there is a greater possibility of quickly exploring different and better designs. Raising the design abstraction to behavioural level or architectural level boosts the design productivity [1, 60, 61, 62]. An architectural level specification describes the algorithm to be implemented, without the details of the structure of the circuit.

1.2 Circuit Design and Synthesis

The Very Large Scale Integration (VLSI) design flow consists of a number of design and test levels to match the design specifications. The design engineer accepts the user requirements, and translates them into specifications. Once the specifications are determined, the designing is performed. The process includes system level, high level, gate or logic level, transistor or circuit level and physical or layout level. The levels can be described as [1]:

- *System level*: This is the highest level of abstraction, where the system is represented as processes, tasks, hardware and software. This level deals with the overall system and the information flow within the system.

- *Behavioural or Algorithmic or High level*: This level controls the computation by individual processors within the system. It monitors the mapping sequences of inputs to the outputs.
- *Register Transfer Level (RTL)*: At this level, the system is specified as a set of storage elements and functional units.
- *Logic or Gate level*: At the logic level system is viewed as a network of gates and flip-flops. The behaviour of the system is specified in terms of logic equations.
- *Circuit or Transistor level*: this level describes the circuits as a netlist of transistors. The issues related to the nature and numbers of transistors to be used are dealt at the circuit level.
- *Physical or Layout level*: This is the lowest level of circuit abstraction in which the system is specified in terms of individual transistors.

The design process proceeds from higher to lower abstraction levels. The automated process of designing the VLSI circuits is referred as synthesis. Specifying the design at a higher abstraction level has been an effective way to deal with the complexity.

1.3 High Level Synthesis (HLS) Details

With the increasing design complexity of ICs the idea of automatically generating circuit implementations from high-level behavioural specifications has gained interest. Initially, multiple prototype tools were developed to call attention to the methodology and to experiment with various algorithms. In late 80s and early 90s, a number of similar HLS tools were built, mostly for research and prototyping. MIMOLA [2], ADAM [3, 4], HAL [5], Hercules/Hebe [7, 8], and Hyper/Hyper-LP [6, 9] were some academic efforts. These tools decompose the synthesis task into following steps:

a) Code transformation,
b) Module selection,
c) Operation scheduling,
d) Datapath allocation, and
e) Controller generation.

These problems were individually addressed later using algorithms like list scheduling algorithm, force-directed scheduling algorithm and many others. This provided a base for HLS. However, these efforts were not enough for wide acceptance of HLS among designers due to low quality solutions generated.

```
r = a+b;
s =c*d;
t = e*f;
u = r+s;
v = u +t
```

Figure 1.1 Sample Behavioural description

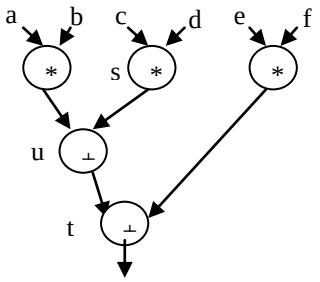


Figure 1.2 Sample Data Flow Graph

In 1995 several tools like Behavioural Compiler [11] from Synopsis, Monet from Mentor Graphics [10] and Visual Architect from Cadence [12] were introduced which received a wide attention. However, the tools were not widely accepted since these tools used Hardware Description Languages (HDL) such as Very High Speed Integrated Circuit Hardware Description Language (VHDL) or Verilog for behavioural description as input. Since then a wide range of tools are developed which are commercially accepted and use C/C++ or C-based languages to capture the design.

1.4 Theoretical Background on HLS

HLS is a process of transforming a software behavioural description into a hardware circuit description with equivalent functionality [60-63]. It is sometimes referred as behavioural or architectural or C-to-gates synthesis. The behavioural description describes the input and output behaviour of the algorithm in terms of operations and data transfers. It consists of algorithmic statements containing the different operations viz. additions, multiplications, logical operations and control operations like loops, conditional statements and function calls. The behavioural description is represented in the form of a Data Flow graph (DFG). The DFG comprises of operations in the algorithmic description and the data dependencies between them are represented by the vertices and edges, respectively. Figure 1.2 gives an example of a sample DFG for the behavioural description shown in Figure 1.1. The hardware circuit description is divided into segments, datapath unit and control unit. The datapath includes the functional units such as multipliers, arithmetic logical units, and the storage units while the control unit coordinates the data flow between the datapath elements. Traditionally HLS is divided into datapath synthesis and controller synthesis. Datapath synthesis can be modelled as the process of searching a complex multidimensional space represented by the

set of possible schedules, allocations, and bindings that can realize a given behavioural specification.

1.5 Phases of HLS

The various phases or tasks of HLS include compilation, transformation, scheduling, allocation, binding and RTL generation. During compilation the behavior of the system specified in the form of an algorithmic description or HDL (VHDL or Verilog) is compiled into internal representations. These internal representations are generally in the form of a DFG or a Control Data Flow Graphs (CDFG). Further in the transformation phase, the generated DFG is transformed into an optimized DFG or a suitable DFG for scheduling and allocation purpose. Dead-code elimination, common subexpression elimination, loop unrolling, constant propagation and code motion are some possible transformations which can be done on an application. Once the transformations are done, the Design Space Exploration (DSE) process is performed. During DSE several choices have to be evaluated for executing any decision. Therefore, it is important to perform DSE at early design stage or higher abstraction level (behavioural level) in order to investigate tradeoffs between all possible design goals, and to select the most appropriate solution. Finally, to realize a RTL design, HLS performs scheduling, allocation and binding. Scheduling divides the algorithmic behaviour/DFG into control steps. Each step contains a small section of code that can be executed in a single clock cycle. This process optimizes the number of execution steps based on constraints of hardware resource and cycle time. Allocation decides how much resources are needed in hardware while binding map the instructions and variables to hardware components, such as adders, multipliers, and registers [64]. The scheduling, allocation and binding phases are described in detail in next sections.

- *Scheduling*

The scheduling is a process which maps operations belonging to the algorithmic description onto a set of discrete time steps, in a way such that all data dependencies/precedence constraints specified in the algorithmic description are met. The mapping of operations to time steps is done such that the total number of time steps required to implement the specified behaviour meets the given timing constraints and minimizes implementation area. Scheduling can either be constructive or iterative [1, 66, 67, 69].

In constructive scheduling the solutions are constructed by adding operations/nodes one at a time until all the operations have been scheduled. As Soon As Possible (ASAP), As Late As

Possible (ALAP), List scheduling [13], Force direct scheduling [14], and Integer linear programming based scheduling [15, 16] fall under constructive category. ASAP is the simplest type of scheduling. It assumes that the number of Functional Units (FU's) required are already specified. Further, process arranges the operations topologically according to their data or control flow. Once the operations are sorted, they are selected one by one from the list in order and scheduled into earliest control step possible, preserving its dependency and the resource availability. However, another constructive scheduling approach, ALAP, places the operations in the latest possible control step. ALAP uses the number of steps resulting from the ASAP schedule as a latency constraint [1]. ASAP, ALAP are also referred as the unconstrained scheduling algorithms. List scheduling is primarily resource-constrained scheduling algorithm. The list-based algorithm takes a sequencing DFG and resource constraints as inputs and generates a scheduled sequencing DFG as output. The operations available for scheduling are kept in a list for each control step. This list is further ordered by some priority function, either mobility of the vertex or the length of path from the operation to the sink while ranking the vertices in decreasing order. An operation on the list is scheduled one by one if the resource needed by the operation is free; otherwise, it is deferred to the next clock cycle. Further, a Force-directed scheduling is a heuristic algorithm that can consider both resource and time constraints. The basic idea of this algorithm is to balance the concurrency of operations without increasing the total execution time to maximize the utilization of resources such that the number of required resources is minimal [1].

However, in iterative scheduling the designer starts with an initial (random) solution and iteratively updates the solution. Finally a scheduling solution is generated which is optimal and satisfies the user constraints of power/area and latency. In iterative scheduling the designer possesses multiple designing solutions which are generated in intermediate steps. Genetic Algorithm (GA) based scheduling, ant colony based scheduling, simulated annealing based scheduling are some examples of iterative approaches [65, 68, 70].

- *Allocation and Binding*

Allocation involves mapping operations onto functional units, assigning values to registers, and providing interconnections between operators and registers using buses and multiplexers. While binding is the task to assign operation to particular resource such as computation to functional unit, storage to register and data transfer to interconnect. Binding can be solved by using various graph theoretic techniques like clique partitioning [1, 13, 71], circular-arc graph colouring [1, 13, 71] or left edge algorithm [1, 13]. In clique partitioning, an undirected graph

reffered as “compatibility graph” is constructed to analyze the compatibility between the operations of the graph. Two operations are compatible and can use same resources if they need resources of same type and are scheduled in different clock cycles.

However, in graph colouring a respurce conflict graph is constructed to analyze the conflicts of the operations, wherein, the graph is an undirected graph whose vertex set is in one-to-one correspondence with the operations and whose edge set denotes the conflicting vertex pairs. In such a resource conflict graph, two operations have a conflict if they are not compatible. The conflict graph and compatibility graph are complementary to each other. The choice between them is driven by the type of circuit.

Furthermore, in the left edge algorithm [1, 13], the birth time of a variable is mapped to the left edge, and the death time of a variable is mapped to the right edge. The variables are sorted in increasing order of their birth time. The first variable is then assigned to the first register. Then, the current register receives the next variable whose birth time is larger or equal to the death time of the previous variable.

1.6 Why HLS?

There has been an increase in trend towards automating synthesis at higher levels of designing in the recent years. Also, there has been substantial interest shown in RTL design obtained from higher levels of abstraction (algorithmic) using HLS. There are a number of reasons for this [1, 63, 65]:

- *Shorter design cycle*: Automation has reduced the designing time and manpower involvement metrics. Hence there is a reduction in the overall cost of the chip.
- *Continuous and reliable design flow*: HLS facilitates a continuous and reliable automatic translation of high level specification into RTL description of the circuit in the form of VHDL or Verilog.
- *Fewer errors*: Correct design decisions at the higher levels of circuit abstraction can ensure that the errors are not propagated to the lower levels.
- *The ability to search the design space:* Automating the design process helps in producing several designs for same specification in a reasonable amount of time. This benefit helps the designer in exploring different trade-offs between cost, speed, power and other factors to take an existing design and produce a functionally equivalent one that is efficient.

- *Easy availability of IC technology*: As more design expertise is moved into the synthesis system, it becomes easier for a non-expert to produce a chip that meets a given set of specifications.

1.7 Thesis Organization

The rest of the thesis is organized as follows: chapter 3 describes the proposed framework to solve the problem of DSE during power-performance trade-off for data intensive applications. Chapter 4 describes a framework to solve the problem of exploration of low cost optimal *k*-cycle transient fault secured datapath during power-performance trade-off for data intensive applications. Chapter 5 solves the problem of exploration of low cost optimal *k*-cycle transient fault secured datapath during area-delay trade-off for control intensive applications. In chapter 6, a framework to solve the problem of exploration of low cost optimal *k*-cycle transient fault tolerant datapath based on power-performance tradeoff for data intensive applications is presented. Moreover, chapter 7 solves the problem of exploration of low cost optimal Trojan secured datapath during behavioural synthesis for data intensive applications. Further, the results of the proposed approaches in context to the problems, for various well known HLS benchmarks indicating exploration time and quality improvements obtained, compared to the current existing approaches are provided in chapter 8. Chapter 9 concludes the research work presented in the thesis and provides future scope of work in this area.

Chapter 2

Previous Work and Thesis Contribution

2.1 Related Work

The problem of DSE in HLS is a NP-complete problem [72, 73, 74]. In the literature, many attempts have been made to solve the DSE problem in HLS [79-83]. The approaches developed; aim at exploring the design space along with balancing multi-conflicting issues during generation of the optimal/near-optimal design alternative (or Pareto front). For solving various NP complete problems, GAs is the most popular evolutionary algorithms in terms of diversity of their applications. In order to solve DSE problem, GA is used by many researchers in [18, 20, 21, 65, 66]. For example, researchers in [18] used a time constrained scheduling based on GA. In [18], authors combined the constructive scheduling methods with GA and later used for searching a suitable order to perform scheduling. The work presented an encoding scheme where allocation of supplementary resources was done during scheduling, to deal with the lower bound estimations. Authors in [20, 21, 65, 66] used GA to solve integrated scheduling and datapath exploration problem. In these approaches, the chromosome contains the scheduling information and the datapath information. In [20], the scheduling information is encoded with '*node priority*'. However, authors in [21], used the scheduling information in chromosome encoded by '*load factor'* and used a heuristic to decode the scheduling information from encoded chromosome. Authors in [65, 66] encoded scheduling information in chromosomes as '*work remaining*'. However, the second part of the chromosome in [20, 21, 65, 66] is encoded with maximum number of functional units available during scheduling. Furthermore, in [20] the cost function is evaluated on the basis on area-latency tradeoff. But, there is no concept of total execution time, data pipelining and power during exploration. Further, the chance of yielding an optimum result is not guaranteed. Researchers in [21] did not consider dynamic power while calculating total power. In the work, a multi-structure chromosome representation for the datapath nodes was used for scheduling. The approach also had a drawback of huge computation time besides

generating non-optimal solutions in some cases. Authors in [65, 66] used binary encoding of the chromosomes for DSE in architectural synthesis for area-latency trade-off. Moreover, authors in [65, 66], optimized area and latency, but failed to consider power and execution time (function of latency as well as cycle time for pipelined dataset), which are critical issues for modern handheld, battery operated high speed devices. In order to explore new solutions the approaches [18, 20, 21, 65, 66] perform genetic operator (such as crossover and mutation) between two chromosomes. In [19] a discrete Particle Swarm Optimization (PSO) based exploration method is proposed to solve the DSE problem in HLS. In the approach every swarm explores the design space by considering all conflicting objective simultaneously. The approach suffers from several drawbacks. The authors divided the swarm into sub-swarms and each objective was accomplished by one sub-swarm only. Hence, the technique required a large swarm size which may lead to heavy computation time per iteration. In the work, the authors have not considered the concept of local best while exploration. While updating velocity, the authors updated only the direction keeping the step length constant. Another drawback of the approach is that, there is no concept of mutation and clamping in case of boundary overreach problem.

Further in [17], authors described an approach based on integration of GA with PSO referred as Weighted Sum Particle Swarm Optimization (WSPSO). In [17] authors adopted the encoding scheme from [20], which is a combination of scheduling information and maximum available FUs. In their work, the concept of global and local best solution/position is used. To find new solutions, crossover is performed between current position with global best position and local best position. Thus, to incorporate GA, crossover is performed, which is the basic operator of GA and to incorporate PSO, the crossover is performed between current position and global and local best position. The shortcoming of this approach is that mathematically no velocity parameter is used while updating the particle position. Moreover, authors used a weighted combination of latency, area and power during fitness evaluation. But, the metrics such as execution time and actual power are not taken into account in cost function determination.

Besides these approaches, certain tools are introduced which deal with the DSE problem in HLS. In [22] a tool called SystemCoDesigner has been introduced which deals with tradeoff between area-delay. The tool offers automated and fast DSE with prototyping of behavioral systemC models. Some other commercial tools like GAUT [23], LegUp [24], ROCCC [25] and CatapultC [26] are also available in the market for electronic design automation. GAUT

takes a C/C++ behavioral description as input for automatically generating equivalent RTL implementation based on constraints of throughput and clock period. LegUp is an open source HLS tool available for FPGA-based processor/accelerator systems. Further, another tool called AutoPilot is introduced in [27] which address the problem of exploration in HLS. It performs C/C++/systemC-to-RTL synthesis. The tool was targeted for FPGAs.

The approaches mentioned so far suffered from multiple drawbacks and were not useful. Therefore, one of the objectives of this thesis is to develop an efficient DSE methodology in HLS which addresses the above drawbacks.

Over the years the process of DSE has evolved where the requirements specified by the user have also become more convoluted, ranging from simple area-delay tradeoff in initial years to complex power-delay-reliability tradeoff in recent years. To resolve this, some HLS approaches were proposed which included the consideration of fault security aspect with hardware redundancy, but without focusing on low cost solution of an optimized fault detectable design based on user power-delay constraint. The literature includes works that only deal with the fault detection issue of the designs without ability to explore a low cost optimized fault detectable datapath based on user specified power-delay. For example, HLS approaches such as [28, 29, 30, 31, 87] just included the aspect of single cycle fault security with hardware redundancy, but without any focus on evolving/exploring an optimal multi-cycle fault secured design based on user power-delay constraints. In [28] authors use duplication of the CDFG and map the second onto the same hardware as the first, adding FUs as needed. The technique uses the algebraic properties of associativity, distributivity, and commutativity to aid mobility in scheduling the duplicate CDFG and thus take better advantage of idle resources. The approach in [29] involves partitioning of the CDFG into regions or sub graphs. The authors presented a hardware redundancy based Concurrent Error Detection (CED) approach which breaks the data dependences between the nodes. This is done to improve the sharing between normal and duplicate computations. The original and the duplicate computations which are represented by a region are performed on distinct hardware. This is done so that, every regions output can be compared to identify the faults within the regions. For this, voting on the results of the regions is done. In [30], a CED scheme is employed to detect and isolate the faults within a system while it is in use. In [31] authors investigated a method for exploring the tradeoff between the area and latency of the CED design in HLS. The approach sometimes used hardware redundancy or time redundancy or a combination of both to produce fault secure designs. Designs were made secure on the

basis of check pointing introduced in the system. Instead of adding extra FUs for fault detection, they use re-computation on the same hardware using different allocations. Therefore approaches [28, 29, 30, 31] are all fault detectable approaches (using hardware redundancy) with no provision of producing an optimized fault detectable datapath system based on conflicting power and delay constraint of user.

Additionally, there have been no approaches developed which concurrently propose a multi-objective DSE process of a multi-cycle (or single cycle) fault tolerant design during HLS. A complete fault tolerant system should possess different capabilities. Depending upon the fault tolerance level required it should be capable of identifying the fault, detecting it, followed by its isolation, masking and then recovering from it. Efforts have only been made for the error detection issue of the designs without ability to isolate the faults as well as explore the optimized fault tolerant datapath based on power-performance objective. The fault detection technique involves the redundancy factor to identify the faults prevailing in the system. It either uses the hardware redundancy or the time redundancy or a combination of both to determine the presence of fault within a system. Therefore, exploration of a multi-cycle fault tolerant datapath for conflicting user constraints becomes non-trivial. In the literature so far, only one approach has been proposed by authors in [32] who discussed a HLS approach for multi-cycle transient fault tolerant datapath. However, there was no algorithm for exploration of an optimal fault tolerant datapath based on power-performance constraint. Also, the work did not include any concept of multi-cycle transient fault during DSE. Moreover, a Triple Modular Redundant (TMR) system for k-cycle faults tolerance for Single Event Transient (SET) was presented. Wherein, the outputs of the units were voted upon by the help of voter, to mask the errors. Additionally, comparators were used to detect the difference in the outputs of the units. Therefore, [32] involved higher redundancy which sometimes involved TMR system with tripled resource usage.

The approaches in the literature so far were not capable to address transient fault security/tolerance and generate low cost optimal fault secured/tolerant datapath simultaneously and therefore were not much useful. However, some of the approaches which could handle the transient faults involved higher redundancy leading to generation of a non-optimal design solution. Therefore, one of the objectives of this thesis is to develop a methodology which generates a low cost optimal transient fault secured/tolerant datapath during HLS.

With the emergent globalization of System-on-Chip (SoC) designs, penetration of hardware Trojan in Intellectual Property (IP) cores resulting from untrustworthy Third Party (3P) vendors has become a matter of grave security concern amongst the SoC integrators. Hardware Trojan's are malicious hardware components embedded by adversaries in order to induce malfunctioning of ICs [95, 96, 97]. During the design process an adversary may corrupt the IP by inserting hardware Trojan into it. This matter gets further intricate as hardware Trojans can be of multiple types [33, 34]. To have a trustworthy design it should be ensured during HLS that any possible infection of 3PIP is detectable. In case of HLS, the hardware Trojan mostly considered is the one which is capable of maliciously altering the digital output of a 3PIP. The detection procedure as suggested by [35] is accomplished by having IP cores of same functionality from different vendors. This is because different vendors will have different implementations and it is less likely that both are Trojan infected. Even if they are, the chances of different vendor IPs generating same output behavior is considered extremely uncommon. However, detection process of the Trojan during design of hardware Trojan secured schedule in HLS inevitably requires multiple redundant hardware instances from different vendors, which if not accounted for its power and delay during fitness evaluation, may result in a secured circuit violating user constraint. The focus on hardware Trojan detection during HLS has been very little with absolutely zero effort so far in DSE of a user multi-objective constraint optimized hardware Trojan secured schedule. However a number of approaches have been proposed for Trojan detection at lower levels of chip design [89, 90, 91, 93, 94]. This problem mandates attention as producing a Trojan secured schedule is not inconsequential. Merely the detection process of Trojan is not as straightforward as CED of transient faults as it involves the concept of multiple 3PIP vendors to facilitate detection [92], let aside the exploration process of a user optimized Trojan secured schedule based on Multiobjective (MO) constraints. Efficient vendor allocation procedure needs to be devised for Trojan detection during HLS, besides robust and adaptive exploration scheme for low cost optimal hardware Trojan secured scheduling. A low cost optimal Trojan secured schedule indicates an optimized robust Double Modular Redundant (DMR) scheduling obtained through a heuristic comprising of intelligent hardware assignment to operations such that any possible Trojan infection in the underlying hardware is detectable.

2.2 Objective

The objective of this thesis is to develop highly reliable/hardware secured designs for data intensive and control intensive applications during design of application specific datapath processor at behavioural level. In order to realize this, the following objectives have been set:

- Develop a methodology to solve the problem of DSE during power-performance trade-off for data intensive applications that produces high quality design solutions.
- Develop an approach to solve the problem of exploration of low cost optimal *k*-cycle transient fault secured datapath during power-performance trade-off for data intensive applications.
- Develop an automated approach to solve the problem of simultaneous exploration of low cost optimal *k*-cycle transient fault secured datapath and unrolling factor for control intensive applications during area-delay trade-off.
- Develop a execution time prediction model for faster exploration process in case of single loop based CDFGs without tediously unrolling CDFG loop completely.
- Develop an approach to solve the problem of exploration of low cost optimal *k*-cycle transient fault tolerant datapath based on power-performance tradeoff for data intensive applications.
- Develop an approach that solves the problem of exploration of low cost optimal Trojan secured datapath during behavioural synthesis for data intensive applications.

2.3 Summary of Contribution

The focus of this thesis is to provide a number of low cost solutions to the aforesaid problem in the field of security (against hardware Trojan) and reliability (against transient fault) aware HLS for both data and control intensive applications.

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

- Solve the problem of DSE during power-performance trade-off for data intensive applications.

  [Publications: J1, J5, C10]

  a) Proposed a novel temperature dependent bacterial foraging optimization methodology for automated exploration of datapath in HLS, capable of yielding optimal results.

b) Introduced a novel chemotaxis algorithm for exploration drift, replication algorithm for inducing efficient exploration ability and elimination-dispersal algorithms for sudden diversity introduction.

- Solve the problem of exploration of low cost optimal *k*-cycle transient fault secured datapath during power-performance trade-off for data intensive applications.
  [Publications: J2, C7, C8, C9]
  a) Proposed a novel fault security algorithm for handling single and multi-cycle transient faults.
  b) Proposed a low cost approach for generating a high quality fault secured structure based on user provided requirements of power-delay, which is capable of transient error detection in the datapath.
  c) Introduced a novel scheme for selecting appropriate edges for inserting cuts in the scheduled DFG minimizing delay overhead associated with transient fault security.
  d) Proposed a novel execution time model for estimating the execution time of a transient fault secured/Trojan secured design during DSE process.
  e) Proposed a novel fitness function, used for design quality assessment in DSE process.
  f) Proposed a novel multi-cycle SET fault security aware multi objective DSE methodology that explores an optimal combination of transient fault secured DMR datapath configuration.
- Solve the problem of exploration of low cost optimal *k*-cycle transient fault secured datapath during area-delay trade-off for control intensive applications
  [Publications: J4]
  a) Proposed a novel multi-cycle SET fault security aware multi objective DSE methodology that explores an optimal combination of transient fault secured DMR datapath configuration and loop Unrolling Factor (UF) for CDFG.
  b) Proposed an estimation model for computation of execution delay of a loop unrolled CDFG (based on a resource configuration explored) without tediously unrolling the entire CDFG for the specified loop value.

- Solve the problem of exploration of low cost optimal *k*-cycle transient fault tolerant datapath based on power-performance tradeoff for data intensive applications.
  [Publications: J3, C11]
  a) Proposed a novel multi-cycle transient fault tolerant algorithm that has capability to isolate original and duplicate units in a DMR with respect to the transient fault.
  b) Proposed a novel equivalent circuit that works with DMR systems performs the function of extracting the correct output from the DMR design.
- Solve the problem of exploration of low cost optimal Trojan secured datapath during behavioural synthesis for data intensive applications.
  [Publications: C6, C12]
  a) Proposed a novel encoding scheme for representing bacterium in the design space (comprising of candidate datapath resource configuration and vendor allocation information for hardware Trojan secured datapath).
  b) Proposed a novel exploration process of an efficient vendor allocation procedure that assists in yielding a low cost hardware Trojan secured datapath within user constraints.

Chapter 3

Adaptive Bacterial Foraging Driven Datapath Optimization: Exploring Power-performance Trade-off in HLS

This chapter presents a novel application of Bacterial Foraging Optimization Algorithm (BFOA) in the area of DSE of datapath in HLS for data intensive applications. For the DSE process, the BFOA has been transformed into an adaptive automated DSE framework that is capable to handle tradeoffs between power and execution time during HLS. The BFOA-DSE is capable to resolve orthogonal issues such as enhancing Quality of Result (QoR) as well as exploration speed, thereby being able to produce higher-quality results in lesser exploration time than existing approaches [20, 36]. This is the first work which directly maps the BFO process for multi-objective DSE during power-performance trade-off for data intensive applications in HLS. The work proposes a novel chemotaxis driven exploration drift algorithm, a novel replication algorithm for manipulating the position of the bacterium by keeping the resource information constant (useful for inducing exploitative ability in the algorithm). Moreover, a novel Elimination-Dispersal (ED) algorithm is proposed to introduce diversity during the exploration process. The detailed explanation of the proposed methodology along with the demonstration of the proposed framework has been given in subsequent sections.

3.1. Description of Proposed Methodology

3.1.1. Problem Formulation

Given a DFG, explore the design space and determine an optimal resource configuration, $X_i = \{N(R_1), N(R_2), N(R_d)....N(R_D)\}$ satisfying conflicting user constraints and minimizing the overall cost. The formal formulation of the problem is:

For a given DFG find a resource combination (X_i):

$$X_i = \{N(R_1), N(R_2), N(R_d)....N(R_D)\};$$

with minimum hybrid cost: ($P_{T,}$ T_E);

subjected to: $P_T \leq P_{cons}$ and $T_E \leq T_{cons}$.

Where, $N(R_d)$ is the number of instances of resource type 'R_d'; 'D' is the total number of resource types; 'X_i' is a candidate resource combination for optimal solution; 'P_T' and 'T_E' are the power and execution time consumed by a candidate resource combination; 'P_{cons}' and 'T_{cons}' are power and execution time constraints specified by the user.

3.1.2. Motivation of using BFOA in Context of Proposed Problem

The DSE process for Application Specific Integrated Circuits (ASICs) is an intricate process which involves identifying the best solution from a set of given design alternatives of assorted nature. The DSE algorithms proposed so far using the evolutionary approach such as PSO, GA and hybrid GA do not provide flexible options for guided/adaptive searching such as change in directions when a certain search path is found unproductive. Moreover, PSO is known to be a highly sensitive algorithm, therefore failing to clinically pre-tune the parameters often would result in convergence to local optima. However, bacterial foraging uses a simplified framework and is less sensitive than other evolutionary techniques. BFOA comprises of primarily of two major steps: chemotaxis and dispersal for locomotion of bacterium. Using locomotion mechanisms (such as flagella) bacteria can move around in their environment, sometimes moving chaotically (tumbling and spinning), and other times moving in a directed manner that may be referred to as swimming. Therefore, the intuition or science behind adopting Bacterial foraging is the simplified nature of its heuristic framework and features that provide directed based searching compared to typical evolutionary algorithms such as GA and PSO.

3.1.3. Proposed BFOA Driven DSE Methodology

Social foraging behaviour of the *E. coli* inspired BFOA is aimed in optimizing the real world problems in several application domains. The real bacterial system involves four basic mechanisms viz. chemotaxis, swarming, replication and elimination dispersal [37] The proposed BFOA-DSE process imitates these basic mechanisms in order to solve the DSE problem in HLS.

The proposed mapping of BFO for DSE is as follows:

Position of bacterium → Resource configuration

Dimension → Number of Resource types

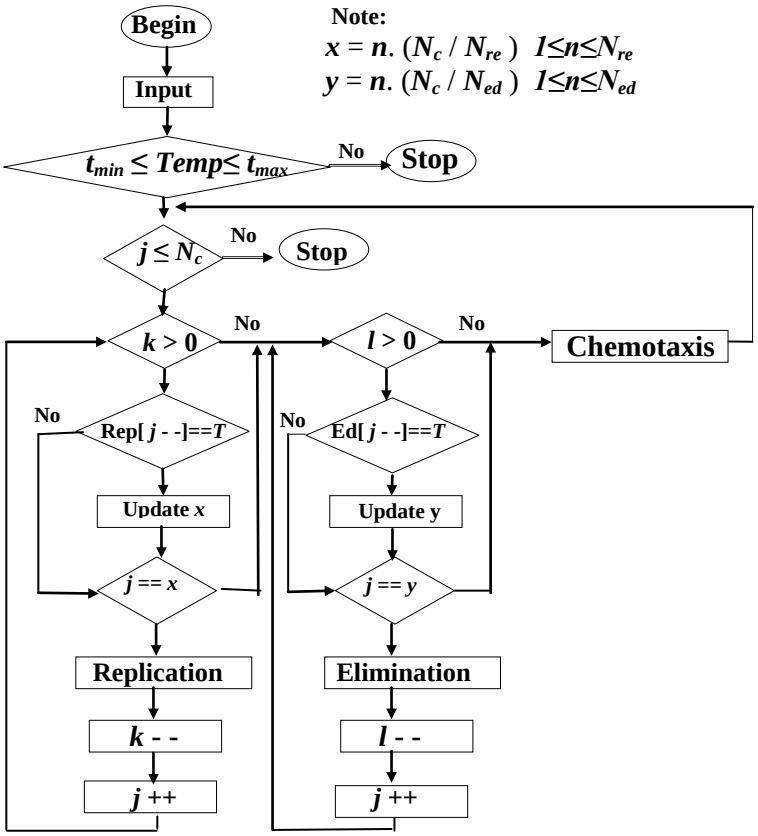


Figure 3.1 Proposed BFOA-DSE Methodology

The flowchart of proposed BFOA driven DSE algorithm is shown in Figure 3.1 Based on the flowchart provided in Figure 3.1, the description is as follows:

The inputs to the proposed framework are behavioral description of application in the form of DFG that describes datapath, user specified design constraints for power and execution time (with user specified weight factor), and module library. Module library comprises information of viz. energy consumed by each resource in Picojoule (pJ), latency of each resource in nanoseconds (ns), hardware area of each resource (#of transistor) and user specified maximum availability of resources. In the proposed approach, the initial population has multiple bacteria. Therefore, the initialization of bacterium positions corresponding to the resource configurations is done. Imitating the biological phenomenon of an *E. coli* bacterium, the proposed DSE methodology iterates within the valid temperature range [t_{min}, t_{max}] at which an *E. coli* can survive (*Note*: Investigations from previous literature [38–41] have revealed the motility range of *E. coli* between, t_{min} = 25° C and t_{max} = 45° C; while elimination of bacterium can occur at high temperature such as 40° C). Within this motility (valid) temperature range through chemotactic movement in every step (j) of each bacterium, the proposed DSE explores new feasible solutions.

Movement of a bacterium from one position to the other is characterized as a chemotactic movement. The bacterium moves to a new unexplored position based on step length (*C(i)*), past position(X_i^{Last})and random number (Δ(i)). However, after a designer specified periodic intervals ('x^{th}' and 'y^{th}' iteration step respectively), the process of replication and elimination dispersal occurs. The replication and ED algorithm is repeated (based on its corresponding periodic intervals) for 'N_{re}' and 'N_{ed}' times, where, 'N_{re}' is the maximum number of replication steps and 'N_{ed}' is the maximum number of elimination dispersal steps to be undertaken throughout the exploration process. Further, corresponding arrays ((Rep [*j*–]) and (Ed [*j*–])) are created for replication and ED process each to store the outcome, checking whether replication and ED has been performed in last iterative step. These storage structures are necessary to determine whether variables '*x*' and '*y*' need up-gradation. If Rep [*j*–] = TRUE then it indicates that in the last iterative step (*j*–), replication has taken place, therefore, '*x*' needs to be updated. Else, the up-gradation is bypassed. Similar logic holds for Ed[*j*–] in terms of operation functionality. In case of DSE, the bacterium positions are dispersed, with an aim of exploring the new positions with better cost. The least fit bacteria eventually die while the healthier bacteria positions yielding better fitness value are retained.

The iteration process continues until the stopping criterion is reached (the stopping criterions are described in later section). Hence, on completion the process yields an optimal solution which is the global best resource configuration for the given application and user constraints.

3.1.4 Models for Evaluation of Design Points During BFOA-DSE

The bacterium positions are determined based on power consumed, execution time, and the cost function illustrating the fitness of the bacteria.

3.1.4.1 Power Model

Power consumption (P_T) by a resource set is represented in terms of static power (P_S) and dynamic power (P_D). 'P_T ' is represented as [21, 36]:

$$P_T = P_S + P_D \tag{3.1}$$

Static power is a function of area of resources and leakage power per transistor. Accordingly, static power is:

$$P_S = \sum_{d=1}^{D} (N(R_d).K(R_d).p_c \tag{3.2}$$

$$P_S = (N(R_1) \cdot K(R_1) + N(R)_2 \cdot K(R)_2 + .. + N(R)_D \cdot K(R)_D).p_c \tag{3.3}$$

Where '$N(R_d)$' represents the number of instances of resource R_d. '$K(R_d)$' represents the area occupied by resource R_d, 'D' is the number of resources (FU's) and 'p_c' denotes the power dissipated per area unit (e.g. transistors).

However, the average dynamic power consumed by a resource configuration is a function of dynamic activity of the resources and can be formulated as [21, 36, 42]:

$$P_D = \frac{N.E_{FU}}{L + (N-1).T_C} \tag{3.4}$$

Where, 'E_{FU}' is the total energy consumed by the resources obtained [43], 'N' is the c, 'L' is the latency of a scheduling solution and 'T_c' is the initiation interval or cycle time of a scheduling solution. Equation (3.4) can be further written as:

$$P_D = \frac{N \cdot (E_{\mathrm{Re}\,s} + E_{mux} + E_{demux})}{L + (N-1) \cdot T_c} \tag{3.5}$$

Where, 'E_{Res}' is the energy consumed by the major FU's such as adders, subtractors, multipliers and comparators. 'E_{mux}' and 'E_{demux}' are the energy consumed by the multiplexers and de-multiplexers used.

Substituting equation (3.5) and (3.2) in equation (3.1):

$$P_T = \frac{N \cdot (E_{\mathrm{Re}\,s} + E_{mux} + E_{demux})}{L + (N-1) \cdot T_c} + \sum_{d=1}^{D} (N(R)_d \cdot K(R)_d) \cdot p_c \tag{3.6}$$

3.1.4.2 Execution Time Model

For a given system with 'D' functional resources the time of execution can be represented as:

$$T_E = [L + (N-1).T_c] \tag{3.7}$$

The equation (3.7) has been adopted from [21, 36, 42], which denotes the total execution time considering data pipelining of N data sets where the mathematical quantity (N-1).T_c indicates the delay consumed by the data (except the first element) during pipelining. The variables L, N and T_c have already been defined in section 3.1.4.1.

3.1.4.3 Model for Fitness Function

The fitness function (considering execution time and power consumption of a solution) is defined as [36]:

$$C_f(X_i) = \phi_1 \frac{P_T - P_{cons}}{P_{\max}} + \phi_2 \frac{T_E - T_{cons}}{T_{\max}} \tag{3.8}$$

Where, $C_f(X_i)$ is the fitness of bacterium X_i, ϕ_1 and ϕ_2 are the user defined weights for power and execution time parameters and $T_{\max}$ is the maximum execution time of a solution in

Table 3.1 Module Library Used

Major FU`s	Add16	Mul16	Sub16
Energy (pJ)	0.739	9.8	0.739
Area(#transistor)	2032	2464	2032
Latency (ns)	270	11000	270

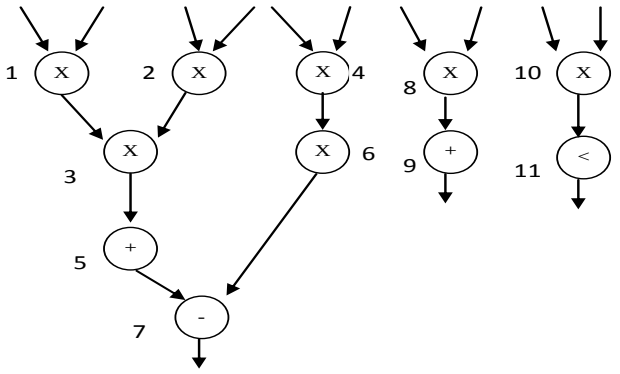


Figure 3.2 DFG of Differential Solver Benchmark [5]

design space while P_{max} is the maximum power of a solution in design space, while the functions to calculate P_T and T_E are stated in equations (3.6) and (3.7) respectively.

3.2 Description of the Proposed Methodology with Demonstration

3.2.1 Module Library Information and Operating Constraints

The module library information used is shown in Table 3.1. The values of E_{FU}, area and latency assumed have been adopted from [43, 101, 102, 103].

For the purpose of explanation, a DFG for Differential Solver benchmark shown in Figure 3.2 is used for demonstration of the proposed algorithm. Figure 3.2 has four types of resources (i.e. D = 4). The assumed values for the sake of demonstration are: maximum available multiplier FUs: 4, adder FUs: 2, subtractor FUs: 2, and comparator FUs: 1; Number of data sets, N = 1000; while power dissipated per transistor (p_c) is assumed to be 29.33 mW; additionally, number/type of mux/demux is directly extracted from the scheduling solution. *Note: the proposed approach is capable to handle evolving technology by altering the component values in the module library. With change in technology, the supply voltage is scaled resulting in different 'p_c' value. Further, due to technology scaling the number of transistors for each component specified in the library can be changed. Therefore, the proposed theory is capable to adapt to evolving technology.* Since, Figure 3.2 has four types of resources (i.e. D = 4), therefore, a bacterium position can be given by: X_i = (N(mul),

N(add), N(sub), N(comp)). Additionally, we will assume some constraint values for Power (P_{cons}) and Execution time (T_{cons}) as well as user defined specifications.

The goal of exploration problem is to generate and evaluate design points or configurations by simultaneously meeting the user provided constraints for power and execution time.

3.2.2 Maximum Threshold

Before evaluation of the fitness of a design, a minimum and maximum value of power and execution time has to be determined. The maximum values of the power and execution time are identified corresponding to the provided boundary constraint values of the resources. Typically, an application consumes maximum execution time when a single instance of a resource type is available at a particular timestamp (i.e. when using minimum resources). Such execution corresponds to a serial implementation of the target application. However, a maximum power is consumed by utilizing maximum available resources to execute the operations. This indicates maximum parallelization of the target application.

3.2.3 Boundary Constraints Check Module

To check whether the provided user constraints values are within the acceptable limits a boundary constraint check is performed. The following conditions are checked for each parametric constraint specified:

1. If $P_{min} > P_{cons} > P_{max}$ *or* $T_{min} > T_{cons} > T_{max}$
2. If above condition is true then stop and correct the constraints.
3. Else proceed to next step of the process.

3.2.4 Initialization of Bacterium

The bacteria are initialized to uniformly cover the design space. For a DFG, the bacterium position 'X_i' of an 'i^{th}' bacterium is given as:

$$X_i = (N(R_1), (N(R_2),..(N(R_d).. (N(R_D))$$

The efficiency of an exploration algorithm depends upon how well is the initial population distributed over the design space. Therefore, to have a better exploration the algorithm initializes the bacteria as follows:

- The first bacterium is initialized by minimum resources (serial implementation):
  $X_1 = (min(R_1), min(R_2),.. min(R_D))$

  $X_1 = (1, 1, 1, 1)$
- The second bacterium is initialized by maximum resources (maximum parallel implementation):
  $X_2 = (max(R_1), max(R_2),.. max(R_D))$

Therefore based on the user defined resources assumed in section 3.2.1, X_2 can be customized as follows:

X_2= (4, 2, 2, 1)

- The third bacterium is initialized by average of maximum and minimum values:
  X_3= *((min(R_1)+max(R_1))/2,.....,((min(R_D)+max(R_D))/2*

  Therefore, X_3 can be customized as:

  X_3 = (2, 1, 1, 1)
- The rest of the bacteria (X_4....X_n) are initialized by following equation:

$$N(R_d) = (\min(R_d) + \max(R_d)) / 2 \pm \alpha \quad (3.9)$$

This function has been proposed to introduce an element of stochasticity (as well as diversity) into the initialization process. Where, '*min(R_d)*' is minimum resource of d^{th} type, '*max(R_d)*' is maximum resource of d^{th} type (obtained from module library) and 'α' is a random value between *max(R_d)* and *min(R_d)*.

3.2.5 Calculation of Fitness of a Bacterium

After the initialization of bacteria is performed (as shown in section 3.2.4), the fitness of initial bacteria is identified. The fitness is evaluated using equation (3.8). For determining the initial cost of the solution/bacterium, P_T and T_E are evaluated from equation (3.6) and (3.7). For example, the calculation of total power (P_T) of X_1= (1, 1, 1, 1) using equation (3.6) is as follows:

$$P_T = \frac{N \cdot (E_{Res} + E_{mux} + E_{demux})}{L + (N-1) \cdot T_c} + \sum_{i=1}^{D} (N_{R_i} \cdot K_{R_i}) \cdot p_c$$

$$P_T = \frac{1000 \cdot (7*(9.8) + 2*(0.739) + 1*(0.739) + 1*(0.739) + 8*(0.2) + 4*(0.2))}{66270 + (1000-1) \cdot 66000}$$

$$+(1*2464 + 1*2032 + 1*2032 + 1*2032) * 29.33$$

= 0.25 mW

Similarly, the execution time is calculated using equation (3.7) as :

$$T_E = [L + (N-1).T_c]$$

$$T_E = [66270 + (1000-1).66000]$$

= 66 ms

*Note- the values of L=66270ns and T_c = 66000ns are derived from the scheduled DFG with resource combination: 1 (*), 1 (+), 1(-), 1(<).*Further, P_{max} = 0.587mW and T_{max} = 66 ms have been calculated based on worst case analysis of the scheduled DFG. For calculating the

Begin

1. $C(i) = C(i) + 2$ *// Set the step size; initial C(i) = 0.*

If $(C(i) > N(R_d)^{\max})$

$C(i) = C(i)^{New} - (C(i)^{New} - (C(i)^{Last} - 2))$

Else if $(C(i) < N(R_d)^{\min})$

$C(i) = C(i)^{New} - (C(i)^{New} - (C(i)^{Last} + 2))$

2. Tumble: Generate a random vector $\Delta(i) \in \Re$ with each element

$\Delta_m(i)\, m = 1, 2, ...D$ a random number in [-1, 1].

3. For *i* = 1 to *p* Do

3.1 Compute cost: $C_f(N(R_1), N(R_2), ...N(R_d).....N(R_D))$

3.2 $X_i^{Last} = X_i$

$C_f(X_i^{Last}) = C_f(N(R_1), N(R_2), ...N(R_d).....N(R_D))$

3.3 **Move:** Let $X_i^{New} = X_i^{Last} + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$

3.3.1 If $(X(R_d)_i^{New} < 0)$

$X(R_d)_i^{New} = X(R_d)_i^{New} + 2 \mid X(R_d)_i^{New} \mid$ *// techniques to handle boundary problem*

Else If $(X(R_d)_i^{New} > X(R_d)^{\max})$

$X(R_d)_i^{New} = N(R_d)^{New} - 1)$ *// techniques to handle boundary problem*

Else If $(X(R_d)_i^{New} < N(R_d)^{\min})$

$X(R_d)_i^{New} = N(R_d)^{New} + 1)$ *// techniques to handle boundary problem*

3.3.2 If X_i^{New} exists

Goto: **Move** in Step 3.3

3.4 **Compute cost:** $C_f(X_i^{New}) = C_f(N(R_1), N(R_2), ...N(R_d).....N(R_D))$

3.4.1. If $(C_f(X_i^{New}) < C_f(X_i^{Last}))$

$C_f(X_i^{Last}) = C_f(X_i^{New})$

$X_i^{Last} = X_i^{New}$

Else

Tumble: Generate a random vector $\Delta(i) \in \Re$ with each element

$\Delta_m(i)\, m = 1, 2, ...D$ a random number in [-1, 1].

Goto: **Move** in Step 3.3

3.5. *i*++

4. *Temp = Temp* + Δt

Figure 3.3 Pseudo code for Proposed Chemotaxis Algorithm

cost equal weightage to power and execution time is given (ϕ_1 = ϕ_2 = 0.5). Finally,

substituting the values in equation (3.8), the fitness of the bacterium X_1 is calculated as :

$$C_f(X_1) = 0.5 * \frac{0.25 - 0.40}{0.58} + 0.5 * \frac{66 - 39}{66}$$

$$= 0.0676$$

Similarly, the fitness of rest bacterium's calculated using equation (3.8) is:

$C_f(X_1)$ = 0.0676 fitness of X_1 (1, 1, 1, 1)

$C_f(X_2)$ = 0.0181 fitness of X_2 (4, 2, 2, 1)

$C_f(X_3)$ = -0.121 fitness of X_3(2, 1, 1, 1)

3.2.6 Determination of New Configuration of the Bacterium

Once the initialization and fitness evaluation of initial population is done, the exploration starts and the bacteria (representing a candidate design solution containing resource configuration) moves to new unexplored positions (X_i^{New}). Every bacterium in the population iterates through a process of chemotaxis, replication and elimination dispersal to explore new resource configurations in the design space. Therefore, the process of DSE is driven through BFO containing its biological steps of chemotaxis replication and elimination dispersal.

3.2.6.1 Proposed Chemotaxis Algorithm for Exploration

The chemotactic movement involves two basic steps viz. move and tumble. The bacterium can either move for a certain period of time in the same direction or it may tumble in the design space, therefore, may alternate between these two locomotive operations. The proposed chemotaxis algorithm, motivated from the basic chemotactic movement is shown in Figure 3.3. It is based on proposed chemotaxis function (equation 3.10) which is a modified derivative of basic chemotaxis function proposed in [37, 44]. In context of DSE, chemotaxis helps in exploring new/unexplored resurce configurations within the design space.

The proposed chemotaxis function incorporates the behavior of tumble/swim in order to explore the new design solutions (resource configurations). The proposed chemotaxis function is:

$$X_i^{New} = X_i^{Last} + C(i) \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \qquad (3.10)$$

Where, X_i^{New} is the new resource configuration of i^{th} bacterium, X_i^{Last} is the last resource configuration of i^{th} bacterium, *C(i)* is the step size taken in random direction specified by the tumble and Δ is a random vector whose elements lie in [-1, 1].

In context of DSE in HLS, a constant (as well as small) step size (*C(i)*) is not productive owing to rendering unable to explore the wide design space quickly. Therefore, *C(i)* is continually increased by a constant length in every iteration in the proposed chemotaxis abiding the lower and upper threshold limits specified by the designer. This feature is shown in step 1 of the algorithm (in Figure 3.3), while step 3 indicates the adaptation ability of the algorithm when invalid solution for a certain dimension is obtained.

3.2.6.2 Demonstration of Proposed Chemotaxis Algorithm

For the proposed DSE, it is assumed that: N_c = 120, N_{re} = 5, N_{ed} = 4; t_{min}= 25deg C and t_{max}=45deg C. Now assuming at *j (chemotactic step counter)* =19 the bacterium's X_1, X_2, X_3 are subjected to chemotactic movement with step size *C(i)* = 2 (as per Step 1 in Figure 3.3), tumble vector = [1, 0.1, 0, 0.9].

Then for first bacterium,

X_1= (1, 1, 1, 1), a new resource configuration, X_i^{New} is yielded as:

$$X_i^{New} = X_i^{Last} + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

$$= (1,1,1,1) + 2*(\frac{1,0.1,0,0.9}{\sqrt{1^2+0.1^2+0^2+0.9^2}})$$
$$= (1,1,1,1) + 2*(\frac{1,0.1,0,0.9}{\sqrt{1.82}})$$
$$= (1,1,1,1) + 2*(\frac{1,0.1,0,0.9}{1.35})$$
$$= (1,1,1,1) + 2*(\frac{1}{1.35},\frac{0.1}{1.35},\frac{0}{1.35},\frac{0.9}{1.35})$$
$$= (1,1,1,1) + 2*(0.74,0.07,0,0.66)$$
$$= (1,1,1,1) + (1.48,0.14,0,1.32)$$
$$= (1,1,1,1) + (2,1,0,2)$$
$$= (3,2,1,3)$$

X_1^{New} = (3, 2, 1, 3) using step 3.3 (Figure.3.3). Since the value of $N(R_4)$is greater than the $N(R_4)^{max}$, therefore, it is clamped using Step 3.3.1 (Figure 3.3).

This yields X_1^{New} = (3, 2, 1, 1).

Begin

For *i* = 1 to *p* Do

For *d* = 1 to *D* Do

1. **Generate** $\alpha \in \Re$**;** where $(N(R_d)^{\min} \le \alpha \le N(R_d)^{\max})$

2. $N(R_d)^{New} = N(R_d) \pm \alpha$

If $(N(R_d)^{New} > N(R_d)^{\max})$

$N(R_d)^{New} = N(R_d)^{New} - 1$ *//techniques to handle boundary problem*

Else if $(N(R_d)^{New} < N(R_d)^{\min})$

$N(R_d)^{New} = N(R_d)^{New} + 1$ *//techniques to handle boundary problem*

//End For of d

3. If X_i^{New} exists

Goto: Step 1

// End For of i

4. *Temp = Temp* + Δt

5. j**++;**

6. **Goto**: Chemotaxis

Figure 3.4 Pseudo code for Proposed Replication Algorithm

Further as the algorithm, this X_1^{New} position has not been explored, so its fitness is evaluated as:

$$C_f(X_1^{New}) = C_f(3,2,1,1) = -0.014;$$

which is accepted as, $C_f(X_1^{New}) < C_f(X_1^{Last})$

where $C_f(X_1^{Last}) = 0.0676.$

Similar, calculations is performed for other bacteria. Once the new values are found, the temp is increased by Δt**.**

3.2.6.3 Proposed Replication Algorithm

In the proposed approach a modified replication algorithm has been proposed which has been customized to the demands of the problem. Regular replication approach where the information is copied to the replicated bacterium will render the resultant configuration redundant in context of DSE. The new bacterium position is therefore manipulated by a random α. However, while replicating from the original, the position ordering) of resource types (dimension) is preserved in the bacterium configuration.

In the proposed approach, 'N_{re}' is the maximum number of times, replication can occur in the entire DSE process. As shown in Figure 3.4, a random variable 'α' manipulates the given

Begin

If $(Temp \geq 40)$

For *i = 1* to *p* Do

1. Arr[i] = $C^{i}_{f}(N(R_1), N(R_2), ...N(R_d).....N(R_D))$

   $i++;$

End For

2. (i) $X_u = X_i(Least \quad fit \quad (Arr[i]))$

   (ii) $X_z = X_i(Best \quad fit \quad (Arr[i]))$

3. **Eliminate the least fit bacterium and perform dispersal**

4. **Dispersal:**

4.1 Select the least fit bacterium.

4.2 Determine the midpoint configuration between the two bacteria (best fit and the least fit).

$X_u(N(R_1), N(R_2), N(R_3).....N(R_D))$ and

$X_z(N(R_1), N(R_2), N(R_3).....N(R_D))$

$$M = \frac{X_u(N(R_d)) + X_z(N(R_d))}{2}$$

4.3 Select any configuration randomly which lies beyond the mid of the configurations.

$X_v(N(R_1), N(R_2), N(R_3).....N(R_D))$

$X_v(N(R_d)^{New}) = M - omega$

$1 \leq omega < X_z(N(R_d))$

4.4 If X_v exists

Goto: Step 4.3 in **Dispersal**

4.5 Calculate cost of this new configuration of the bacterium:

$C_f(X_v) = C_f(X_v(N(R_1), N(R_2), N(R_3).....N(R_D)))$

4.6 If $C_f(X_v) < C_f(X_u)$

$X_u(N(R_1), N(R_2), N(R_3).....N(R_D)) = X_v(N(R_1), N(R_2), N(R_3).....N(R_D))$

Else

Goto: Step 4.2 in **Dispersal**

5. $Temp = Temp + \Delta t$

6. j++;

7. Goto: Chemotaxis

Figure 3.5 Pseudo code for Proposed Elimination-Dispersal Algorithm

configuration with respect to each dimension ($N(R_d)$). After performing replication, resource clamping is performed (if necessary) which limits the resource magnitude on the basis of maximum and minimum available resources of a certain type. Finally, if the new solution

(X_i^{New}) found after replication is found to be already explored, then the replication (step 1) is again performed. However, it is important to note that if the new cost of the replicated bacterium position is found to be higher than its original position, then it is not accepted.

3.2.6.4 Demonstration of Proposed Replication Algorithm

Let us assume *j (current iteration count)* = 24 in the iterative process, then at this step according to the flowchart shown in Figure 3.4 x = 24 (as $k > 0$; N_{re}= 5) which indicates replication can be performed for the bacterium's at this current '*j*' step. Now for X_1= (3, 2, 1, 1), as per replication algorithm after generating new random 'α' for every resource type (every dimension), say we get, X_1^{New} = (5, 2, 2, 2) as new resource configuration.

However, $N(R_1) > N(R_1)^{max}$ and $N(R_4) > N(R_4)^{max}$, therefore resource clamping is needed, which results in X_1^{New} = (4,2,2,1). Since, if this resource configuration is found explored so far, then a new configuration is explored using different 'α' as shown in step3 (Figure 3.4). Once new values are found, temp is increased by Δt.

3.2.6.5 Proposed Elimination-Dispersal Algorithm to Introduce Diversity

The number of times ED algorithm is performed is denoted by 'N_{ed}'. We imitate the biological phenomenon of an *E.coli* where a small rise in the temperature may kill a certain group of bacteria [37, 44] in our ED algorithm of the proposed DSE. Here, the temperature chosen after which the elimination has to be performed is 40deg C. So, in order to implement this behavior, new replacements are randomly initialized over the search space (between the least fit and best fit bacterium position but beyond their midpoint) by eliminating the least fit bacterium as shown in Figure 3.5. If the new replacement found (X_v) is already found to be explored, then the dispersal is repeated (step 4.3). Moreover, similar to replication algorithm it is important to note that if the new cost of the dispersed bacterium position is found to be higher than the replaced bacterium, then it is not accepted.

3.2.6.6 Demonstration of Proposed Elimination-Dispersal Algorithm

Let us assume *j*= 30 in the iterative process, then at this step according to the flowchart in Figure 3.5. x = 30 (as $l > 0$; N_{ed} = 4) which indicates ED can be performed for the bacterium's at this current '*j*' step. However, before performing ED, the initial temp is verified in order to simulate the real life biological phenomenon. Since, the value of Temp is not ≥ 40 deg C, therefore the ED is not executed at this *j* step.

3.2.7 Termination Criteria (Z)

Two important aspects have been considered while deciding the condition of termination:

- The algorithm must not go into infinite loop.
- The proposed approach should not prematurely converge.

Considering these aspects, the termination criteria for the proposed approach has been fixed. The criteria are:

- Terminates when the temperature has reached to the maximum value (45 °C) or reached designer specified '*Nc* ' (i.e. maximum possible chemotactic step).
- When no improvement is seen in the global best among the bacteria population over last 10 iterations (chemotactic steps).

If either of them is true then the exploration process will terminate.

Note – Results of the proposed method are given in chapter 8 section 8.1.

3.3 Summary

This chapter presents a fast and efficient DSE methodology for exploring power/area-performance tradeoff in HLS. The proposed methodology transforms a BFO algorithm for solving DSE of datapaths in HLS. The algorithm mimics the biological phenomenon of *E. coli* bacteria and simulates the DSE process within the operating temperature of *E. coli*. The process is able to efficiently explore the architectures within the design space by yielding optimal results and resolves the multi-conflicting objectives by concurrently handling the orthogonal issues such as QoR and exploration time.

Chapter 4

Automated Design Space Exploration of Multi-Cycle Transient Fault Detectable Datapath based on Multi-Objective User Constraints for Application Specific Computing

Solving the DSE problem for data intensive applications and optimizing area, power and performance has no longer been sufficient now. Specifically, for current generation of systems which demand designs (especially for space applications where radiation induced faults are highly possible) that requires ability to detect errors occurring due to transient faults (such as single event upsets). Transient faults are radiation induced faults which are non-permanent in nature. These nonrecurring faults can be caused by energized particles, environmental noise or electromagnetic interference. The duration of such faults is in order of a few picoseconds [28**,** 37]. In order to achieve high reliability, multi cycle transient fault security [48-52] should be considered as design metric (or constraint) during multi-objective DSE in HLS. Generation of an optimal fault secured datapath structure based on user power-delay budget during HLS in the context *k*-cycle (k_c) transient faults is considered a NP complete problem. This is due to the fact that for every type of candidate design solution produced during exploration; a feasible k_c fault secured datapath may not exist satisfying the conflicting user constraints/budget.

This chapter presents an automated DSE approach of multi-cycle transient fault detectable datapath based on multi-objective user constraints (power and delay) for application specific computing. The proposed DSE framework is driven by an intelligent PSO algorithm which incorporates multiple parameters and conditions to handle efficient exploration. To the best of the authors' knowledge, this is the first work in the literature to address this problem. Moreover, novel schemes for selecting appropriate edges for inserting cuts in the scheduled DFG, minimizing delay overhead associated with fault security are

proposed in the chapter. The detailed description of the proposed methodology is given in subsequent sections.

4.1 Problem Formulation

To explore the design space of a given DFG, and determine an optimal resource set

$$X_i = \{N(R_1), N(R_2), N(R_d)....N(R_D)\}$$

which satisfies conflicting user constraints and minimizes the overall cost.

The problem can be formulated as:

$$\textit{Find:}\ \text{an optimal}\ X_i$$

$$\textit{with minimum hybrid}\ \ \text{Cost}(P_T^{DMR}, T_E^{DMR})$$

$$\textit{subjected to:}\ P_T^{DMR} \le P_{cons} \qquad \textit{and} \qquad T_E^{DMR} \le T_{cons}\ \text{and}\ k_c\ \text{fault.}$$

Where, $N(R_d)$ is number of instances of a resource type 'd', P_T^{DMR} is power consumed by a fault secured DMR system, T_E^{DMR} is the delay of a fault secured DMR, T_{cons} and P_{cons} are the user specified execution delay and power constraints while k_c is the strength of the fault.

4.2 Proposed Methodology

4.2.1 Motivation

As already discussed in chapter 2, due to escalation in technology trends and density per unit area, there have been serious concerns related to security and faults in the devices. It can be said that the increase in density per unit area is negatively impacting the device and overall systems reliability by making it susceptible to transient fault or the Single Event Upset (SEU)

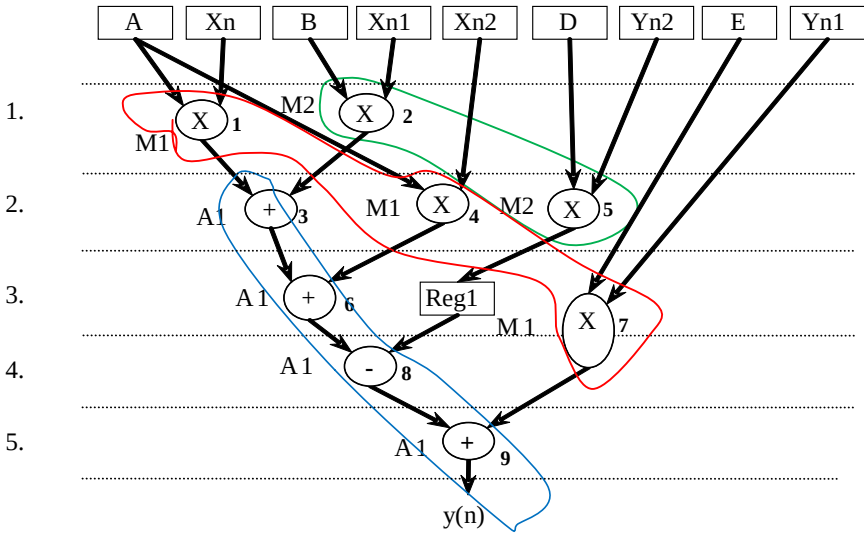


Figure 4.1 Scheduled Sequencing Graph with Data Registers

[48] leading to SET especially in space applications. In terrestrial applications, the SEU can also be caused by the alpha particles emitted from the impurities during IC packaging. Transient faults can be single or multi-cycle in nature. So, it is important to consider the strength of the transient fault (k_c) during specification as design objective while designing a system. This worst case transient pulse duration (k_c) value is used as a design specification (or fault constraint) before initiating the exploration of optimal k_c transient fault secured datapath during HLS [49].

Let us take an example to demonstrate the multi-cycle transient faults and its effect. Given a sample scheduled data flow graph (SDFG) in Figure 4.1. It shows a scheduled data flow graph of an application which uses two multipliers (M1, M2) and one adder (A1). Under standard conditions, the circuit undergoes a traditional computation, thereby generating a feasible error free output. However, if a transient fault occurs at any unit in the circuit due to particle strike, the corresponding output becomes erroneous, thereby affecting the entire circuit. For example, let us assume, if a two cycle fault occurs at Multiplier M1, when the state of the system is in control step 1. Then the error developed affects all the operations performed by the operator M1 during those two cycles. The span of the error affecting similar operators depends upon the nature of the transient fault (cycle duration). Thus, M1 incorrectly executes operation 1 at step 1 and also, operation 4 at step 2. But as soon as the system propagates to step 3, the effect of fault generated on M1 normalizes and the fault disappears. Hence, M1 operates correctly for the operation 7 at step 3. Such faults which occur once and then disappear are referred as transient faults. Once this fault occurs on a logic element of a system, the fault is associated as transient fault of the operator.

Therefore, it is important to consider transient fault (k_c) as a design metric while

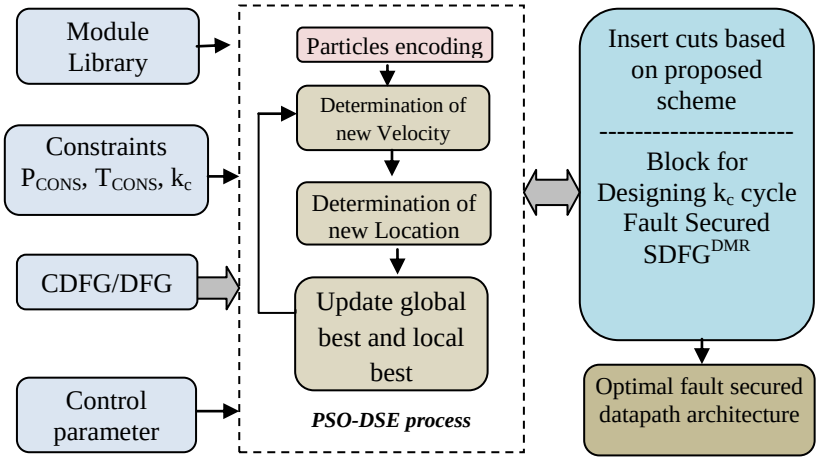


Figure 4.2 Block Diagram of Proposed Approach

designing a datapath during HLS. The framework to obtain a low cost k_c cycle transient fault secured datapath during behavioural level (reliability centric design) is explained in later sections.

4.2.2 Proposed Framework

This section presents a framework which handles transient faults with k_c strength and generates a low cost k_c cycle transient fault secured datapath during behavioural level. The framework of the proposed multi cycle fault secured PSO-DSE is shown in Figure 4.2. The input block comprises of module library, behavioral description of DFG, predefined user parametric constraints for power and delay as well as *k*-cycle fault constraint (k_c). Further, the input block for control parameters such as acceleration coefficient, inertia weight, swarm size and terminating criteria are used for regulating the exploration process. The proposed framework has a subunit for initialization/encoding of the particles. Each encoded particle is passed through the block for designing fault secured $SDFG^{DMR}$, which is responsible for converting an untimed DFG into a scheduled *kc* fault secured DMR system.

During this process appropriate cut (for checkpointing) is inserted based on proposed scheme (discussed in later sections) to optimize the delay overhead associated with fault security. Once the optimized $SDFG^{DMR}$ is built, it is subjected to fitness evaluation and the new velocity of each particle is determined for obtaining the new design solution (new location in the design space). The new design solutions obtained are again similarly subjected to the fault secured $SDFG^{DMR}$ block to convert it into a fault secured SDFG, followed by its fitness evaluation. Subsequently, the global best and local best solutions in the process are also updated. This process continues until the terminating criterion is reached yielding an optimal fault secured datapath architecture (or $SDFG^{DMR}$) which comprehensively satisfies the constraints of P_{cons}, T_{cons}, k_c and minimizes cost.

4.2.2.1 DSE Framework

The DSE framework used for generating a lost cost k_c cycle transient fault secured datapath during HLS is PSO-DSE. To solve the problem mentioned in this chapter, PSO as DSE framework is used to explore the design space. This is because PSO is considered more suitable than other Evolutionary Algorithms (EA) such as GA, hybrid GA and BFOA. This is due to reason that the later approaches do not provide enough flexible options for introducing stochasticity into the exploration process as well as is computationally more expensive.

Algorithm: PSO-DSE

Input- DFG, Module library, User Constraints Output- Optimal resource configuration

{

Read Library ( )

Read DFG ( )

Determine boundary constraints for power and execution time

If (($P_{min} > P_{cons} > P_{max} \parallel T_{min} > T_{cons} > T_{max}$)) *//checking validity of user constraints*

{

!! Show error message and request for valid constraints

}

Initialization (resource configuration, velocity)

For *i* =1 to **S** //S = # of particles)

{

$C_f(X_i) = f(Power, Execution\ time)$ *// calculate fitness of all particle*

}

//find best resource configuration that is the current global best resource configuration

// M = # of iteration

$X_{gb} = X_i[Min(C_{f_{lb1}}(X_1), C_{f_{lb1}}(X_2), C_{f_{lb1}}(X_3)....C_{f_{lbn}}(X_n))]$

While (**Z**)

{

For *i*=1 to *p* *//p* = (# of particle)

{

For *d*=1 to **D**

{

// determine new resource configuration and velocity for i^{th} particle and d^{th} dimension

$R_{d_i}^{+} = f(V_{d_i}^{+}, R_{d_i})$

IF ($-V_{d_i}^{max} > V_{d_i}^{+} > V_{d_i}^{max}$)

{

Perform Velocity Clamping ( )

}

//check boundary constraints outreach

IF ($\min(R_d) > R_{d_i}^{+} > \max(R_d)$)

{

Adaptive-end-terminal-perturbation ( )

}

} *// check for local best resource configuration*

IF ($C_f(X_i)(t) < C_{flb}(X_i)$)

{

$C_{flb}(X_i) = C_f(X_i)(t)$

$X_{lbi} = X_i(t)$

}

}

// determine new global best resource configuration

$X_{gb} = X_i[Min(C_{f_{lb1}}(X_1), C_{f_{lb1}}(X_2), C_{f_{lb1}}(X_3)....C_{f_{lbn}}(X_n))]$

Adaptive-Rotation-Mutation

$X_{gb} = X_i[Min(C_{f_{lb1}}(X_1), C_{f_{lb1}}(X_2), C_{f_{lb1}}(X_3)....C_{f_{lbn}}(X_n))]$

t++;

} *// end of while loop*;

Output optimal resource configuration

}

Figure 4.3 Pseudo code for PSO-DSE

Moreover, it has been proved in previous works [53, 54, 55, 56] that PSO is highly adaptable, provides faster convergence and offers higher chances of reaching optimal solution in less exploration time.

The pseudo code of the PSO-DSE approach is presented in Figure 4.3. While the proposed mapping is given as follows:

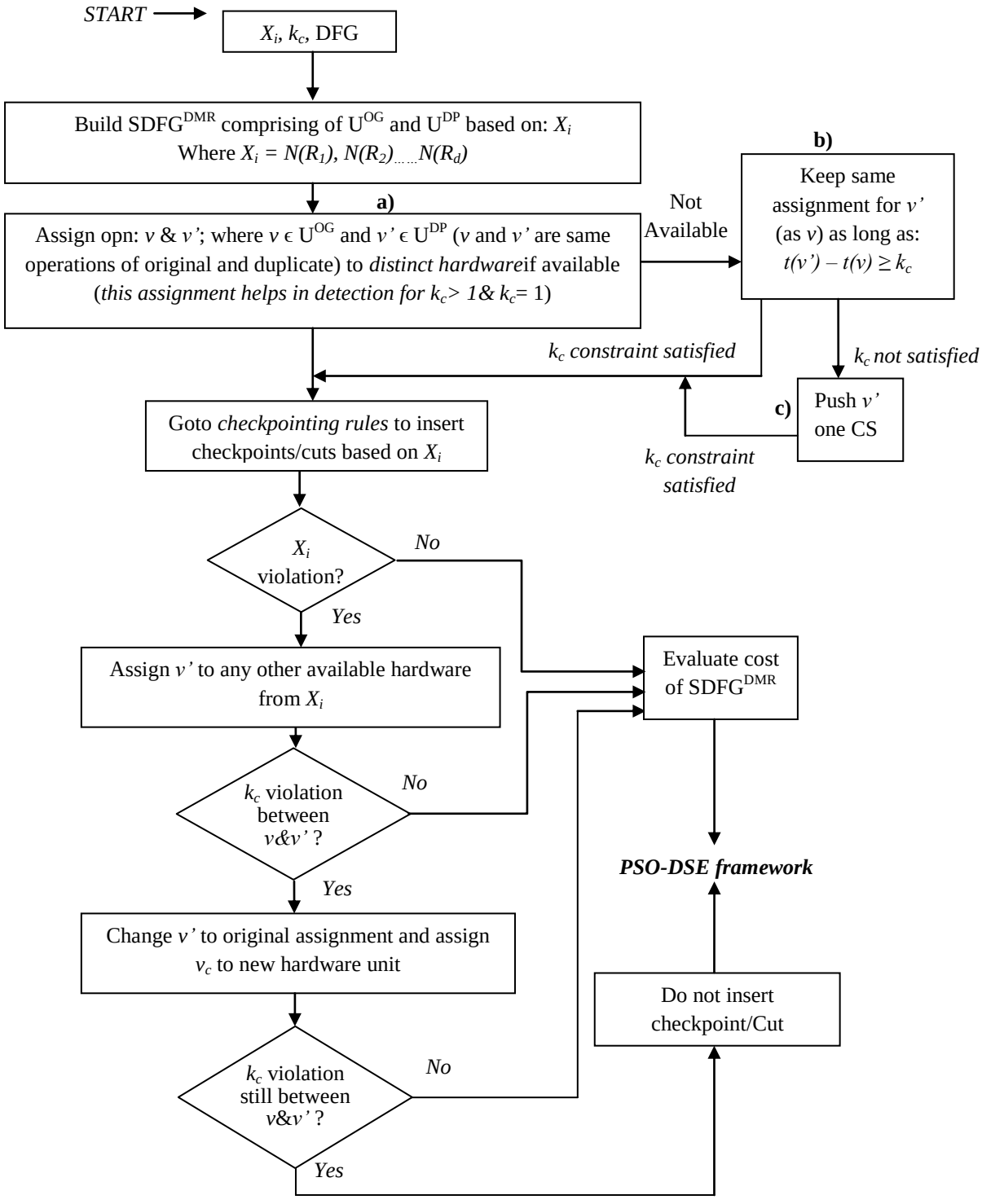


Figure 4.4 Algorithm for generating a *k-cycle* Fault Secured SDFG^{DMR}

Position of particle	→	Resource configuration
Velocity of particle	→	Exploration deviation/drift
Dimension	→	Number of Resource type

To transform the PSO into multi-objective DSE problem the position of a particle is represented by a set comprising of resource combination, total number of dimensions is represented by sum of the number of resource types, while the velocity of the particle in d^{th} dimension acts as a parameter that provides the drift during DSE.

The later subsection describes the proposed approach (based on particle swarm optimization [84, 85])

4.2.2.2 Assumptions of Proposed Algorithm

This subsection illustrates the assumptions which have been considered while designing the proposed PSO driven multi objective DSE for multi-cycle fault detectable datapath.

- Single fault model i.e. fault occurring at a single site in the circuit. Note: consideration of single fault model for transient faults is widely assumed and adopted in all related works such as [28, 30, 32]. Therefore, the proposed work on DSE of single/multi-cycle transient fault detectable datapath also uses the same assumption.
- The faults occur only at the hardware units and not at interconnecting wires.
- The system only handles the transient-faults and not permanent faults.
- The pair of unit in the DMR system has a comparator for error detection, whereby the comparators are considered fault detectable.

4.2.2.3 Proposed Algorithm for Design of k_c Fault Secured DMR System

The proposed methodology for designing k_c fault secured DMR system is shown in Figure 4.4. The proposed algorithm accepts the following as inputs: X_i (particle position denoting datapath configuration), the DFG, fault security constraint (k_c) indicating the strength of the fault and module library indicating the hardware units available for allocation. The output of the proposed algorithm is a valid k_c cycle fault secured scheduled DMR system that is iteratively feedback to the PSO-DSE framework for exploring the next design solution based on the fitness evaluation. The DMR system involves a $SDFG^{DMR}$, consisting of schedules of original unit (U^{OG}) and duplicate unit (U^{DP}). The pair of units is concurrently scheduled on the basis of ASAP scheduling using the user supplied resource constraints X_i and available dependency information of the nodes. After obtaining the scheduled DMR system, the

hardware allocation of both the units (U^{OG} and U^{DP}) is performed. Operations of the $SDFG^{DMR}$ system are allocated to hardware on the basis of fault security conditions (schemes) shown in Figure 4.4 (sub-block (a), (b) & (c)). Allocation of hardware to duplication unit of $SDFG^{DMR}$ without obeying the rules proposed in the algorithm may result in Transient Fault Hazards (TFH) between similar operations (of original and duplicate) assigned to same hardware unit i.e. TFH between similar operations belonging to a same hardware exists when:

$$t(v') - t(v) < k_c, \text{ where } v \in U^{OG} \text{ and } v' \in U^{DP.} \quad (4.1)$$

These hazards are resolved in the proposed algorithm by pushing the affected operation *v'* (and accordingly its successor) of the duplicate unit in later control steps, if assignment (allocation) rules (a) and (b) fails. The push is done such that the interval between $v \in U^{OG}$ and $v' \in U^{DP}$ is greater than (or equals to) k_c. This resolution of the TFH is done until the TFH of the whole DMR system is resolved, i.e. $SDFG^{DMR}$ obeys either of the fault security scheme ((a) or (b) or (c)) proposed in Figure 4.4. The blocks after the cut condition block are for handling the possible assignment violations that could occur in the modified fault secured $SDFG^{DMR}$ due to insertion of cut.

4.2.2.4 Demonstration of Proposed k_c fault secured DMR system

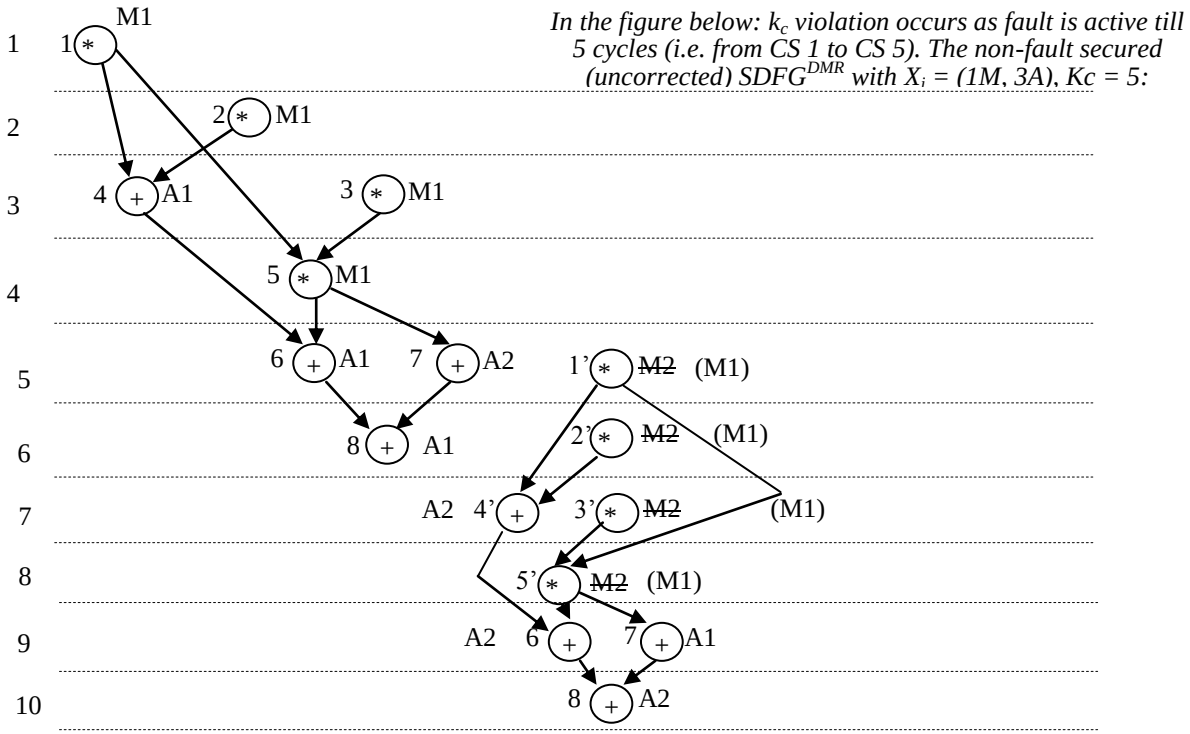


Figure 4.5 Uncorrected 5-cycle fault secured $SDFG^{DMR}$

Let us consider an example and demonstrate such condition. Consider the example shown in Figure 4.5. In the Figure 4.5 both the conditions, a) and b) mentioned in Figure. 4.4 are not satisfied. Therefore, condition c) (from Figure 4.4) is applied which successfully converts the non-fault secured SDFG^{DMR} into a k_c fault secured SDFG^{DMR}. Let us analyze the SDFG^{DMR} shown in Figure 4.5. Assuming the particle position X_i as: (1M, 3A) and k_c = 5, a fault secured SDFG^{DMR} has to be designed. Based on the availability of resources (as per X_i), only one multiplier is present for building the entire DMR system during scheduling. However, for enabling single fault model security feature for k_c> 1, distinct hardware assignment is necessary between similar operations in original and duplicate e.g. opn 1 and opn 1' cannot be assigned to same hardware units (according to condition (a)) (*Note*: in duplicate unit, hardware M2 is crossed to indicate that it is prohibited to use distinct hardware due to lack of availability specified in X_i). Next, according to the condition (b) of Figure 4.4 stated above, same hardware assignment may be kept if $t(v') - t(v) \geq k_c$, however $t(1) - t(1') \leq k_c$. *Similarly,* $t(2) - t(2') \leq k_c$, $t(3) - t(3') \leq k_c$ etc. Therefore as per proposed algorithm, condition (c) is applied which pushes $1' \in U^{DP}$ into next CS. Automatically, $2' \in U^{DP}$ is pushed down into next CS due to lack of available multiplier. Similarly other operations suffering from k_c violation is also pushed down in lower CS. The resultant corrected 5-cycle fault secured SDFG^{DMR} is

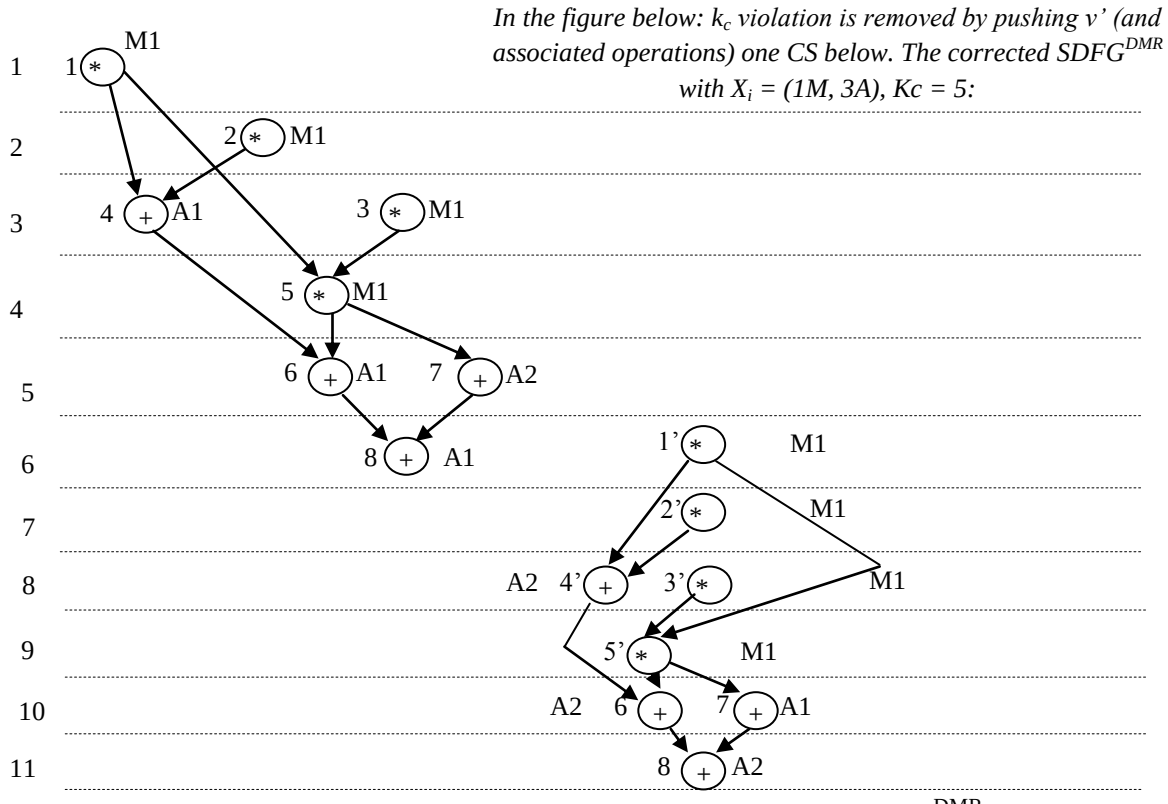


Figure 4.6 Corrected 5cycle Fault Secured SDFG^{DMR}

shown in Figure 4.6.

4.2.2.5 Proposed Schemes for Insertion of Appropriate Cuts in Corrected k_c SDFG^{DMR}

This section proposes schemes for inserting cuts in corrected k_c SDFG^{DMR}. Insertion of inapt cut to optimize delay overhead associated with fault security in most cases may not yield optimal solutions in the context of user constraints/budgets. In cutting some data edge of the duplicate unit is broken to remove the data dependency between operations thereby moving the dependent operation in upper CS (which then obtains its dependent output from similar operation in original unit). According to our algorithm (Figure 4.4) explained in previous section, insertion of appropriate cut is performed after the corrected k_c SDFG^{DMR} is obtained. However, insertion of appropriate cut (i.e. selecting the correct location/edge in the scheduled DMR) is not a trivial task. This is due to the fact that inapt cutting does not facilitate in optimizing delay occurring due to fault security, thereby resulting in resulting in longer delay increasing chances of possible violation of user delay constraint. Further, hit and trial process of inserting cut is highly time consuming thereby may increase exploration runtime beyond an acceptable range.

Motivating from these bottlenecks, schemes for insertion of appropriate cuts in corrected k_c SDFG^{DMR} are discussed later in this section. The schemes guarantee reduction of delay overhead due to fault security, if any possibility exists. The cuts (checkpoints) are inserted by traversing each node of corrected DMR schedule bottom to top searching for existence of any condition 3 (C3)/condition 4 (C4) illustrated later. This is because as described each of C3 or C4 is able to reduce delay overhead in fault secured DMR system.

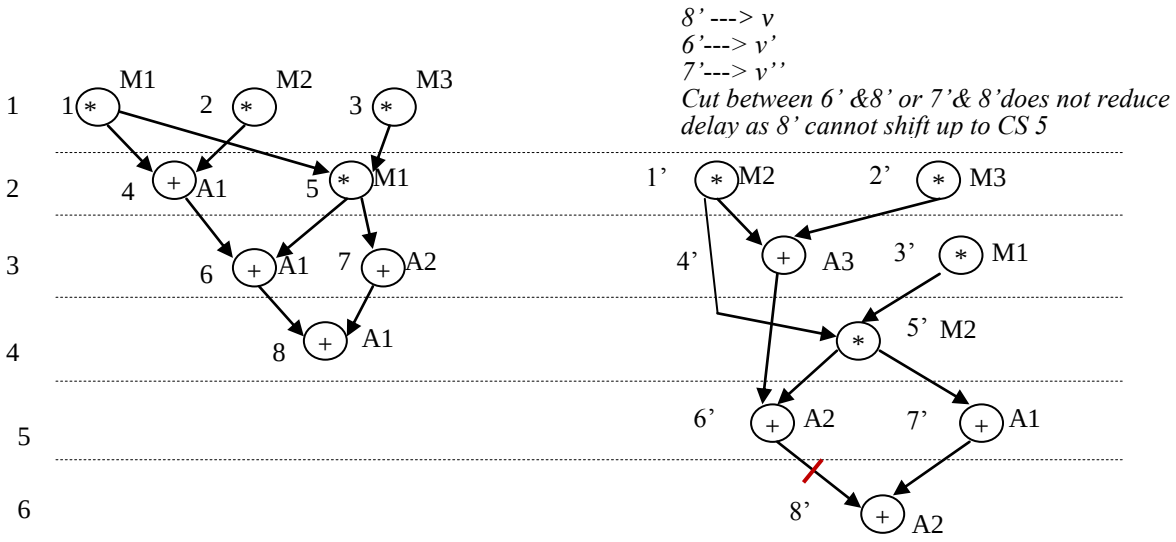


Figure 4.7 Example for C1

Note: Only single cuts (single additional checkpoint besides the regular checkpoint at the final output) are allowed in all cases in order to avoid excess hardware overhead (comparator/voter).

The Conditions are as follows:

- **C1***: If v' & v'' are the inputs to v, such that CS (v'') – CS (v') = 0 (i.e. v' & v'' are allocated in same CS),*

  then:

  No cut is allowed between *v*' & *v* or *v*'' & *v*.

  - **Demonstration of C1**

For example, consider a SDFG^{DMR} with X_i = (3M, 3A) shown in Figure 4.7. If a fault of k_c>1 (multi cycle) effects the system, then, a cut between 6' & 8' or 7'& 8'does not reduce delay as 8' cannot shift up to CS 5. Therefore, no cut is allowed between any nodes.

- **C2***: If v' & v'' of* <u>same</u> *operator type are the outputs of v, such that CS (v'') – CS (v') = 0 (i.e. v' & v'' are allocated* in same CS),

  then:

  No cut is allowed between *v*' & *v* or *v*'' & *v*.

  - **Demonstration of C2**

For example, consider a SDFG^{DMR} with X_i= (3M, 3A) shown in Figure 4.8. If a fault of k_c>1 (multi cycle) effects the system, then, cut between 5' &6' (or 5' & 7') does not reduce delay as shifting 6' (or 7') to CS5 does not benefit. Therefore, no cut is allowed between any nodes.

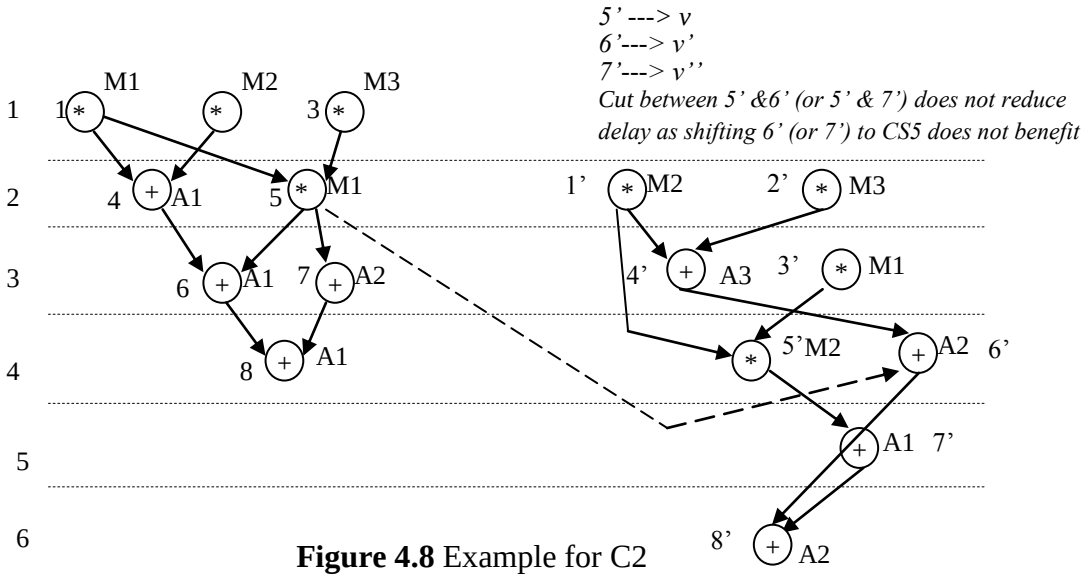


Figure 4.8 Example for C2

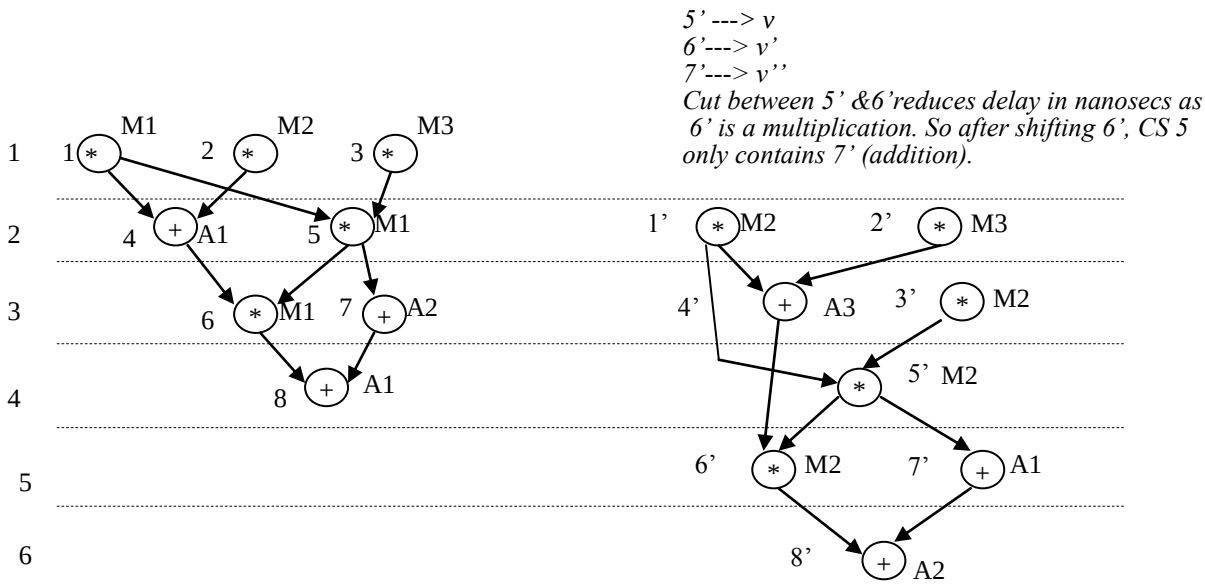


Figure 4.9 Example for C3 Before Cut

- **C3*:*** *If v' & v'' of* <u>*different*</u> *operator type (tv'' < tv' i.e. delay of v''< delay of v') are the outputs of v, such that CS (v'') – CS (v') = 0 (i.e. v' & v'' are allocated in same CS),*

  then:

  Cut is allowed between *v*'& *v*.

  - **Demonstration of C3**

For example, consider a SDFG^{DMR} with X_i= (3M, 3A) shown in Figure 4.9. If a fault of k_c>1 (multi cycle) effects the system in Figure 4.9 then, a cut between 5' & 6'reduces delay in nanosecs as 6' is a multiplication. So after shifting 6', CS 5 only contains 7' (addition).

- **C4:** *If v' & v'' are the inputs to v, such that v is a single operation in a CS in duplicate and CS (v') – CS (v'') > 1 (i.e. v' & v'' are greater than one CS apart),*

  then,

  Cut is allowed between *v* & *v*'.

  - **Demonstration of C4**

For example, consider a SDFG^{DMR} with X_i= (3M, 3A) shown in Figure 4.10. If a fault of k_c = 1 effects the system in Figure 4.10, then, as seen in Figure 4.11, an additional checkpoint is inserted at the output of 3'. Now, if a fault occurs at M3 affecting opn 3, then this faulty output affect opn 5'. The checkpoint at final output (comparing 8 and 8') is not able to detect the fault. However opn 3' remains fault secured. Therefore, the additional checkpoint inserted at output of opn 3' (comparing 3 and 3') detects the transient fault.

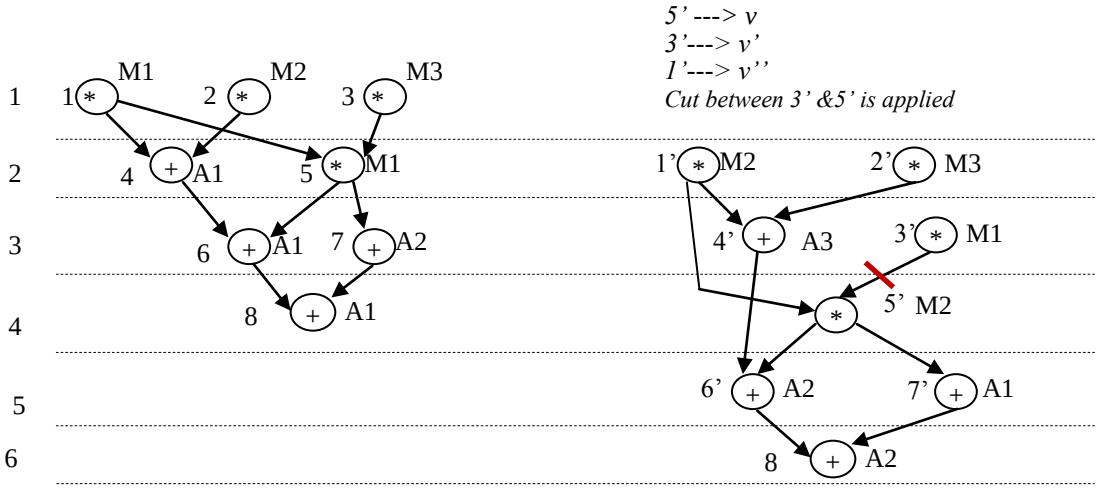


Figure 4.10 Example for C4 Before Cut

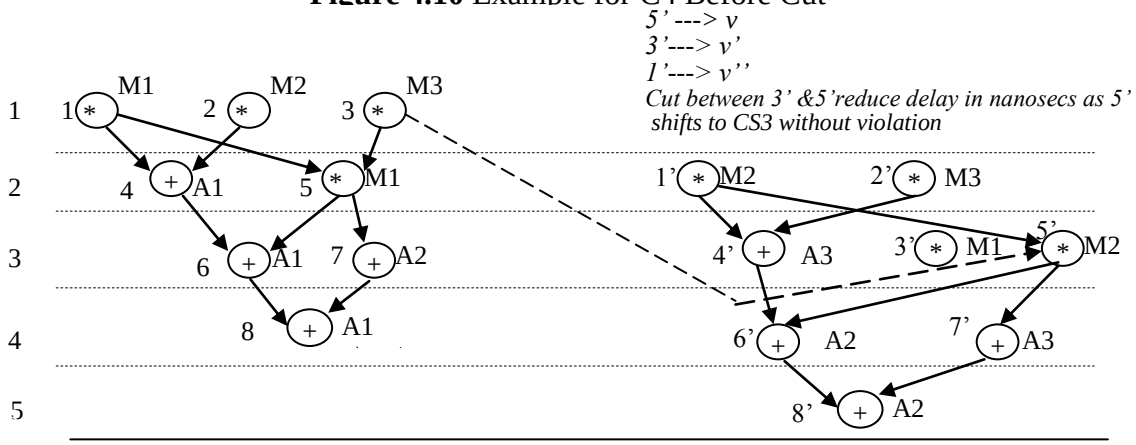


Figure 4.11 Example for Condition 4 After Cut

Note: While deciding for inserting cuts in duplicate, only C3 andC4 is checked. If either of the conditions (C3/C4) yield benefit, further checking is not carried out for that $SDFG^{DMR}$ i.e. if C3 is found to yield benefit, then C4 is not checked further.

4.3 Proposed Evaluation Models

For evaluation of a particle (or design point), the following models have been proposed.

4.3.1 Proposed Power Model

P_T^{DMR} of a resource set is represented in terms of Static Power (P_S^{DMR}) and Dynamic Power (P_D^{DMR}). 'P_T^{DMR}' is represented as:

$$P_T^{DMR} = P_S^{DMR} + P_D^{DMR} \tag{4.2}$$

P_S^{DMR} is a function of area of resources and leakage power per transistor. It can be formulated as:-

$$P_S^{DMR} = \sum_{d=1}^{D} (N(R_d).K(R_d).p_c \tag{4.3}$$

Where '$N(R_d)$' represents the number of instances of resource R_d. '$K(R_d)$' represents the area occupied by resource R_d, 'D' is the number of resources (FU's) and 'p_c' denotes the power dissipated per area unit (e.g. transistors).

While, the average dynamic power consumed by a resource configuration is a function of dynamic activity of the resources and can be given as:

$$P_D{}^{DMR} = \frac{E_{FU}^{DMR}}{T_E{}^{DMR}} \tag{4.4}$$

Where, E_{FU}^{DMR} is the total energy consumption of the resources in fault secured DMR system and $T_E{}^{DMR}$ is the total execution time of DMR system.

4.3.2 Proposed Execution Time (Delay) Model

For given 'D' functional resources the execution time is:

$$T_E{}^{DMR} = \sum_{c.s=1}^{n} Max(D(op_i),....D(op_n), D(op_{'i}),.....D(op_{'n})) \tag{4.5}$$

Where, $1 \le i \le n$ and '$1 \le$ '$i \le$ 'n. (Here, operations in original and duplicate are labelled as i and i' respectively; n and n' are maximum value of node); $D(opn)$ is the delay of operation 'n' while c.s is the control step.

4.3.3 Proposed fitness function

To assess the quality of explored configuration, the proposed fitness function motivated from the existing fitness function discussed in equation (3.8) is defined as:

$$C_f(X_i) = \phi_1 \frac{P_T{}^{DMR} - P_{cons}}{P_{\max}{}^{DMR}} + \phi_2 \frac{T_E{}^{DMR} - T_{cons}}{T_{\max}{}^{DMR}} \tag{4.6}$$

Where, $C_f(X_i)$ is the cost of particle with resource set X_i, ϕ_1 and ϕ_2 are the user defined weights for power and execution time parameters, $T_{\max}{}^{DMR}$ is the maximum execution time of a fault secured DMR system in design space while $P_{\max}{}^{DMR}$ is the maximum power of a fault secured DMR system in design space. The above function is a normalized penalty function where the cost value obtained, considers the power and execution time of DMR design. The normalization is achieved by dividing the value obtained by placing the maximum value in the denominator of the function.

4.4 Demonstration of PSO-DSE Methodology

4.4.1 User Specification

The goal of exploration problem is to simultaneously meet the user provided constraints for power and execution time and generate a low cost optimal k_c transient fault secured datapath. Therefore, the exploration process requires constraint inputs of power and execution time i.e. P_{cons} and T_{cons}, before initiation.

4.4.2 Boundary Constraints Check Module

This module checks whether the specified user constraint falls in the valid range of boundary limits. The following condition is checked for each parametric constraint specified:

1. Check: $P_{\min}^{DMR} > P_{cons} > P_{\max}^{DMR} \,||\, T_{\min}^{DMR} > T_{cons} > T_{\max}^{DMR}$

2. If the above condition is true then stop and correct the constraints.

   Else the above condition fails and goes to step 3.

3. Execute the initialization process of Module.

4.4.3 Particle Encoding/Initialization

The particles are initialized to uniformly cover the design space. The initialization is done on the basis of proposed scheme discussed in previous chapter. For example, in Figure 4.1 used for demonstration, as evident there are three types of resources (i.e. D= 3) viz. multiplier, adder and subtractor. Therefore with respect to the example, a particle position is given by: X_i

Adaptive end terminal perturbation
Input- Resource configuration which crosses the design space
Output- New value of resource configuration with in design space

//When R_{id} crosses the design space boundary

While (R_{id}< L)
{
    $R_{id} = R_{id}$ + Y
}
While(R_{id}> U)
{
    R_{id}=R_{id}-Y
}
/* where 'Y' is a random value between minimum resource constraints and maximum resource constraints.
'L' is lower boundary which means minimum resource value single instance.
'U' is the upper boundary which means maximum # of resources*/

Figure 4.12 Adaptive End Terminal Perturbation Algorithm

= *(N(mul), N(add), N(sub))*. Hence, using initialization as done in chapter 3, X_1 = (1, 1, 1); X_2 = (4, 2, 2); X_3= (2, 1, 1) can be obtained assuming maximum available multiplier resources: 4, and adder resources: 2 and subtractor resources: 2.

4.4.4 Initialization of Velocity, Acceleration Coefficient

Velocities of all particles are initialized to zero. Further, in order for the PSO-DSE to achieve convergence, it has been theoretically established before in [56] that the cognitive learning factor (b_1) and the social learning factor (b_2) can be initialized to any value between [1-2]. (Note: It is mathematically proved by authors in [51] that, when the value of '*b*' is pre-tuned between [1-2] and value of 'ω' between [0.9 to 0.1], the algorithm will converge for any given initial value of position and velocity). Therefore, the value of *b*=2 and ω linearly decreasing from 0.9 to 0.1has been used during experimentation.

4.4.5 Determination of Fitness and Update Local and Global Best Position

Based on the initialization of particles performed in section 4.4.3, the initial fitness of the individual values of power and execution time for all particles needs to be calculated.

```
Adaptive rotation mutation
Input – Local best resource configuration R^lb
Output – New mutated local best resource configuration
R^lb

For i=1 to p    // where p = Swarm size(#of particles)
{
        if (i%2==0) // Left Rotation
        {
                For  j=1 to D
                {
                        Temp = R_j
                        R_j = R_j +1
                        R_j+1 = temp
                        j++
                }
        }

        if(i%2==1)
        {
                For j=1 to D
                {
                        R_j = R_j ± X
                // X is a random number between [1,3]
                j++
                }
        }
        i++;
}
```

Figure 4.13 Adaptive Rotation Mutation Algorithm

4.4.6 Determination of Local and Global Best Position

Since in iteration 1, there is no previous local best position for an i^{th} particle (X_{lbi}) therefore the current position (X_i) assumes the value of X_{lbi}. The global best position (X_{gb}) of the population so far is determined using equation (15) as follows [31, 32]:

$$X_{gb} = X_i[Min(C_{f_{lb1}}(X_1), C_{f_{lb1}}(X_2), C_{f_{lb1}}(X_3)....C_{f_{lbn}}(X_n)] \quad (4.7)$$

Where, $C_{f_{lbi}}(X_i)$ is the local best fitness of particle 'X_i' and 'X_{gb}' indicates the global best particle position with minimum cost among all particle positions (X_1 …..X_n).

The PSO-DSE [31, 32] also comprises of mutation performed on the local best with probability P_m = 1.0 and adaptive algorithms to handle boundary overreach (shown in Figure 4.12) and mutation (shown in Figure 4.13) during exploration.

4.4.7 Determination of new position of each particle

Iteration process initiates at this step. According to PSO-DSE, each individual iteration computes new resource value of a particle X_i in d^{th} dimension through: $R_{d_i}^{+} = f(V_{d_i}^{+}, R_{d_i})$ which can be expanded as specified in equation 4.8 [54, 55, 57]:

$$R_{d_i}^{+} = R_{d_i} + V_{d_i}^{+} \quad (4.8)$$

Where, $R_{d_i}^{+}$ is the new resource value of particle X_i in d^{th} dimension and R_{d_i} is the previous resource value of particle X_i in d^{th} dimension; $V_{d_i}^{+}$ is the new velocity of particle X_i in d^{th} dimension (i.e. step length taken per unit time in d^{th} dimension) which is updated by equation (4.9) [54, 55, 57]:

$$V_{d_i}^{+} = \omega V_{d_i} + b_1 r_1 \left[R_{d_{lbi}} - R_{d_i}\right] + b_2 r_2 \left[R_{d_{gb}} - R_{d_i}\right] \quad (4.9)$$

Where, '$R_{d_{lbi}}$' is the resource value of X_{lbi} in d^{th} dimension and '$R_{d_{gb}}$'is the resource value of X_{gb} in d^{th} dimension.

Note- $X_{lbi} = \{R_{1_{lbi}}, R_{2_{lbi}} ... R_{D_{lbi}}\}$ *and* $X_{gb} = \{R_{1_{gb}}, R_{2_{gb}} ... R_{D_{gb}}\}$

4.5 Stopping Criteria (Z)

The proposed algorithm terminates when one of following condition holds true:

- When the maximum number of iteration have been exceeded (M = 100) or,
- S^1: When no improvement is seen in R_{gb} over '£' number of iteration. (£=10) or,
- S^2: If the population reaches to equilibrium state i.e. all particles velocity become zero ($V^+ = 0$).

Note: Results of the proposed method are given in chapter 8 section 8.2.

4.6 Summary

Over the years the DSE process has evolved where the requirements specified by the user have also convoluted, ranging from simple area-delay tradeoff in initial years to complex power-delay tradeoff in recent years. The approaches developed so far aimed at exploring the design space along with balancing some multi-conflicting issues during generation of the best possible solution. Solving the DSE problems with such objectives has no longer been sufficient now. There is demand for designs, which require ability to detect errors occurring due to transient faults. To achieve this, high reliable designs that have ability to detect errors are generated. This chapter presented an automated DSE approach to detect transient faults and generate an optimal fault secured datapath for data intensive applications based on user specified power-delay budget during HLS.

Chapter 5

Multi-Cycle Single Event Transient Fault Security Aware MO-DSE for Single loop CDFGs in HLS

The availability of faster devices is a feature of future technologies that induces major concerns to the fault detection community. For those technologies, even particles with modest Linear Energy Transfer (LET) values will produce transients lasting longer than the predicted cycle time of circuits. Therefore the technology evolution and LET of particle impact both plays a major role in inducing multi-cycle (*k-cycle*) transient fault (longer duration transient) in a device [58. 59]. Therefore, fault security should be considered early in the design cycle as design objective, besides traditional design objectives such as area and delay during DSE. Multi-cycle SET fault security aware Multi Objective Design Space Exploration (MO-DSE) for single loop CDFGs during behavioural synthesis has not received much attention in the literature. Solving the aforesaid problem in the context of CDFG is non-trivial. This is because it involves simultaneous generation of an optimal combination of multi-cycle fault secured datapath and loop unrolling factor satisfying conflicting user constraints (such as hardware area and delay).

This chapter solves the aforementioned problems and proposes a multi-cycle SET fault security aware MO-DSE methodology that explores an optimal combination of transient fault secured DMR datapath configuration and loop UF for CDFG. The proposed approach maintains a trade-off between hardware area and delay as user constraints during exploration process. The detail description of the proposed approach is given in subsequent sections in this chapter.

5.1 Problem Formulation

The problem can be formulated as:

$$\text{Find: Optimal } (X_i) = (R_x, UF_N)$$

$$\text{with minimum hybrid Cost } (A_T^{DMR}, T_E^{DMR})$$

$$\text{Subjected to: } A_T^{DMR} \leq A_{cons} \text{ and } \quad T_E^{DMR} \leq T_{cons} \text{ and } k_c \text{ transient fault constraint;}$$

where, 'X_i' is a set comprising of resource combination and UF formally represented as :

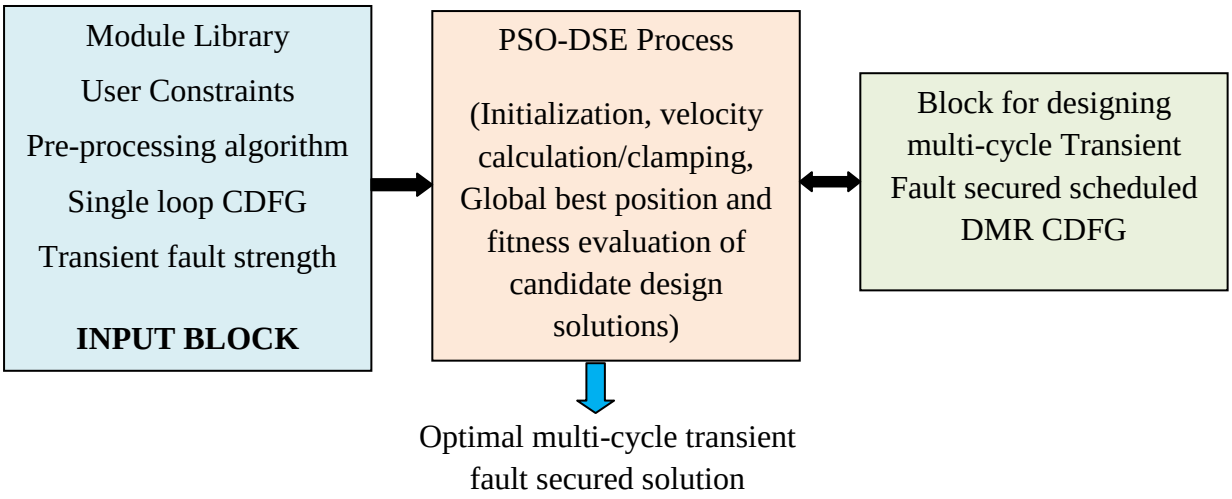


Figure 5.1 Proposed Multi-cycle Transient Fault Security Aware DSE During Behavioural Synthesis

$$X_i = (R_x, UF_N) = \{N(R_1), N(R_2), N(R_d)....N(R_D), UF_N\}$$

where, *'N(R_d)'* is the number of instances of resource type *'R_d'*; *'D'* is the total number of resource types; UF_N is the N^{th} unrolling factor; *'R_x'* is a candidate resource combination for optimal solution; UF_N is a candidate UF; *'A_T^{DMR}'* and *'T_E^{DMR}'* are the areas used by a fault secured DMR system and execution delay of a fault secured DMR system respectively; 'A_{cons}' and 'T_{cons}' is area and execution time constraints specified by the user.

5.2 The Proposed Framework and Mapping Process

The framework for exploration of an optimal multi-cycle transient fault secured solution is presented in Figure 5.1. To transform the PSO into multi-objective DSE problem the position of a particle is represented by a set comprising of resource combination and UF; total number of dimensions is represented by sum of the number of resource types and UF. Finally, the velocity of the particle in dth dimension acts as a parameter that provides the drift during DSE.

During exploration process the design points are evaluated. To evaluate the design points, model for execution time, model for area evaluation and model for cost (fitness) have been presented in the upcoming section.

5.3 Proposed Evaluation Models and Formulation

In the proposed PSO-DSE, each particle position represents a resource set (R_x) in the design space.

5.3.1 Proposed Model for Execution Time

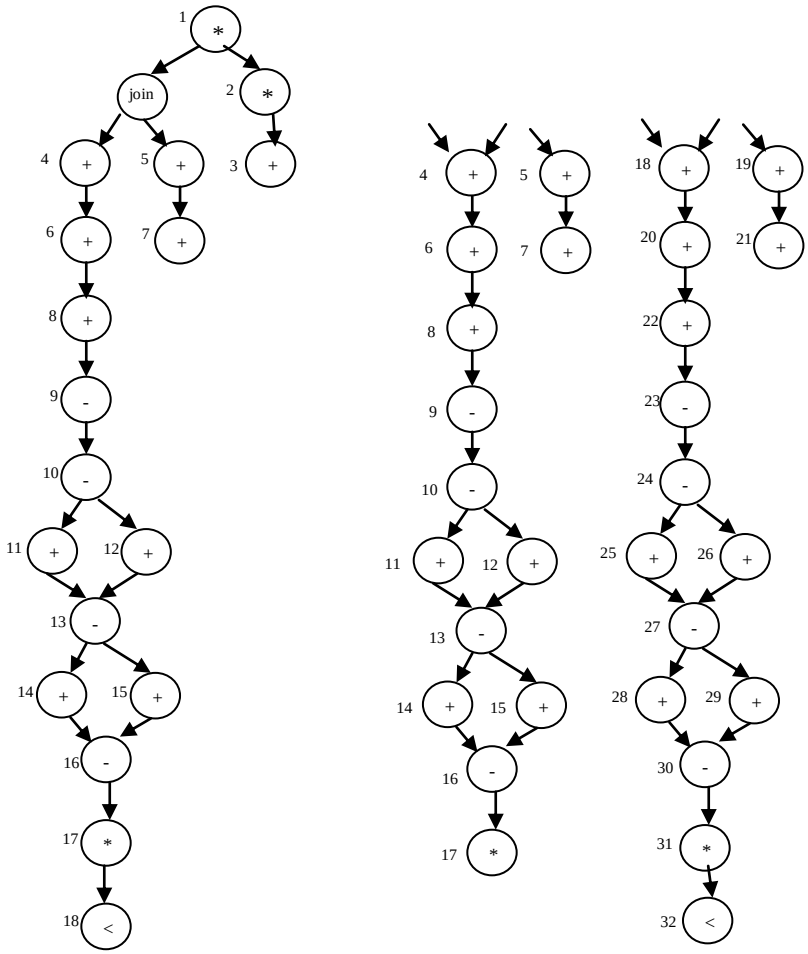


Figure 5.2(a) FFT Loop **Figure 5.2(b)** FFT Loop Unrolled Twice

In order to describe the formulation of proposed execution time (T_E^{DMR}) (function of loop unrolling factor) for a CDFG, an example of loop unrolling is used shown in Figure 5.2. Figure 5.2(a) shows the original loop part of CDFG for FFT and Figure 5.2(b) shows the same loop part unrolled twice. Figure 5.3 shows ASAP scheduled CDFG^{DMR} for FFT unrolled twice with resource constraint of 4(+), 2(*), 1(-) and 1(<); UF=2 and iteration count=4. It also shows the trailing loop part of the unrolled CDFG is not available for this case.

The generic execution delay model for a loop unrolled CDFG^{DMR} is shown as follows:

$$C_T^{DMR} = \left(C_{body}^{DMR} * \left[\frac{I}{UF} \right]^{floor} \right) + \left(I \bmod \quad UF \right) * C_{first}^{DMR}, \qquad (5.1)$$

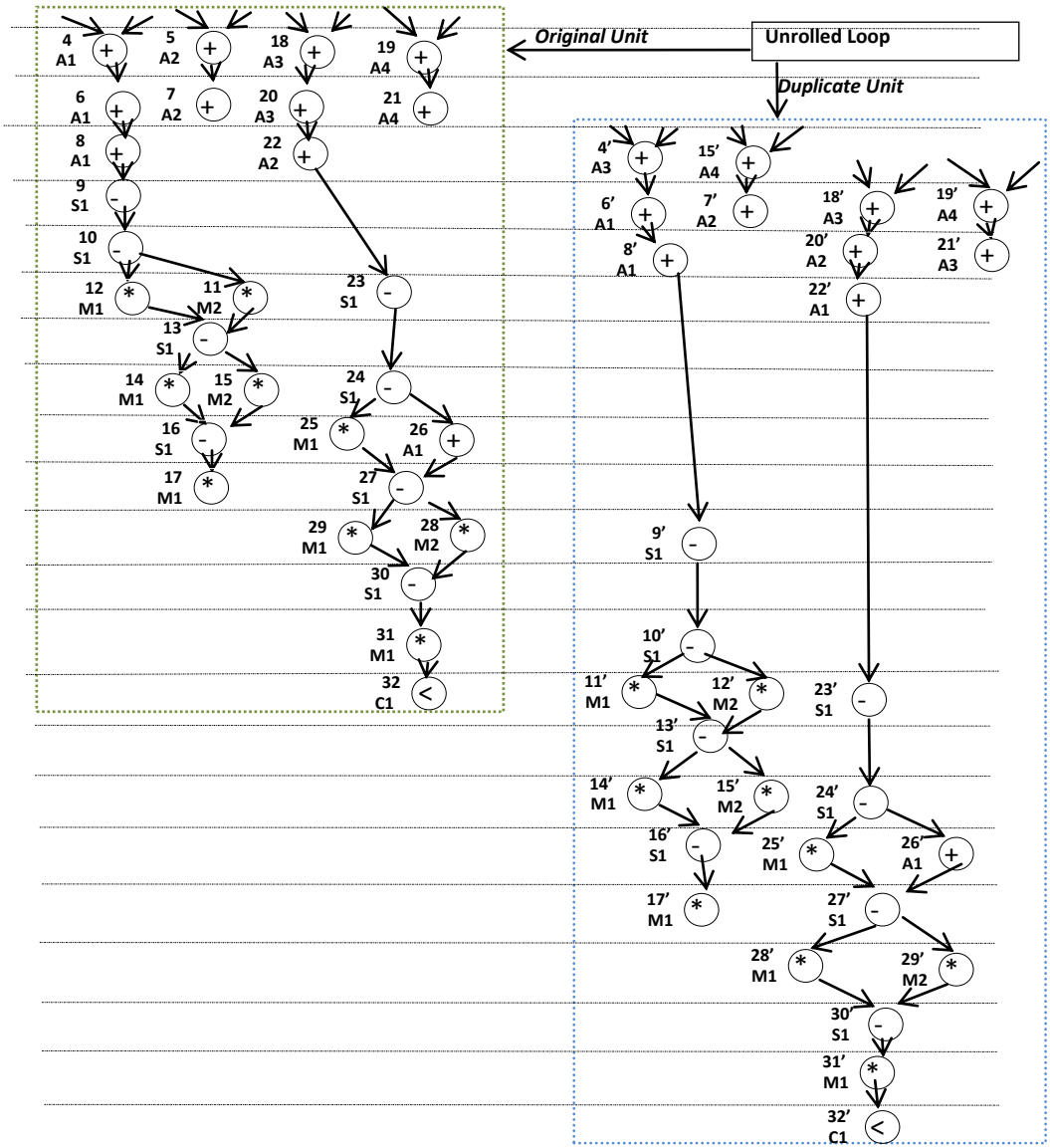


Figure 5.3 k_c Fault Secured SCDFG^{DMR} of FFT Loop body for Resource configuration (4(+),2(*),1(-),1(<), UF=2) at I=4 and k_c =2

Where, $\left(C_{body}^{DMR} * \left[\frac{I}{UF} \right]^{floor} \right)$ are total CS for unrolled loop, $(I \bmod UF) * C_{first}^{DMR}$ are total CS for sequential loop. C_T^{DMR} is total CS required to execute the loop of CDFG^{DMR} completely, C_{body}^{DMR} is the number of CS required to execute loop body of CDFG^{DMR} once, 'I' is the maximum number of iteration (loop count), C_{first}^{DMR} is number of CS required to execute first iteration of the CDFG^{DMR}. However, if the system design supports enough hardware instances such that sequential loops are possible to be fed to multiple hardware instances in parallel, then the total CSs for sequential loops from above equation (5.1) is, C_{first}^{DMR}. Finally, execution time for the system calculated as:

$$T_E^{DMR} = \Delta * C_T^{DMR} \quad (5.2)$$

where, 'Δ' is the delay of one CS in nanoseconds.

5.3.2 Proposed area model

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

$$A_T^{DMR} = \sum_{i=1}^{n} (N(R_i) * K(R_i)) \quad (5.3)$$

where, A_T^{DMR} is the total area of a DMR design, *'N(R_i)'* is the number of instances of resource type *'R_i'*.Note: The area component includes area due to functional resources, interconnect units (mux and demux), comparator (for error detection) as well as overhead incurred from internal buffering (during temporary storage of operation output in DMR scheduling).

5.3.3 Proposed Fitness Function

A fitness function which is a normalized penalty function where the cost value obtained considers the constraints for area and execution time of DMR design is proposed as follows:

$$C_f(X_i) = \phi_1 \frac{A_T^{DMR} - A_{cons}}{A_{\max}^{DMR}} + \phi_2 \frac{T_E^{DMR} - T_{cons}}{T_{\max}^{DMR}} \quad (5.4)$$

The above equation is motivated from the existing fitness function discussed in equation (3.8). Where, $C_f(X_i)$ is the cost of particle with resource set X_i; $T_{\max}^{DMR}$ is the maximum execution time of a fault secured DMR system in design space while $A_{\max}^{DMR}$ is the maximum area of a fault secured DMR system in design space.

5.4 Proposed Methodology

As seen from Figure 5.1 the input blocks comprise of module library, behavioural description of CDFG, predefined user parametric constraints for area and delay as well as pre-processing of unrolling factors. The pre-processing of UFs is explained in later sections. Wherein, the iteration count is provided as an input for pre-processing. Furthermore, the input block for control parameters comprises of acceleration coefficient, inertia weight, swarm size and terminating criteria which are used for regulating the PSO driven exploration process. PSO exploration process used has been explained in chapter 4.

The inputs required are further fed to the PSO-DSE block where, initialization and encoding of the particles, velocity and position up-gradation, velocity clamping and end-terminal perturbation, mutation and finally the updating of local best and global best positions

Pre-processing of unrolling factor

Input – value of 'I' (Total no. of loop iteration)

Output – screened set of unrolling factor (UF)

1 **Begin**

// Screening of UF//

2 **For** UF =2 to I **Do**

2.1 **IF** ((I mod UF < $\frac{UF}{2}$) &&

(UF <= I/2)) **Then**

//Add UF into the accepted UF list//

2.2 Accepted UF[k] = UF

2.3 k++

2.4 **End IF**

2.5 **End For**

3 **End**

Figure 5.4 Pre-processing of UF

Algorithm

1 Begin

2 For UF =2 to I do

//All U F are added into the accepted list until (I mod UF) < $\frac{UF}{2}$ *//*

2.1 **IF** (*(I mod UF)*< $\frac{UF}{2}$) **Then**

2.2 Terminate adding process jump to the end of the function

2.3 **End IF**

2.4 Accepted UF[k] =UF

2.5 k++

2.6 **End For**

3 **End**

Figure 5.5 Algorithm for Inclusion of Some Special UFs

are done. To evaluate the fitness of a particle, each encoded particle is passed through transient fault security block for designing a fault secured DMR Scheduled Control Data Flow Graph ($SCDFG^{DMR}$) which is responsible for converting an untimed CDFG into a scheduled k_c fault secured DMR Control Data Flow Graph ($CDFG^{DMR}$). After this process, appropriate cut for additional checkpointing is inserted based on proposed scheme to optimize delay overhead associated with fault security, followed by its fitness evaluation. Every such new design solutions (particle) obtained are again similarly convert it into a fault secured $SCDFG^{DMR}$. Subsequently, the global best and local best solutions in the PSO process are also updated. This process continues until the terminating criterion is reached yielding an optimal fault secured datapath architecture (or $SCDFG^{DMR}$) which comprehensively satisfies the constraints of A_{cons}, T_{cons}, k_c and minimizes cost.

5.4.1 Pre-processing of Unrolling Factors

A pre-processing of the unrolling factors is done to prune the design space. The pre-processing algorithm, shown in Figure 5.4, filters unfit UFs to create a list of viable solutions. The algorithm filters UFs with higher value since UFs with higher value offer minor improvement in the execution time and consume more power thereby increasing the overall cost of the solution. UFs which produce higher sequential loops are also removed from the

set. However, some special UFs are added which might be initially screened out in pre-processing to include good solutions. This is accomplished using the algorithm shown in Figure 5.5.

5.4.2 Proposed Initialization Process of Particles

After preprocessing step initialization of the particle take place. During initialization process particles position, are initialized as follows:

$$X_i = (N(R_1), (N(R_2),..(N(R_d).. (N(R_{D-1}),UF)$$

the initialization of particles is such that it uniformly covers the entire design space

$$X_1 = (min(R_1), min(R_2),.. min(R_{D-1}),min(UF)) \quad (5.5)$$

$$X_2 = (max(R_1), max(R_2),.. max(R_{D-1}),max(UF) \quad (5.6)$$

$$X_3=(((min(R_1)+max(R_1))/2..,((min(R_{D-1})+max(R_{D-1}))/2,max(UF)/2) \quad (5.7)$$

Rest of the particle positions($X_4 \ldots X_n$) are initialized with random values between minimum and maximum values of resources and UF. Since, an optimal design solution to a multi-objective exploration problem will always lies between the maximum parallel and serial implementation of the application. Therefore, keeping in mind the above, X_1 is represented by the serial implementation, X_2 by parallel implementation, X_3 with the mid value between serial and parallel implementation and X_4-X_n scattered anywhere between serial and parallel implementation.

5.4.3 Initialization of Velocity, Acceleration Coefficient and Inertia Weight

The details about velocity, acceleration coefficient and inertias weight initialization have already been discussed in section 4.4.4 of chapter 4.

5.4.4 Assumptions of Proposed Algorithm

This subsection illustrates the assumptions which have been considered while designing the proposed PSO driven multi objective DSE for multi-cycle fault detectable datapath.

- Single fault model i.e. fault occurring at a single site in the circuit. Note: consideration of single fault model for transient faults is widely assumed and adopted in all related works such as [28, 30, 32]. Therefore, the proposed work on DSE of single/multi-cycle transient fault detectable datapath also uses the same assumption.
- The faults occur only at the hardware units and not at interconnecting wires.
- The system only handles the transient-faults and not permanent faults.
- The pair of unit in the DMR system has a comparator for error detection, whereby the comparators are considered fault detectable.

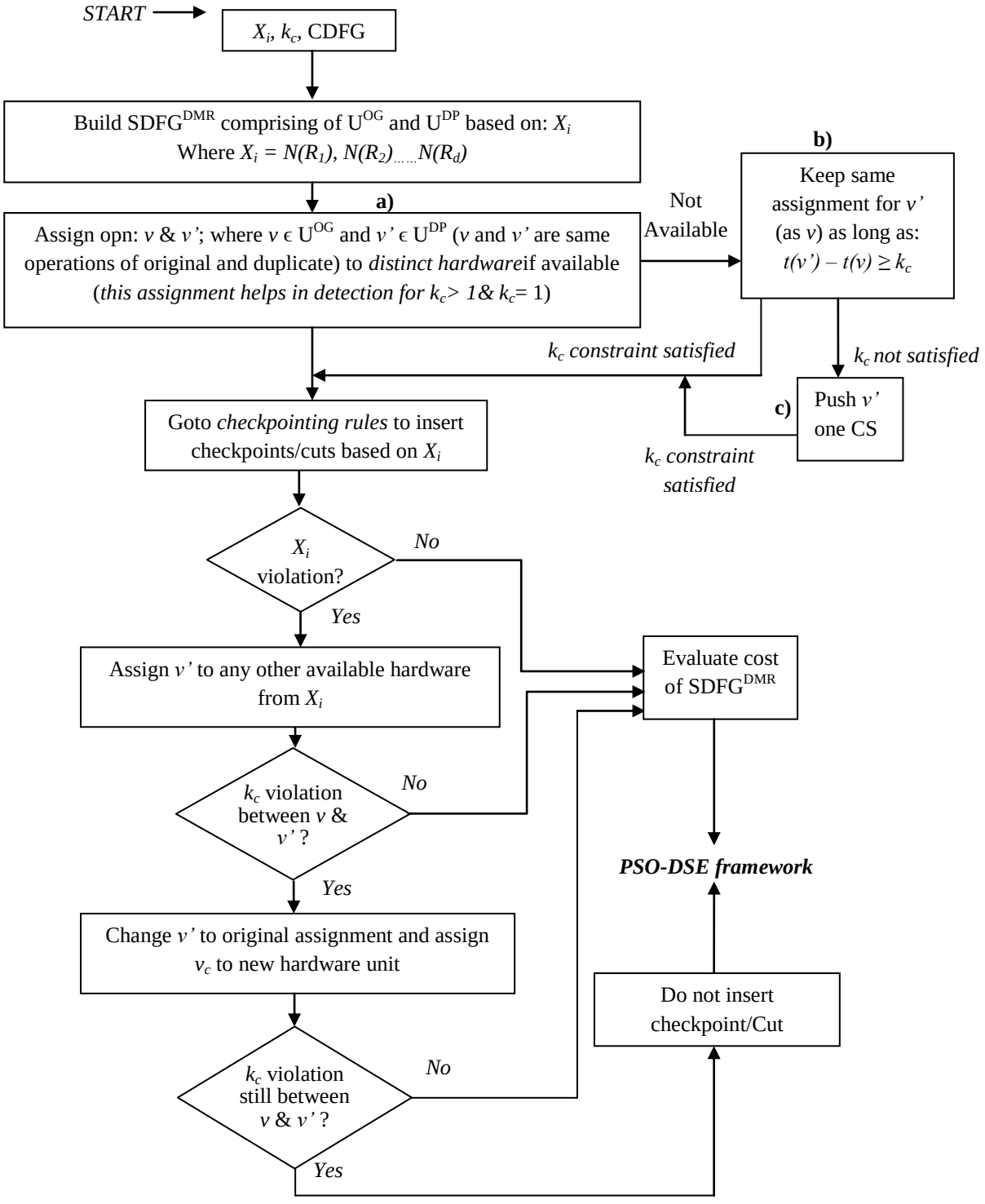


Figure 5.6 Algorithm for Generating a k_c Fault Secured SCDFG^{DMR}

5.4.5 Proposed Algorithm for Design of k_c Fault Secured DMR system

The proposed methodology for designing k_c fault secured DMR system is shown in Figure 5.6. The proposed algorithm accepts the following as inputs: X_i (particle position denoting datapath configuration), the CDFG, fault security constraint (k_c) indicating the strength of the fault and module library indicating the hardware units available for allocation. The output of

the proposed algorithm is a valid k_c cycle fault secured scheduled DMR system that is iteratively feedback to the PSO-DSE framework for exploring the next design solution based on the fitness evaluation. The DMR system involves a $SCDFG^{DMR}$, consisting of schedules of U^{OG} and U^{DP}. The pair of units is concurrently scheduled on the basis of ASAP scheduling using the user supplied resource constraints X_i and available dependency information of the nodes. After obtaining the scheduled DMR system, the hardware allocation of both the units (U^{OG} and U^{DP}) is performed. Operations of the $SCDFG^{DMR}$ system are allocated to hardware on the basis of fault security conditions (schemes) shown in Figure 5.6 (sub-block (a), (b) & (c)). Allocation of hardware to duplication unit of $SCDFG^{DMR}$ without obeying the rules proposed in the algorithm may result in TFH between similar operations (of original and duplicate) assigned to same hardware unit i.e. TFH between similar operations belonging to a same hardware exists when:

$$t(v') - t(v) < k_c, \text{ where } v \in U^{OG} \text{ and } v' \in U^{DP.} \quad (5.8)$$

These hazards are resolved in the proposed algorithm by pushing the affected operation v' (and accordingly its successor) of the duplicate unit in later control steps, if assignment (allocation) rules (a) and (b) fails. The push is done such that the interval between $v \in U^{OG}$ and $v' \in U^{DP}$ is greater than (or equals to) k_c. This resolution of the TFH is done until the TFH of the whole DMR system is resolved, i.e. $SDFG^{DMR}$ obeys either of the fault security scheme ((a) or (b) or (c)) proposed in Figure 5.6. The blocks after the cut condition block are for handling the possible assignment violations that could occur in the modified fault secured $SCDFG^{DMR}$ due to insertion of cut.

The cut conditions employed in order to reduce the additional execution time delay incurred due to shifting of operations in later control steps have been discussed in section 4.2.2.5 of chapter 4.

5.4.6 Determine Global Best Position

The global best position of the population is determined as follows:

$$X_{gb} = X_i[Min(C_{f_{lb1}}(X_1), C_{f_{lb1}}(X_2), C_{f_{lb1}}(X_3)....C_{f_{lbn}}(X_n))] \quad (5.9)$$

The global best particle position has minimum cost among all particle positions (X_1 …..X_n).

5.4.7 Determination of New Position of Each Particle

Iteration process initiates at this step. According to PSO-DSE, each individual iteration computes new resource value of a particle X_i in d^{th} dimension through: $R_{d_i}^{+} = f(V_{d_i}^{+}, R_{d_i})$ which can be expanded as specified in equation 4.8 [54, 55, 57]:

$$R_{d_i}^{+} = R_{d_i} + V_{d_i}^{+} \quad (5.10)$$

Where, $R_{d_i}^{+}$ is the new resource value of particle X_i in d^{th} dimension and R_{d_i} is the previous resource value of particle X_i in d^{th} dimension; $V_{d_i}^{+}$ is the new velocity of particle X_i in d^{th} dimension (i.e. step length taken per unit time in d^{th} dimension) which is updated by equation (5.11) [54, 55, 57]:

$$V_{d_i}^{+} = \omega V_{d_i} + b_1 r_1 \left[R_{d_{lbi}} - R_{d_i} \right] + b_2 r_2 \left[R_{d_{gb}} - R_{d_i} \right] \quad (5.11)$$

Where, ' $R_{d_{lbi}}$ ' is the resource value of X_{lbi} in d^{th} dimension and ' $R_{d_{gb}}$ 'is the resource value of X_{gb} in d^{th} dimension.

Note- $X_{lbi} = \{R_{1_{lbi}}, R_{2_{lbi}} \ldots R_{D_{lbi}}\}$ *and* $X_{gb} = \{R_{1_{gb}}, R_{2_{gb}} \ldots R_{D_{gb}}\}$

5.4.8 Adaptive end Terminal Perturbation and Adaptive Rotation Mutation

To handle boundary outreach problem during exploration process we propose adaptive end terminal perturbation, described in chapter 4.

In order to increase variation and diversity, mutation is performed on all the local best position of each particles with probability M_p =1.0 using *Adaptive rotation mutation* described in chapter 4.

5.5 Stopping Condition (Z)

The proposed algorithm terminates when the maximum number of iterations exceeds 100, or when no improvement is visible in X_{gb} over '£' number of iteration. (£=10). Details on stopping criteria have already been discussed in chapter 4.

Note: Results of the proposed solution are explained in chapter 8 section 8.3.

5.6 Summary

This chapter presented a novel multi-cycle SET fault security aware MO-DSE approach which explores an optimal transient fault secured datapath configuration and loop UF for control intensive applications. The datapath generated abides by the user specified area-delay constraints during exploration process.

Chapter 6

Bacterial Foraging Driven Exploration of Multi Cycle Fault Tolerant Datapath based on Power-Performance Tradeoff in High Level Synthesis

Due to recent advancements in technology, the idea of packing millions of transistors on a single chip has become more feasible. Technology evolution and impact of particle both plays a major role in inducing multi-cycle transient fault (longer duration transient) in a device. However, designing an optimized multi-cycle fault tolerant system is non-trivial. A multi-cycle fault tolerant system is a design that is, resilient against transient fault eminating due to SETs. For the current and future technology transient faults can span more than one clock cycle resulting in its multi cycle nature. Therefore, a multi-cycle fault tolerant system not only has capability to detct a transient fault but also to recover from it.

This chapter presents a novel multi-cycle fault tolerant DSE approach based on power-performance tradeoff during HLS. To the best of the authors' belief, this is the first effort to solve this problem in the literature so far. The proposed methodology is based on an adaptive BFOA that allows reaching the true Pareto optimal curve. The chapter also discusses about a novel DMR with equivalent circuit scheme that performs the equivalent function of extracting the correct output.

6.1 Problem Formulation

To explore the design space of a given DFG, and determine an optimal resource set

$$X_i = \{N(R_1), N(R_2), N(R_d)....N(R_D)\}$$

which satisfies conflicting user constraints and minimizes the overall cost.

The problem can be formulated as:

Find: an optimal X_i

with minimum hybrid $\mathrm{Cost}(P_T^{DMR}, T_E^{DMR})$

subjected to: $P_T^{DMR} \le P_{cons}$ *and* $T_E^{DMR} \le T_{cons}$ and k_c fault.

Where, $N(R_d)$ is number of instances of a resource type 'd', P_T^{DMR} is power consumed by a fault tolerant DMR system, T_E^{DMR} is the delay of a fault tolerant DMR, T_{cons} and P_{cons} are the user specified execution delay and power constraints while k_c is the strength of the fault.

6.2 Proposed Framework

The framework of a fault tolerant DSE scheme has been shown in Figure 6.1. A BFOA driven DSE framework is used for exploration of designs. The input block comprises of: module library, behavioral description of DFG, predefined user parametric constraints for power and time execution as well as k_c. Further, the control parameters such as N_c, N_{ed}, p explained in chapter 3 are used for regulating the BFOA driven exploration process. The proposed framework has a subunit for initialization/ encoding of bacteria. The encoded

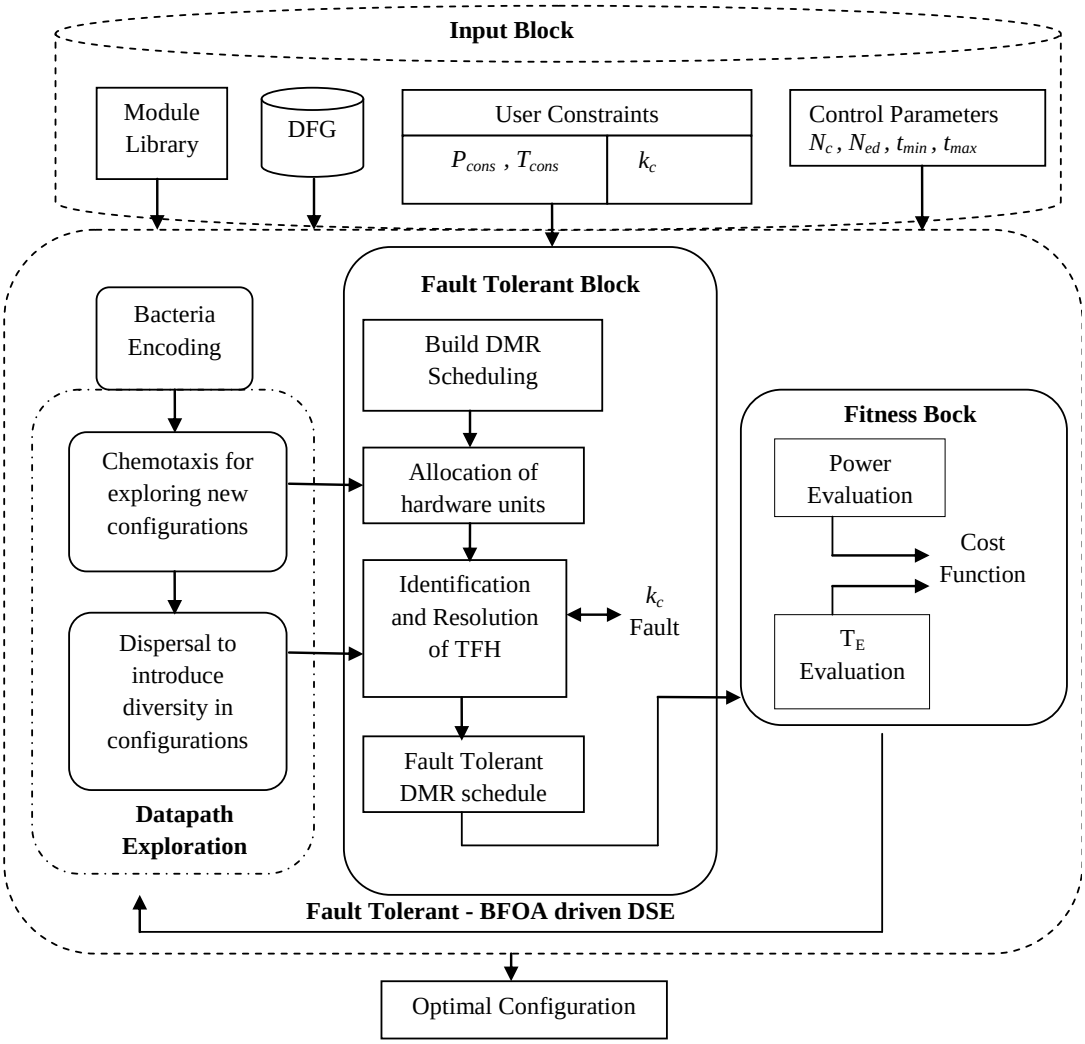


Figure 6.1 Proposed Multi Objective Multi Cycle Fault Tolerant BFOA-DSE Approach

bacteria are then subjected to chemotaxis and dispersal algorithms during the DSE process to explore new and diverse resource configurations. The solutions generated through the algorithms are fed into the fault tolerant block for converting into a k_c error-correctable design (masking the fault) by evaluating the DMR schedule on the basis of k_c fault behavior. A DMR design is obtained corresponding to each bacterium solution during DSE where the TFH due to k_c fault are identified and resolved subsequently to obtain a fault tolerant DMR schedule through the proposed algorithm. The obtained fault tolerant DMR schedule is passed into the fitness block to determine the cost of the fault-tolerant solutions generated. This process continues through the proposed BFOA-DSE framework to generate an optimal fault tolerant DMR system that comprehensively satisfies P_{cons}, T_{cons}, k_c as well as minimizes the hybrid cost.

6.2.1 Framework for DSE

BFOA DSE framework generates intermediate solutions during exploration that are fed into the proposed multi cycle fault tolerant algorithm. The multi cycle fault tolerant algorithm uses these explored solutions to convert them into a fault tolerant DMR schedule considering the user specified power budget and performance requirement. This process continues until an optimal solution i.e. a fault tolerant DMR system that comprehensively minimizes and satisfies the multi objective power and execution time constraint. The proposed multi cycle fault tolerant algorithm is described in next section.

6.2.2 Proposed DMR System for k_c Fault Tolerance

6.2.2.1 Assumptions of Proposed Algorithm

In the proposed work, following assumptions have been considered while designing the proposed BFOA driven multi objective DSE for multi-cycle fault tolerant datapath.

- Single fault model i.e. fault occurring at a single site in the circuit.
- Faults occurring only in the original unit of the DMR design.
- The pair of unit in the DMR system has a comparator for error detection, whereby the comparators are considered fault tolerant.
- The system only handles the transient-faults and not permanent faults.
- The faults occur only at the hardware units and not at interconnecting wires.

6.2.2.2 Proposed Multi Cycle Fault Tolerance (MCFT) Algorithm

An explored fault tolerant DMR system for dealing with k_c faults, based on user specified power budget and execution time constraint has been proposed in Figure 6.2. The DMR

Inputs: DFG, X_i, k_c, D_C
Output: Fault tolerant SDFG^{DMR}
Begin

1. Build a DMR scheduling graph (SDFG^{DMR}) comprising of $U^{OG} + U^{DP}$;
   $SDFG^{DMR} = U^{OG} + U^{DP}$
   Subjected to: a. Constraint X_i; where $X_i = (R_{1,} R_{2,}....R_{d-1,} R_D)$
   b. Constraint D_c
2. Identify the critical path (p_{cri}) from both U^{OG} and U^{DP}.
3. Allocate opn (o_i) of $p_{cri} \epsilon U^{OG}$ and $p_{cri} \epsilon U^{DP}$ to distinct operators (hardware units).
4. Allocate opn (o_i) of non-critical paths by keeping assigned operations to similar operators in both U^{OG} and U^{DP} if available.

   Loop 1:
   do
   {
5. Identify the *transient fault hazards* (TFH), if any, and prepare a list *L[k]* which indicates the *transient fault hazards* (TFH) between v and v' of similar operators in the current scheduling ($SDFG^{DMR}$) such that:
   $v \epsilon U^{OG}$
   $v' \epsilon U^{DP}$
   Loop 2:
   do
   {
6. Select TFH to be resolved from *L[k]*.
7. Push v' and its successor's ϵU^{DP} in lower CS such that:
   a) $t(v')- t(v) => k_c$ (i.e. interval between v and v' is greater than k_c)
   b) Constraints X_i and D_c satisfies.
8. k++;
9. **Goto**: Step 6

   }**while** (*L(k)*!= *φ);*
   Goto: Step 5

   }**while**(all $t(v')- t(v) => k_c$ in $SDFG^{DMR}$ (i.e. no *TFH* exists in the $SDFG^{DMR}$))
10. Are similar operators of U^{DP} used in U^{OG} in subsequent control steps within range of k_c cycle? If reused then, adjust the conflicting operations of U^{DP}

Loop 1: Prepares a list *L[k]* containing TFH in intermediate schedules of DMR.

Loop 2: Iterates to resolve the successive TFH from *L[k]*.

Figure 6.2 Pseudo code for Multi Cycle Fault Tolerant Algorithm

system involves a $SDFG^{DMR}$, consisting of schedules of U^{OG} and U^{DP}. The pair of units is concurrently scheduled on the basis of ASAP scheduling using the user supplied resource constraints $X_{i,}$ and available dependency information (D_c) of the nodes. After obtaining the scheduled DMR system, the critical paths (p_{cri}) from the units (both U^{OG} and U^{DP}) are identified. Operations of the $SDFG^{DMR}$ system are allocated to operators on the basis of following scheme:-

i. Allocate opn (o_i) of $p_{cri} \epsilon U^{OG}$ and $p_{cri} \epsilon U^{DP}$ to distinct operators (hardware units).

ii. Allocate the remaining operations by:

- If opn(o_i) ϵU^{OG} , then assign operator on the basis of availability

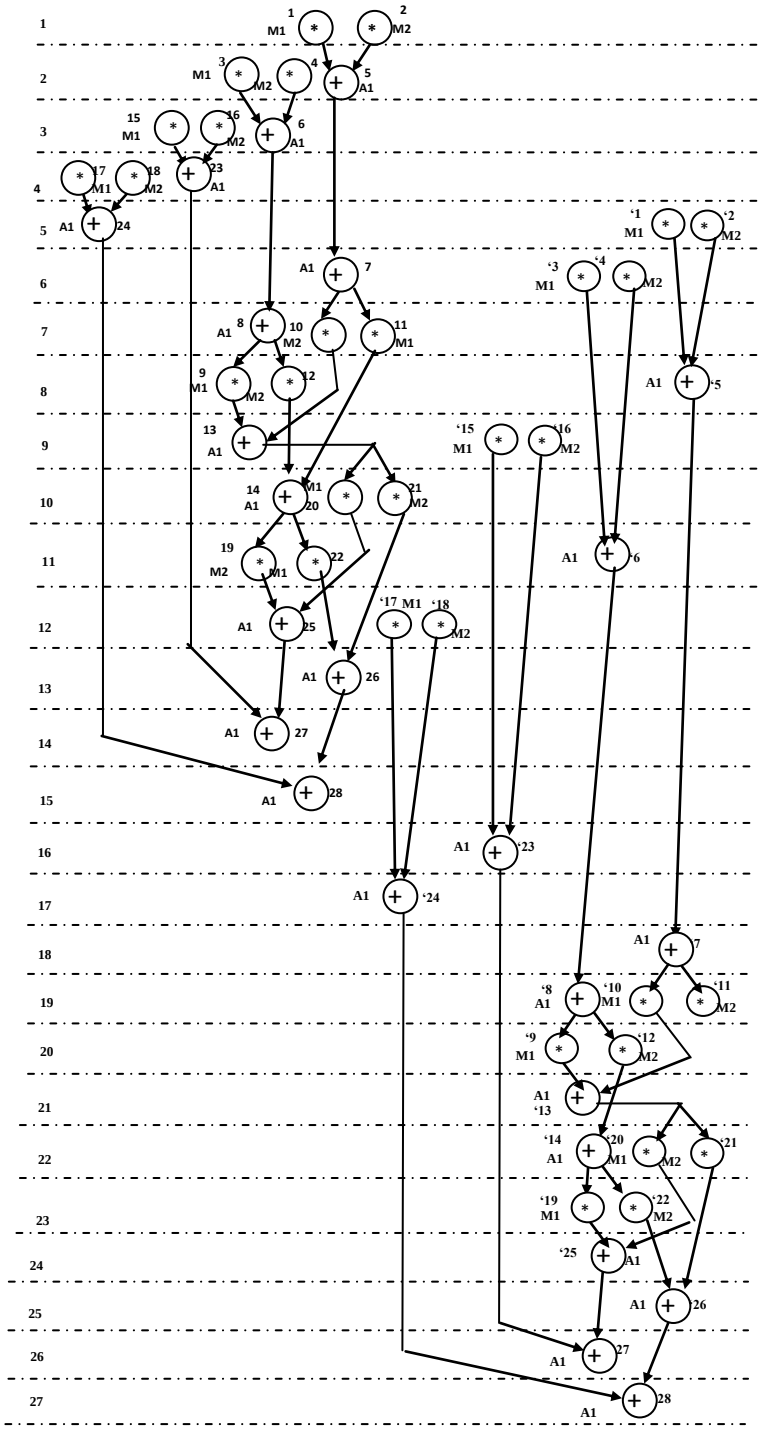


Figure 6.3 SDFG^{DMR} of ARF with X_i = 1(+), 2(*)

- If opn(o_i) ϵ U^{DP}, then assign operations to similar operators as in U^{OG}, if available to enable resource sharing and reduction of usage of extra operators.

NOTE: This is because during designing fault tolerant datapath (for multi cycle faults) assigning to distinct hardware operator sin duplicate unit does not assist in masking the fault in duplicate. This is owing to the reason that faults anyways affects some other operation

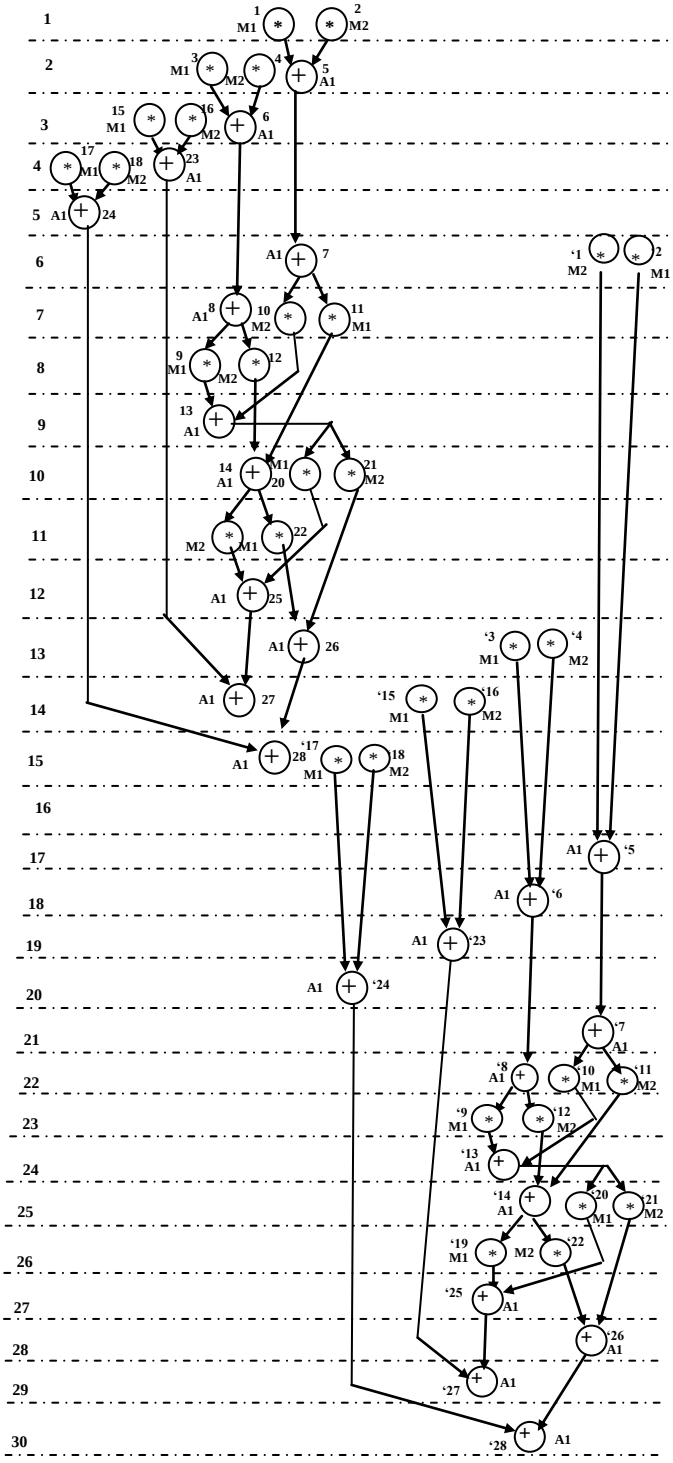


Figure 6.4 Intermediate Fault Tolerant SDFG^{DMR} of ARF for $k_c = 2$

assigned to the same operator in duplicate. Assigning to distinct hardware only assists in fault security (i.e. detection).

Once the assignment of operators is done, the behavior of the system due to *k*-cycle fault is observed by identifying the TFH between any operations belonging to an operator. The TFH between any operations belonging to an operator exists when :

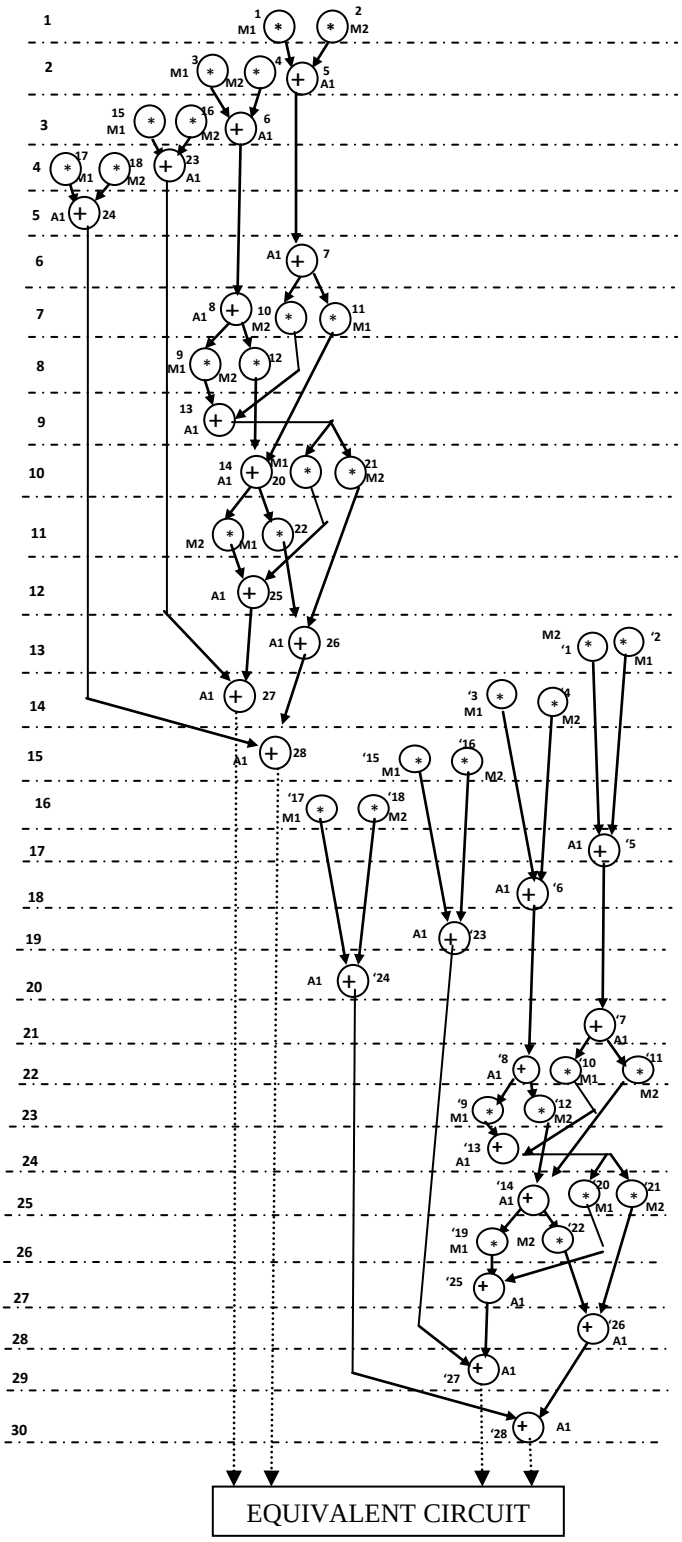


Figure 6.5 Fault Tolerant SDFG^{DMR} of ARF for $k_c = 2$

$$t(v') - t(v) < k_c, \text{ where } v \in U^{OG} \text{ and } v' \in U^{DP.} \quad (6.1)$$

These hazards are then resolved by pushing the affected operation of the duplicate unit in later control steps, thereby shifting its successors accordingly. The push is done such that the interval between $v \in U^{OG}$ and $v' \in U^{DP}$ is greater than (or equals to) k_c. This identification and

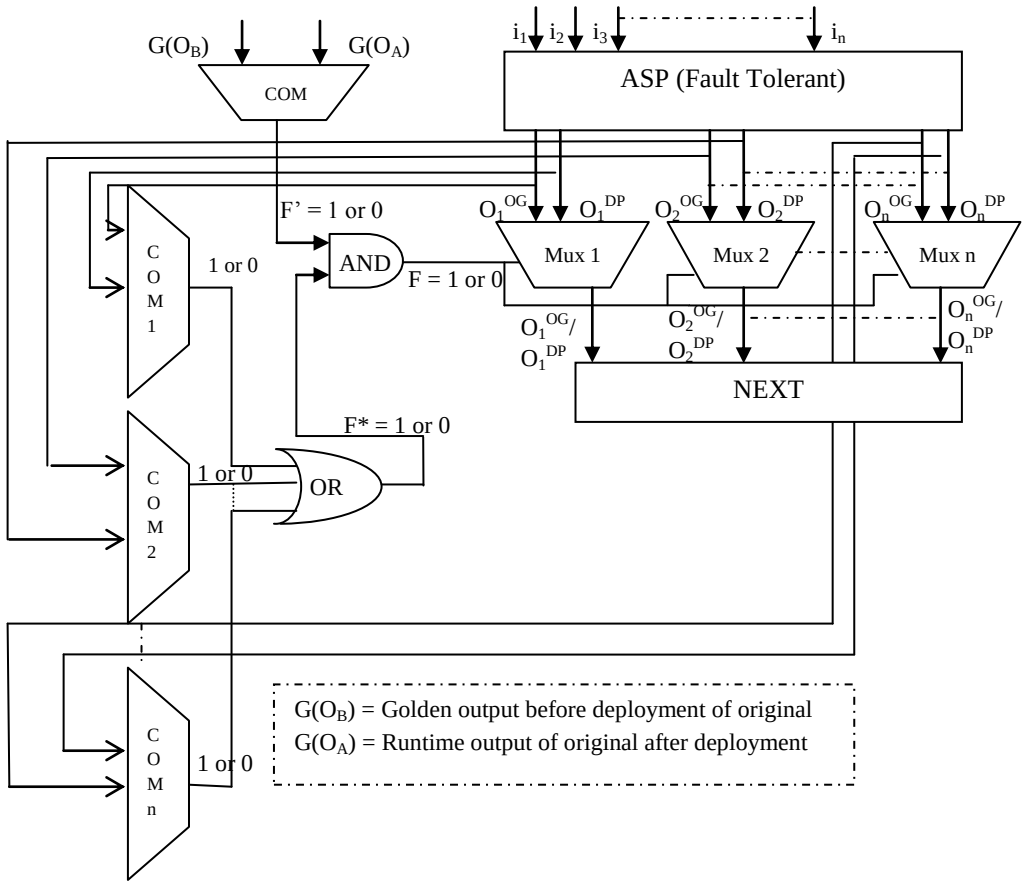


Figure 6.6 Circuit Diagram for Voting Scheme

resolution of the TFH is done until all the TFH of the whole DMR system are resolved, i.e. in $SDFG^{DMR}$ all:

$$t(v') - t(v) \geq k_c \quad (6.2)$$

Figure 6.2 illustrates the pseudo code of the proposed MCFT algorithm.

6.2.2.3 Demonstration of MCFT using DMR

The proposed approach is explained with the demonstration of Auto Regression filter (ARF) DFG. Figure 6.3 and Figure 6.4 shows the non fault tolerant $SDFG^{DMR}$ and Figure 6.5 shows the fault tolerant $SDFG^{DMR}$ of ARF respectively based on the final explored solution of 1(+), 2(*) obtained through the proposed DSE framework. In other words, this indicates while designing DMR schedule of ARF the operations of both the units have to simultaneously obey the resource configurations explored by BFOA DSE. The DMR system has operations labeled as 1, 2, … *n* for the original unit while the duplicate unit operations are represented as ‘1, ‘2, … ‘*n*, where *n* and *‘n* are the values of maximum number of node in the particular application. In the $SDFG^{DMR}$ shown in Figure 6.3, the value of *n* is 28 for the original unit while that of *‘n* is ‘28 for the duplicate unit. Suppose if a 2-cycle fault occurs in the system, then, the TFHs occurring in the system are first identified. TFH occur between operations

Table 6.1 Output Unit Selection

F*	F’	F	Output
1	1	1	U^{DP}
1	0	0	U^{OG}
0	1	0	NOT VALID
0	0	0	U^{OG}

belonging to a particular operator in U^{OG} and operations performed by similar operator in U^{DP}, if the operations are not k-cycle apart. Corresponding to the Figure 6.3, list L[k] contains hazards between 18(M2) and ‘1(M2), 17(M1) and ‘2(M1), 8(A1) and ‘5(A1) and so on. For example, if a 2 cycle fault occurs at M1 at control step 4, whose effect continues until step 5. Accordingly multiplier M1 may incorrectly execute operation ‘2 at step 5 in U^{DP}, thereby producing a faulty output. Therefore, in order to make the system fault tolerant (i.e. mask the fault occurring in original) and to generate an error free output, operation ‘2 assigned to M1 of U^{DP}, is pushed below into step 6 (where equation 6.2 is satisfied and explored resource configuration is met (in this case: 1(+), 2(*))) to avoid the propagation of 2-cycle fault in the U^{DP} as well as propagation of fault from UDP to UOG (using step 10 of algorithm in Figure 6.2). Similarly as per step 7 and 10 of the algorithm (Figure 6.2), opn ‘3 and ‘4 is scheduled in step 14 as in the prior steps either resource constraint (X_i) was being violated or k_c fault was being propagated. Figure 6.5 shows 2-cycle fault isolated $SDFG^{DMR}$ obtained through this process. This arrangement ensures that similar operators in both units are isolated by more than 2-cycles to prevent propagation of faults from one unit to other.

6.2.2.4 Proposed equivalent circuit for a voter

Figure 6.6 shows the equivalent circuit diagram for a voter to compare the outputs of the respective units in DMR system. The outputs are compared, in order to find whether a fault has occurred in the system or not. If a fault has occurred in the system, the duplicate scheduling unit obtained through the proposed algorithm always remains fault free. This indicates the outputs of the original and duplicate scheduling unit will always have a difference (original producing faulty output, while duplicate producing non faulty). Therefore, the comparator units are used to perform this comparison of outputs. In case of multiple outputs from original and duplicate the outputs of comparators are OR’ed (oring the outputs of comparison from multiple comparators helps the system in indicating fault if atleast one of the output is faulty i.e atleast one comparator produces a difference) and fed

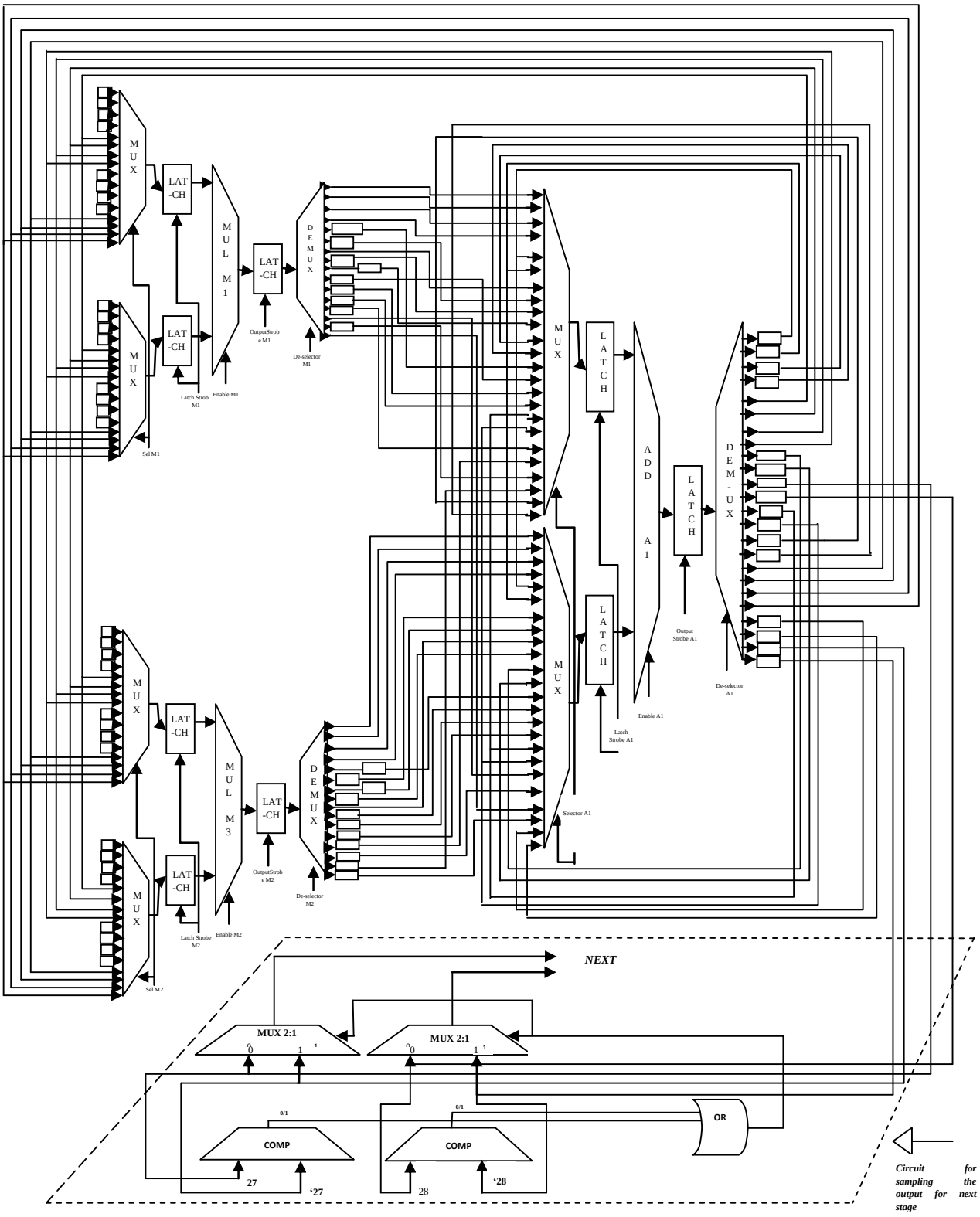


Figure 6.7 Datapath Circuit Corresponding ARF with X_i = 1(+), 2(*)

into the select line 'F' of the multiplexers. As seen in Table 6.1, if F = 1 (indicating fault because of difference produced by comparators), then the output is taken from duplicate unit else for a fault free system (F=0 indicating no difference in outputs of both units), the output is taken from the original. This scheme described above assists in extracting the outputs from

respective units without extra redundancy (such as TMR) and saves unnecessary clock cycles.

Figure 6.7 shows the datapath circuit of the demonstrated fault tolerant ARF application (Figure 6.5). The data path circuit incorporates multiplexers and demultiplexers into the system. These multiplexing and demultiplexing units are used for representing the systems resources with their respective inputs, outputs, the operations performed by them and the necessary storage units along with the necessary interconnections. The empty boxes in Figure 6.7 represent the register units, required to store the inputs or the intermediate results of an operation. The equivalent circuit of voter is represented with a dashed block. The block contains multiplexers and comparators units to compare the outputs ((27 or ’27) and (28 or ’28)) based on the value of select line ‘F’ (either 0 or 1). Further, generating the error free output from the system.

6.3 Proposed Evaluation Models

For evaluation of a particle (or design point), the following models have been proposed.

6.3.1 Proposed Power Model

P_T^{DMR} of a resource set is represented in terms of Static Power (P_S^{DMR}) and Dynamic Power (P_D^{DMR}). ‘P_T^{DMR}’ is represented as:

$$P_T^{DMR} = P_S^{DMR} + P_D^{DMR} \tag{6.3}$$

P_S^{DMR} is a function of area of resources and leakage power per transistor. It can be formulated as:-

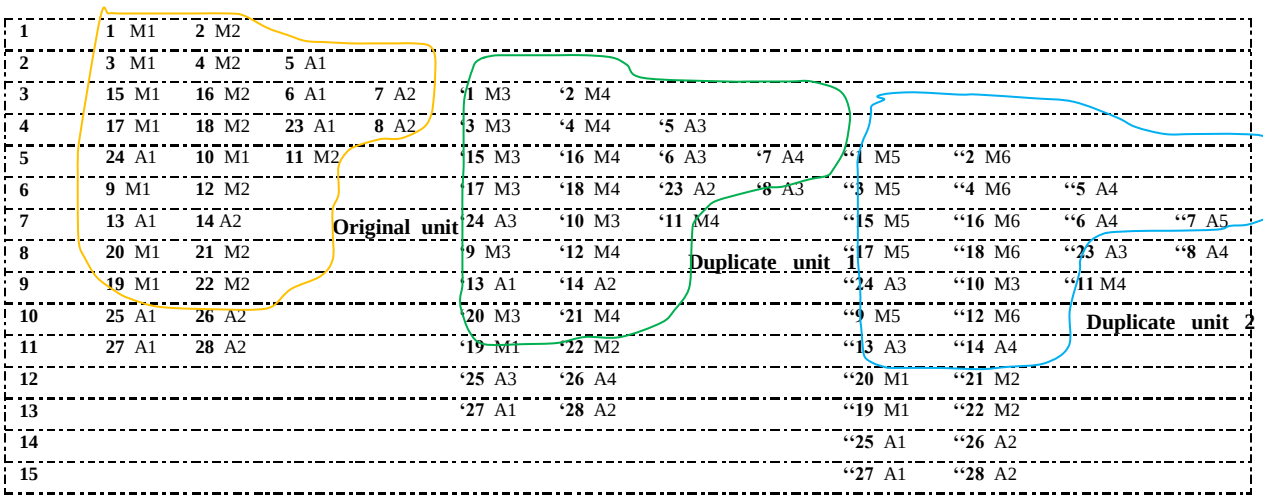
$$P_S^{DMR} = \sum_{d=1}^{D} (N(R_d).K(R_d).p_c \tag{6.4}$$

Where ‘$N(R_d)$’ represents the number of instances of resource R_d. ‘$K(R_d)$’ represents the area occupied by resource R_d, ‘D’ is the number of resources (FU’s) and ‘p_c’ denotes the power dissipated per area unit (e.g. transistors).

While, the average dynamic power consumed by a resource configuration is a function of dynamic activity of the resources and can be given as:

$$P_D^{DMR} = \frac{E_{FU}^{DMR}}{T_E^{DMR}} \tag{6.5}$$

Where, E_{FU}^{DMR} is the total energy consumption of the resources in fault secured DMR system and T_E^{DMR} is the total execution time of DMR system.



Note: The operation numbers above correspond to the nodes of ARF benchmark shown previously in Figure 6.3

Figure. 6.8 SDFG^{TMR} for [32] Corresponding ARF with X_i = 5(+), 6(*) for k_c = 2

6.3.2 Proposed Execution Time (Delay) Model

For given 'D' functional resources the execution time is:

$$T_E^{DMR} = \sum_{c.s=1}^{n} Max(D(op_i),....D(op_n),D(op_{'i}),.....D(op_{'n})) \quad (6.6)$$

Where, $1 \le i \le n$ and '$1 \le$ '$i \le$ 'n. (Here, operations in original and duplicate are labelled as i and i' respectively; n and n' are maximum value of node); $D(opn)$ is the delay of operation 'n' while c.s is the control step.

6.3.3 Proposed fitness function

The proposed fitness function is defined as:

$$C_f(X_i) = \phi_1 \frac{P_T^{DMR} - P_{cons}}{P_{\max}^{DMR}} + \phi_2 \frac{T_E^{DMR} - T_{cons}}{T_{\max}^{DMR}} \quad (6.7)$$

Where, $C_f(X_i)$ is the cost of particle with resource set X_i, $T_{\max}^{DMR}$ is the maximum execution time of a fault tolerant DMR system in design space while $P_{\max}^{DMR}$ is the maximum power of a fault tolerant DMR system in design space.

6.3.4 Advantages of proposed scheme over existing scheme

The proposed MCFT scheme offers several novelties/ advantages over the fault tolerant approach [32].

- The existing approach [32] employs triple modular redundancy (TMR) scheme to make the entire schedule fault tolerant. While the proposed scheme generates a fault tolerant schedule using double redundancy. The respective TMR obtained for ARF benchmark through [32] is shown in Figure 6.8. However, in contrast, the proposed

algorithm obtains a fault tolerant schedule for ARF using DMR as shown in Figure 6.5 before. As observed from Figure 6.8, the structure obtained consumes 5(+), 6(*) and control steps = 15 (Latency = 121.5 us); while the final fault tolerant DMR for ARF (Figure 6.5) obtained through proposed BFOA-DSE approach occupies 1(+), 2(*) and control steps = 30 (Latency = 179.5 us). The corresponding cost of the fault tolerant solutions calculated using equation (6.7) for proposed and [32] is -0.217 and 0.0443 respectively. This indicates a substantial improvement in final cost (quality).

- The final solutions obtained through the proposed and [32] have a significant difference in quality (optimality). This is because the solution obtained through [32] does not satisfy the power budget and execution time constraint specified by user. However, the proposed approach explores (by iteratively refining through BFOA) a solution which not only satisfies the power budget and execution time constraint specified by user but also comprehensively minimizes the total cost of the solution.
- The proposed approach produces a fault tolerant structure using DMR (without using conventional voter scheme) in contrast to previous fault tolerant approach [32] using TMR.

6.4 Termination criteria

The BFOA driven exploration process has following terminating criteria:

- Terminates when a designer specified 'N_c' is reached.
- When no improvement is seen in global best among bacteria population over last 10 iterations (chemotactic steps).

Details on termination criteria have already been discussed in chapter 3.

Note: Results of proposed approach are explained in chapter 8 section 8.4.

6.5 Summary

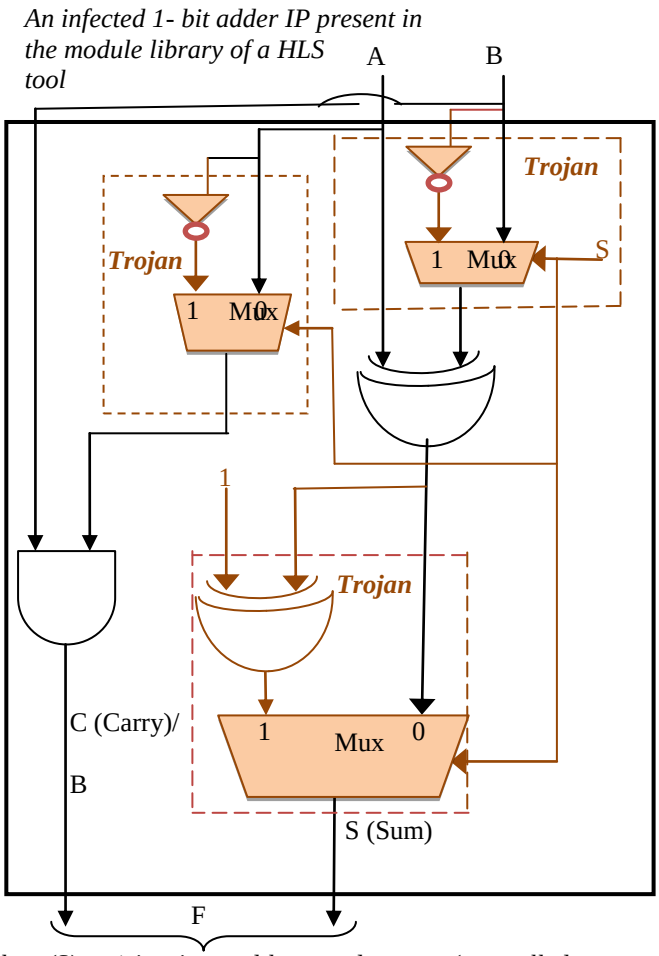
The availability of faster devices is a feature of future technologies that induces major concerns to the fault detection community for longer duration transient faults. The technology evolution and LET of particle both play a major role in inducing multi-cycle (*k*-cycle) transient fault (longer duration transient) in a device. Hence, optimizing power and delay remains no longer sufficient now, specifically for current generation of systems which demand designs (especially for space applications where radiation induced faults are highly possible) that requires ability to detect errors occurring due to transient faults (such as single event upsets). Therefore, an adaptive/intelligent system for solving the DSE problem of multi-cycle transient fault tolerant datapath during HLS has been proposed in this chapter.

Chapter 7

Untrusted Third Party Digital IP cores: Power-Delay Trade-off Driven Exploration of Hardware Trojan Secured Datapath during High Level Synthesis

Complexity of the SoC has increased tremendously over the years. This allows us to have more complex systems. However, the design productivity has not increased with the same pace. Therefore, to address this issue reuse based methodology has come into context, which will benefit in producing complex designs at a higher productivity. To design complex systems IPs are used which increase the productivity of design. This process requires globalization of IPs through third party vendors. That is, globalization incurs importing IPs from various 3P vendors. But there are serious security concerns for SoC integrators, due to involvement of untrustworthy 3P vendors supplying IP cores. During the design stage of a 3PIP, an adversary (possibly an untrustworthy vendor) can deliberately infuse a Trojan logic resulting in malfunctioning of the digital circuit. Typically, the Register Transfer Level (RTL) files of the modules/IPs of the library are provided by the HLS Company which it may have imported from third party vendors as RTL files. Therefore, to have a trustworthy design it should be ensured during HLS that any possible infection of 3PIP is detectable. Detection process of the Trojan during design of hardware Trojan secured schedule in HLS inevitably requires multiple redundant hardware instances from different vendors, which if not accounted for its power and delay during fitness evaluation, may result in a secured circuit violating user constraint.

This chapter solves the aforementioned problem and proposes an approach which generates a low cost Trojan secured schedule during HLS. The focus on hardware Trojan secured schedule generation during HLS has been very little with absolutely zero effort so far in DSE of a user MO constraint optimized hardware Trojan secured schedule. The design process of hardware Trojan secured schedule should hence administer the usage of intelligent



Note: Only when select (S) = 1 is triggered by an adversary (controlled externally), then, Trojan blocks get activated and the adder IP starts performing subtraction resulting in functional failure. Until triggered, it remains dormant in the system and behaves like a normal adder IP.

Figure 7.1 An Infected 1- bit Adder IP Present in Module Library of a HLS Tool

DSE strategy that is driven through user power-performance constraints for exploring an optimized hardware Trojan secured schedule. The detail description of the proposed approach is given in subsequent sections of this chapter.

7.1 Problem Formulation

To explore the design space of a given DFG, and determine an optimal solution

$$X_i = (\overrightarrow{R_n}, A_v)$$

$$\overrightarrow{R_n} = (N(R_1), N(R_2), N(R_d)...N(R_D))$$

$$X_i = \{N(R_1), N(R_2), N(R_d)....N(R_D), A_V\}$$

which satisfies conflicting user constraints and minimizes the overall cost.

The problem can be formulated as:

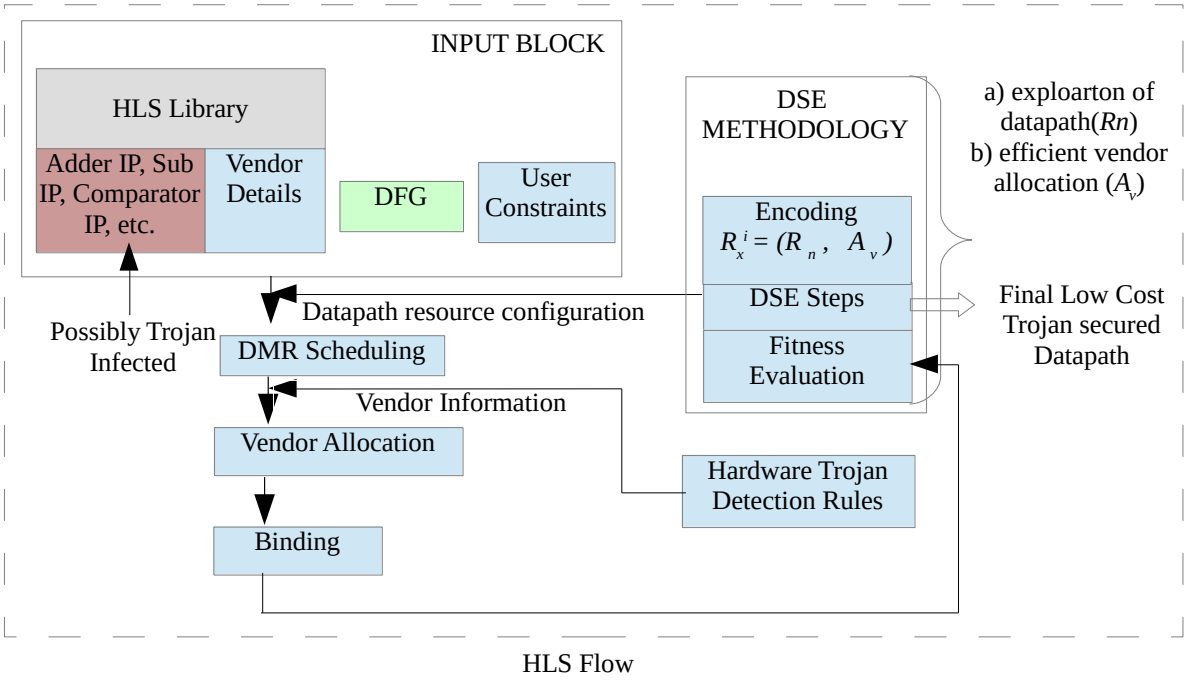


Figure 7.2 Proposed Methodology for Trojan Secured Datapath

Find: an optimal X_i

with minimum hybrid $\mathrm{Cost}(P_T^{DMR}, T_E^{DMR})$

subjected to: $P_T^{DMR} \leq P_{cons}$ *and* $T_E^{DMR} \leq T_{cons}$ and hardware Trojan fault secured.

Where, $N(R_d)$ is number of instances of a resource type 'd', P_T^{DMR} is power consumed by a fault secured double modular redundant (DMR) system, T_E^{DMR} is the delay of a DMR design, T_{cons} and P_{cons} are the user specified execution delay and power constraints while A_v is the vendor allocation procedure type (where A_v = '1' or '0').

7.2 Proposed Methodology

7.2.1 Motivation

Let us consider a scenario to explain the problem of hardware Trojan in 3PIP and hardware security during HLS. During HLS, an untrusted IP vendor may malevolently insert Trojan logic into the module/IP that is used in the module library of a HLS tool. This Trojan logic remains hidden until triggered externally by the adversary and is therefore not possible to detect during normal RTL simulation. This is because during normal situations (when not triggered), it behaves like a functionally correct IP. Figure 7.1 shows an example of Trojan in a third party IP/module present in the module library of a HLS tool. Here a 1 bit adder IP that is used in the module library of a HLS tool may behave as a subtractor IP on triggering (through external activation by setting S = 1). The detection process of such Trojans during

HLS becomes impossible with the Trojan detection techniques applied at lower levels of abstraction such as side channel analysis and RTL simulation.

Note: The work presented in this chapter targets Trojans in 3PIPs that affect notmal functional output.

The detection procedure suggested in the recent literature is accomplished by having IP cores of same functionality from different vendors. This is because different vendors will have different implementations and it is less likely that both are Trojan infected. Even if they are, the chances of different vendor IPs generating same output behavior is considered extremely uncommon. However, detection process of the Trojan during design of hardware Trojan secured schedule in HLS inevitably requires multiple redundant hardware instances from different vendors, which if not accounted for its power and delay during fitness evaluation, may result in a secured circuit violating user constraint. Therefore, the design process of hardware Trojan secured schedule should govern the usage of adaptive intelligent DSE based on user power-delay constraint as well as effective vendor allocation procedure during scheduling. The framework to obtain Trojan secured schedule is explained in the subsequent sections of this chapter.

7.2.2 Proposed Framework

This section presents a framework which generates a low cost optimal hardware Trojan secured schedule based on user power-delay constraint during HLS. The framework has been shown in Figure 7.2 Module library, behavioral description of DFG and predefined user parametric constraints for power and time executions (or delay) are provided as inputs to the exploration process. A set of control parameters such as 'N_c' (maximum number of chemotaxis steps allowed which is the stopping criterion that indicates the maximum limit of the iterations that the proposed approach is allowed to execute) and '*p*' (population size) are used for regulating the BFOA driven exploration process where *'p'* indicates the number of individuals/bacterium (initial design solutions) participating in the evolutionary process of exploration.

7.2.3 DSE Framework

The DSE framework employed for generating a lost cost Trojan secured schedule during HLS is BFOA-DSE. To solve the problem mentioned in this chapter, BFOA as DSE framework is used to explore the design space. The framework of this algorithm provides the flexibility to be configured in a proficient way for eliciting efficient search behavior for this

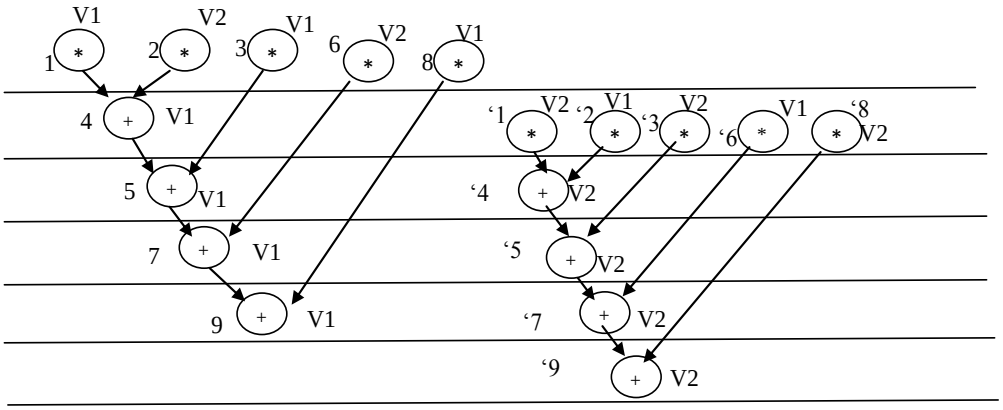


Figure 7.3 IIR Filter for $A_v = 0$; $\overrightarrow{R_n}$ = 2(+), 5(*) indicating Alternate Assignment Procedure of Two Vendor Types

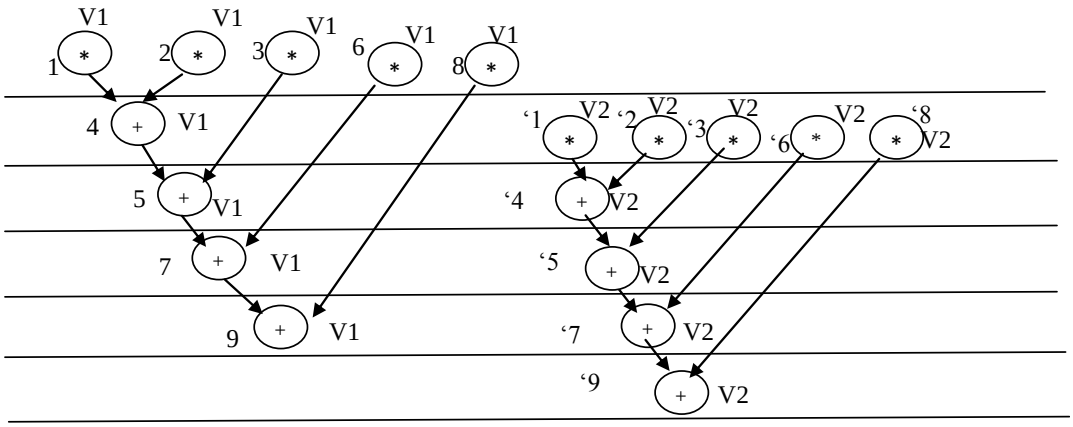


Figure 7.4 IIR Filter for $A_v = 1$; $\overrightarrow{R_n}$ = 2(+), 5(*) indicating Each Entire Unit Strictly Assigned to Same Vendor Type (U^{OG} to ‘*V*1’ and U^{DP} to ‘V2)

problem. BFOA comprises of primarily of two major steps: chemotaxis and dispersal for locomotion of bacterium. Using locomotive mechanisms (such as flagella) bacteria can move around in their environment, sometimes moving chaotically, and other times moving in a directed manner, referred to as swimming. The details about BFOA-DSE have already been explained in chapter 3.

7.2.4 Proposed Encoding

A bacterium position (candidate design solution) is labeled as X_i:

$$X_i = (\overrightarrow{R_n} , A_v) \tag{7.1}$$

Where, $\overrightarrow{R_n}$ indicates the resource array (resource configuration e.g. number of adders, multipliers etc) and ‘A_v’ is the vendor allocation procedure type adopted. The reason behind

incorporating the last dimension with vendor allocation procedure type 'A_v' is discussed in later sections.

7.2.5 Motivation of using Vendor Allocation Procedure 'A_v' in Problem Encoding During Exploration

In order to detect hardware Trojans a minimum of two vendors are always needed to provide distinctness. However, technique of usage of the two vendors during allocation inside the DMR scheduling (i.e. assignment process) of each vendor IPs inside the system during allocation) dictates the final latency and power of entire system. This is because same resource type/IP from two different vendors has different area, power and delay. Hence, merely using distinctive vendor assignment for detection without probing into the procedure of allocation (assignment) of vendor type in DMR system may lead to skipping of an alternate better solution in context of DSE of a low cost optimal Trojan secured schedule, exploration of an additional dimension, 'A_v' (indicating allocation procedure of IP's from different vendor type) which can either be '0' or '1' is incorporated in the bacterium encoding along with resource array. The value of 'A_v' as '0' or '1' is interpreted as follows:

7.2.5.1 Vendor Allocation Procedure (Type 1): $A_v = 1$

- All operations of a specific unit being strictly assigned to resources of **same** vendor type (say: all operations of original unit strictly assigned to **same** vendor 'V_1' and all operations of duplication to **same** vendor 'V_2').
- Similar operations of both original unit U^{OG} and duplicate unit U^{DP} being assigned to different vendors.

7.2.5.2 Vendor Allocation Procedure (Type 2): $A_v = 0$

- **Alternate** vendor assignment to operations in control step of a unit. (Example, in Figure 7.3, operation 3 & 6 assigned alternatively to 'V_1' and 'V_2'. Next multiplication if any would have been assigned to 'V_1' alternately).
- Similar operations of both U^{OG} and U^{DP} being assigned to different vendors**.**

In both above cases, whenever there is a conflict of operation during scheduling between operation of U^{OG} and U^{DP}, preference is given to the operation of U^{OG} during scheduling.

7.2.6 Library Assumed

It is assumed that multiplier and adder provided by vendor V_1 has area = '2468au' & '2034au', latency = '10000ns' & '265ns', and energy = '10.0pJ' & '0.80pJ' while multiplier

and adder provided by vendor V_2 has area = '2464au' & '2032au', latency = '11000ns' & '270ns' and energy = '9.8pJ' & '0.739pJ' respectively.

7.2.7 Proposed Evaluation Models

For evaluation of a particle (or design point), the following models have been proposed.

7.2.7.1 Proposed power model

Total power consumption (P_T^{DMR}) by a resource set is represented in terms of Static Power (P_S^{DMR}) and Dynamic Power (P_D^{DMR}). 'P_T^{DMR}' is represented:

$$P_T^{\mathrm{DMR}} = P_S^{\mathrm{DMR}} + P_D^{\mathrm{DMR}} \tag{7.2}$$

Static power (P_S^{DMR}) is a function of area of resources and leakage power per transistor. It can be formulated as:-

$$P_S^{DMR} = (\sum_{j=1}^{2} \sum_{i=1}^{n} (A(R_i^{V_j}) * R_i^{Vj})) * p_c \tag{7.3}$$

where, R_i^{Vj} is the number of instances utilized from vendor V_j for a resource type R_i, and 'n' is the maximum number of instances of resource type R_i for vendor V_j while $A(R_i^{Vj})$ is the area of a resource type (R_i) corresponding to vendor (V_j). On the other hand, the average dynamic power consumed by a resource configuration is a function of dynamic activity of the resources and can be given as:

$$P_D^{DMR} = \frac{E_{FU}}{L_T^{DMR}} \tag{7.4}$$

Where, E_{FU} is the total energy consumed by the resources. *Note: The power component includes power due to functional resources, interconnect units (mux and demux), comparator (for error detection) as well as overhead incurred from internal buffering (during temporary storage of operation output in DMR scheduling).*

7.2.7.2 Proposed Delay (Latency) model

For given 'D' functional resources the delay is:

$$L_T^{DMR} = \sum_{c.s=1}^{c.s(\max)} \sum_{j=1}^{2} Max(D(op_i^{Vj}), ...D(op_n^{Vj}), D(op_{i}^{Vj}), ...D(op_{n}^{Vj})) \tag{7.5}$$

Where, $1 \leq i \leq n$ and '$1 \leq$ '$i \leq$ 'n. (Here, operations in original and duplicate is labeled as i and 'i respectively; n and 'n = maximum number of operations in original and duplicate unit).

Here, $D(op_i^{Vj})$ is the delay of operation i, assigned to vendor V_j, $c.s$ represents control steps, while $c.s(max)$ is the maximum number of control steps in a schedule.

7.2.7.3 Proposed Cost model

The proposed fitness function (considering total delay and power consumption of a solution) is defined as:

$$C_f(X_i) = W_1 \frac{P_T^{DMR} - P_{cons}}{P_{\max}^{DMR}} + W_2 \frac{L_T^{DMR} - L_{cons}}{L_{\max}^{DMR}} \quad (7.6)$$

Where, $C_f(X_i)$ is the cost of bacterium with resource set X_i, P_{max}^{DMR} and L_{max}^{DMR} are the maximum power and delay of the *DMR* system and W_1 and W_2 are the user defined weights both kept at ½ during exploration to provide equal preference.

7.2.8 Demonstration

For a resource set $\overrightarrow{R_n}$ = 2(+), 5(*), there are two possible DMR schedules generated for IIR filter benchmark on the basis of A_v = 0 and 1, as seen in Figure 7.3 and 7.4. More specifically, for R_x = (2(+), 5(*), 0), the latency is: 23,080ns and power is: 0.58mW; while, for R_x = (2(+), 5(*), 1), the latency is: 22,080ns and power is: 0.88mW. Clearly, a difference is observed in the delay and power of the two generated scheduling solutions both abiding by distinct vendor type assignment to similar operations for detect ability. The schedules generated in Figure 7.3 and 7.4 are both hardware Trojan fault secured (with two vendor needed), however, one is better than the other in different parameter. Only using distinct vendor assignment without probing into the procedure of allocation of vendor type in DMR system may lead to missing of better alternative (or optimal) solution in context of DSE. Therefore, in context of DSE, it is worth to explore the additional dimension 'A_v' incorporated in the proposed bacterial encoding.

7.3 Termination Criteria

The BFOA driven exploration process has following terminating criteria:

- Terminates when reached designer specified 'N_c' (maximum chemotactic steps).
- When no improvement is seen in global best among bacteria population over last 10 iterations (chemotactic steps).

Details on termination criteria have already been discussed in chapter 3.

Note: Results of the proposed method are given in chapter 8 section 8.5.

7.4 Summary

Due to globalization, there have been serious concerns on the security and trustworthiness of 3PIPs , rendering the IP susceptible to possible hardware threats. To provide secure information processing through digital ICs within user constraints and to ensure trustworthiness while designing a low cost optimized DMR, Trojan secured HLS methodology is crucial. This chapter presented a novel low cost Trojan security aware HLS methodology. The approach explores efficient vendor allocation procedure within the proposed DSE framework. It also provides a significant reduction in the cost of security aware HLS solution in comparison to similar prior work.

Chapter 8

Results and Analysis

This chapter describes the complete experimental results of the proposed methodologies for DSE described in previous chapters. This chapter divided into five sections where each section present results of the respective methodology. The sections are as follows:

8.1 Experimental Results: Adaptive Bacterial foraging driven Datapath Optimization: Exploring Power-performance Trade-off in High level synthesis

This section describes the experimental results of the proposed approach explained in Chapter 3 and the improvements obtained compared to recent approach [20, 21]. The proposed approach has been implemented in java and run on Intel Core-i5-3210M CPU with 3MB L3 cache memory and 4GB DDR3 primary memory. The processor has frequency of 2.5 GHz. Various HLS benchmarks were chosen for experimentation such as JPEG Downsample [1, 45], JPEG IDCT2 [98], IDCT [99], Feedback Points [1, 98], ARF [45, 100], BPF [1], FIR [1, 98, 86], and MESA Matrix Multiplication [86, 45]. The proposed approach can handle problems of any size. Many large size benchmarks have also been tested through our approach. The library is given in chapter 3 Table 3.1.

Experimentation is carried out considering two aspects:

- Analysis of variation of multiple BFOA parameters and their impact on the BFOA driven DSE performance.
- Comparison of BFOA-DSE with previous DSE approaches in terms of Quality of Results (for cost) and exploration time of the process.

The QoR is calculated as:

$$QoR = \frac{1}{2}\left(\frac{P_T}{P_{\max}} + \frac{T_E}{T_{\max}} \right) \tag{8.1}$$

8.1.1 Analysis of Proposed BFOA-DSE with variation of multiple BFOA parameters

In this section multiple internal BFOA parameters are varied and their impact on the results of proposed approach for selected benchmarks is noted. The bacterium size '*p*' and the

Table 8.1 Comparison of QoR and Exploration Time with respect to Bacterium size (*p*) for the Proposed Approach

Benchmark [45, 86, 98, 99, 100]	**Problem size of Benchmarks in terms of Nodes**	**Bacterium Size (*p*)**	**Cost**	**Convergence Time**	**Exploration time**
JPEG DOWNSAMPLE	33	3	-0.281	203	624
		5	-0.281	510	1220
		7	-0.281	700	1505
JPEG IDCT2	112	3	-0.372	34100	71510
		5	-0.372	73475	126595
		7	-0.371	182720	252425
IDCT	42	3	-0.301	1635	4310
		5	-0.301	2915	6725
		7	-0.301	4295	8695
FEEDBACK POINTS	41	3	-0.340	4134	12714
		5	-0.340	8545	13420
		7	-0.340	9235	15200
ARF	28	3	-0.239	540	2500
		5	-0.239	2800	5035
		7	-0.239	5035	8725
BPF	29	3	-0.296	475	1140
		5	-0.296	835	1330
		7	-0.296	1165	2085
FIR	23	3	-0.268	155	565
		5	-0.268	330	1145
		7	-0.268	675	1675
MESA MATRIX MULTIPLICATION	84	3	-0.342	26675	51375
		5	-0.342	92424	126532
		7	-0.342	118605	160453

step size (*C(i)*) parameters are varied and results are analyzed on the basis of quality of results, convergence time and exploration time of the proposed DSE. The quality of solution found and its comparison with other DSE approaches will be discussed in the next subsection.

Table 8.2 Impact in the Variation of Step Size (*C(i)*) on the Performance of Proposed DSE

Benchmarks [45, 86, 98, 99, 100]	*C(i)=C(i)+1*		*C(i)=C(i)+2*		Configuration
	Convergence Time(ms)	Exploration Time(ms)	Convergence Time(ms)	Exploration Time(ms)	
JPEG DOWNSAMPLE	312 (*p*=3)	687 (*p* =3)	203 (*p* =3)	624 (*p* =3)	3(*), 1(+)
JPEG IDCT2	85929 (*p* =5)	195481 (*p* =5)	73475 (*p* =5)	126595 (*p* =5)	7(*), 1(+)
IDCT	1605 (*p* =3)	4320 (*p* =3)	1635 (*p* =3)	4310 (*p* =3)	3(*), 1(+)
FEEDBACK POINTS	7845 (*p* =3)	11520 (*p* =3)	4134 (*p* =3)	12714 (*p* =3)	6(*), 1(+)
ARF	2106 (*p* =3)	4602 (*p* =3)	540 (*p* =3)	2500 (*p* =3)	3(*), 1(+)
BPF	780 (*p* =3)	3120 (*p* =3)	475 (*p* =3)	1140 (*p* =3)	4(*), 1(+)
FIR	421 (*p* =3)	1934 (*p* =3)	155 (*p* =3)	565 (*p* =3)	3(*), 1(+)
MESA MATRIX MULTIPLICATION	50115 (*p* =3)	80392 (*p* =3)	26675 (*p* =3)	51375 (*p* =3)	6(*), 1(+)

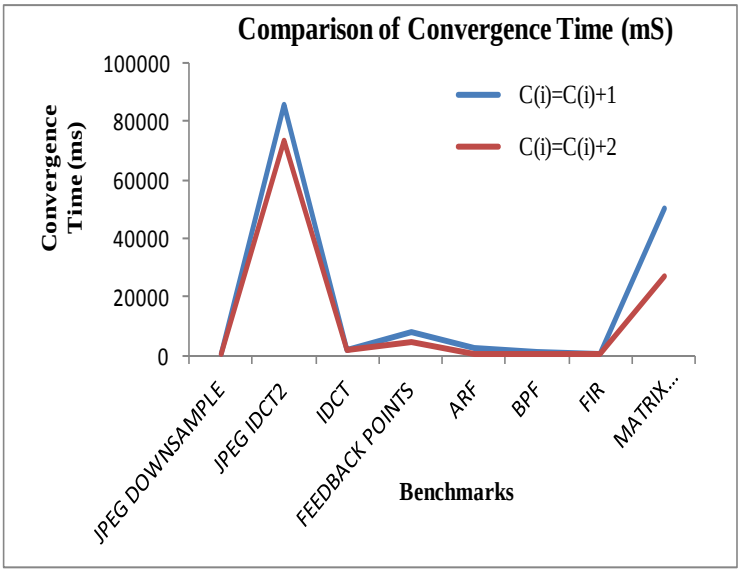


Figure 8.1 Comparison of Convergence Time with respect to Step Size *C*(*i*)

8.1.1.1 Bacterium size, *p*

In the proposed approach (BFOA-DSE), the bacterium size and configuration chosen is able to comprehensively cover a design space. Generally, larger the bacterium size '*p*', larger will be the coverage of exploration of the design space in each iteration. However, during experimentation of different benchmarks it was found that best size of bacterium for

proposed BFOA-DSE is p= 3 for most of the benchmarks. Had the design space been even larger (or a very large size application), the advantage of having a large size '*p'* would had been visible. During the experiment, the results have been evaluated for three different bacterium size, i.e for p = 3, 5 and 7.

The results are shown in the Table 8.1. As evident from the results for most benchmarks, the best balance between achieving the fast exploration speed and having an optimal solution is obtained at the bacterium size 'p' = 3. The primary selection criteria was low cost (or high quality) solution. However, if the results found for different bacterium size 'p' were found to be same, then the bacterium size for which the fastest convergence (and exploration) was obtained was selected. However, there are some cases, for the larger size benchmarks like JPEG IDCT2, better solutions are obtained at the bacterium size p = 5. This behavior for the particular benchmark is due to the ability of attaining an optimal solution with more bacteria initialized (p=5) over the design space.

As evident from the Table 8.1, it can be stated that a faster convergence is achieved for smaller bacterium size as compared to the larger ones. Also, as the bacterium size increases the exploration time also increases, since the number of bacteria per iteration increases, thereby, increasing the computation complexity per iteration. The underlined bacterium size 'p' indicates the selected size which yields the most efficient results in terms of quality (followed by small exploration time if the quality remains same). Therefore, after analyzing the results it can be observed that for the tested benchmarks, the quality of results obtained after varying the bacterium size indicates that in most of the cases faster convergence and exploration of an optimal solution is achieved at smaller bacterium size.

8.1.1.2 Step size, *(C(i))*

During experimentation, the impact in the variation of step size on the performance of proposed DSE has been investigated. The first variation is a step size $C(i) = C(i) + 1$ while the other is a step size $C(i) = C(i) + 2$. The effect of these variations is evaluated on the basis of convergence time exploration time and resource combination found. Table 8.2 shows the results obtained after varying the step size for the tested benchmarks. the variations in $C(i)$ do not have any impact on the quality of result found(which is evident from the fact that resultant resource configuration found is same for both $C(i) = C(i) +1$ and $C(i) = C(i) +2$)).However, the convergence time and exploration time for step size $C(i) = C(i) +1$ is higher compared to the step size of $C(i)= C(i) +2$. This behavior is due to the fact when the step size is small (i.e. varied with a change of one unit), the exploration process consumes

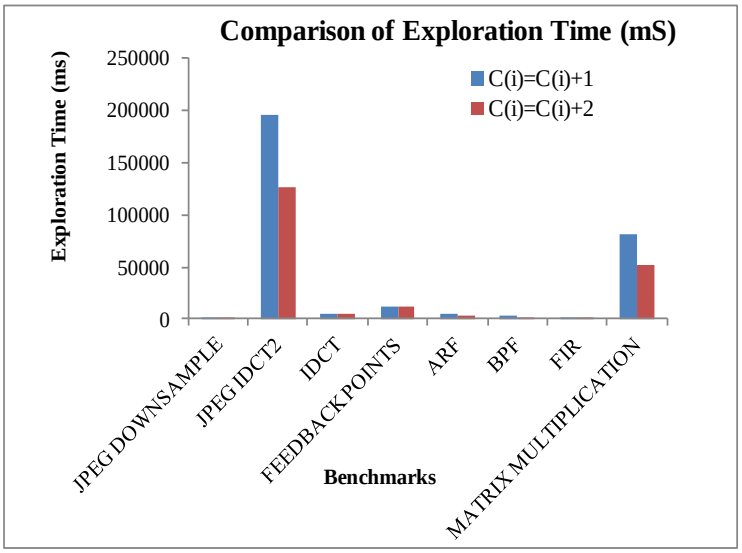


Figure 8.2 Comparison of Exploration Time with respect to Step Size *C*(*i*)

more time to achieve an optimal solution as the chances of occurrence of repetitive configurations is higher (when *C(i)* = *C(i)* +1 is substituted in equation (3.9) during chemotaxis). On the contrary, with a bigger step size (i.e. *C(i)* = *C(i)* +2), the exploration ability of the algorithm is better resulting in new (changed) configurations during chemotaxis.

The difference in step size only impacts the time to converge and not the quality (i.e. final configuration found). However, there is some exception to this observation as obtained for IDCT benchmark, where the convergence time for step size *C(i)* = *C(i)* +1 is found lesser than the convergence time of step size *C(i)* = *C(i)* +2. This is a special case, where at *C(i)* = *C(i)* + 1), the most optimal solution (1(+), 3(*)) was explored at a very early stage (at chemotactic step *j*= 2) from a bacterium position (1(+),1(*)). While at *C(i)* = *C(i)* +2, due to higher step jump, the chemotaxis function(eqn. (11)) from a bacterium position (1(+),1(*)) skips the optimal solution lying in between to reach a new bacterium position (2(+),3(*)). Therefore, the optimal solution (1(+), 3(*)) was attained after few more evaluations at *j* =5. This is an example where the exploitation ability of the algorithm (using small step size) has better productivity than the exploration ability (using bigger step size). Figure 8.1 and 8.2 shows the graphical representation of the variation of convergence time and exploration with the change in step size.

8.1.2 Comparison of BFOA-DSE with Previous DSE Approaches

This subsection describes the comparison of proposed BFOA-DSE with various previous approaches [20] and [21]. Based on these parameter selections, the detailed results for power

Table 8.3 Results of Estimated Power and Execution Time using Proposed Approach for DFGs

Note: For proposed approached baseline parameters : $\varphi_1 = \varphi_2 = 0.5$, the value of population size,p =3 or 5 $N_c = 120$, $N_{re} = 5$, $N_{ed} = 4$					
Benchmark [45, 86, 98, 99, 100]	**Resources found**	**Execution Time**		**Power**	
		Constraint	**Proposed solution**	**Constraint**	**Proposed solution**
JPEG DOWNSAMPLE	3(*), 1(+)	21ms	10ms	0.45mW	0.28mW
JPEG IDCT2	7(*), 1(+)	207ms	43ms	1.655mW	0.57mW
IDCT	3(*), 1(+)	86ms	32ms	0.45mW	0.28mW
FEEDBACK POINTS	6(*), 1(+)	108ms	21ms	0.9mW	0.50mW
ARF	3(*), 1(+)	110ms	54ms	0.4mW	0.27mW
BPF	4(*), 1(+)	89ms	11ms	0.3mW	0.35mW
FIR	3(*), 1(+)	48ms	31ms	0.59mW	0.27mW
MESA MATRIX MULTIPLICATION	6(*), 1(+)	240ms	65ms	1.125mW	0.49mW

Table 8.4 Comparison Of Proposed Approach With [20] in Terms of Exploration Time and Cost

Benchmark [45, 86, 98, 99, 100]	**Resource Configuration**		**Exploration Time**		**QoR (cost)**	
	BFOA	**[20]**	**BFOA**	**[20]**	**BFOA**	**[20]**
JPEG DOWNSAMPLE	**3(*), 1(+)**	1(*),1(+)	**0.624 sec**	13.65sec	**0.31**	0.51
JPEG IDCT2	**7(*), 1(+)**	4(*), 3(+)	126.5 sec	110.6sec	**0.22**	0.30
IDCT	**3(*), 1(+)**	2(*), 2(+)	**4.31sec**	12.6sec	**0.21**	0.37
FEEDBACK POINTS	**6(*), 1(+)**	3(*), 1(+)	**12.71sec**	25.6sec	**0.20**	0.29
ARF	**3(*), 1(+)**	4(*), 1(+)	**2.5sec**	14.3sec	**0.32**	0.35
BPF	**4(*), 1(+)**	2(*), 1(+)	**1.14sec**	10.54sec	**0.27**	0.43
FIR	**3(*), 1(+)**	4(*), 4(+)	**0.565sec**	8.2sec	**0.30**	0.38
MESA MATRIX MULTIPLICATION	**6(*), 1(+)**	3(*), 2(+)	51.3sec	11.65sec	**0.18**	0.51
Average reduction in Run Time w.r.t [20] =4 %			**Average reduction in cost w.r.t [20] = 35.98%**			

and execution time for the proposed approach are reported in Table 8.3. As evident from Table 8.3, the proposed approach has been comprehensively able to minimize and satisfy the specified constraints. For example, in case of IDCT, the explored solution, (3(*), 1(+)), consumes a power of 0.28mW and execution time of 32ms which substantially minimizes

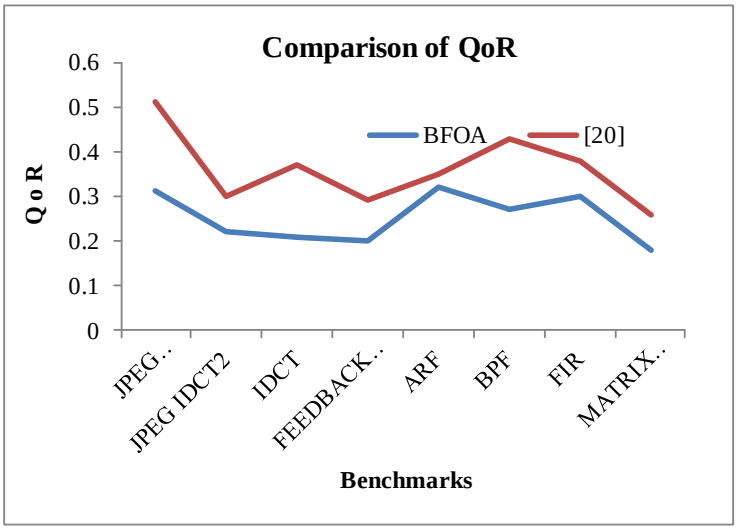


Figure 8.3 Comparison of QoR between BFOA-DSE and [20] Approach

Table 8.5 Comparison Of Proposed Approach With [21] in Terms of Exploration Time and Cost

Benchmark	Resource Configuration		Exploration Time		QoR (cost)	
	BFOA	**[21]**	**BFOA**	**[21]**	**BFOA**	**[21]**
JPEG DOWNSAMPLE	**3(*), 1(+)**	2(*), 2(+)	**0.624 sec**	27.8 sec	**0.31**	0.54
JPEG IDCT2	**7(*), 1(+)**	9(*), 2(+)	126.5 sec	14.2 min	**0.22**	0.26
IDCT	**3(*), 1(+)**	1(*), 8(+)	**4.31sec**	5.08min	**0.21**	0.69
FEEDBACK POINTS	**6(*), 1(+)**	9(*), 5(+)	**12.71sec**	1.26min	**0.20**	0.32
ARF	**3(*), 1(+)**	1(*), 8(+)	**2.5sec**	3.50min	**0.32**	0.85
BPF	**4(*), 1(+)**	1(*), 3(+)	**1.14sec**	2.08min	**0.27**	0.64
FIR	**3(*), 1(+)**	8(*), 1(+)	**0.565sec**	43.7sec	**0.30**	0.36
MESA MATRIX MULTIPLICATION	**6(*), 1(+)**	9(*), 1(+)	51.3sec	6.57 min	**0.18**	0.21
Average reduction in Run Time w.r.t [21] =90 %			**Average reduction in cost w.r.t [21]= 48 %**			

power and execution time as well as satisfies the user constraints specified. Similar results were obtained for other benchmarks. During experimentation, for proposed BFOA driven DSE, the following settings were maintained based on inferences drawn from the results obtained in section 8.1.1.1 and 8.1.1.2 : $\varphi_1 = \varphi_2 = 0.5$, $p = 3$ or 5, $N_c = 120$, $N_{re} = 5$, $N_{ed} = 4$.

8.1.2.1 Comparison with [20]

The proposed approach when compared with [20] gave substantially better results. As evident from theTable 8.4, the exploration time of proposed approach is much lesser than the [20] approach. Also, there is a substantial improvement in the quality of result obtained in case of

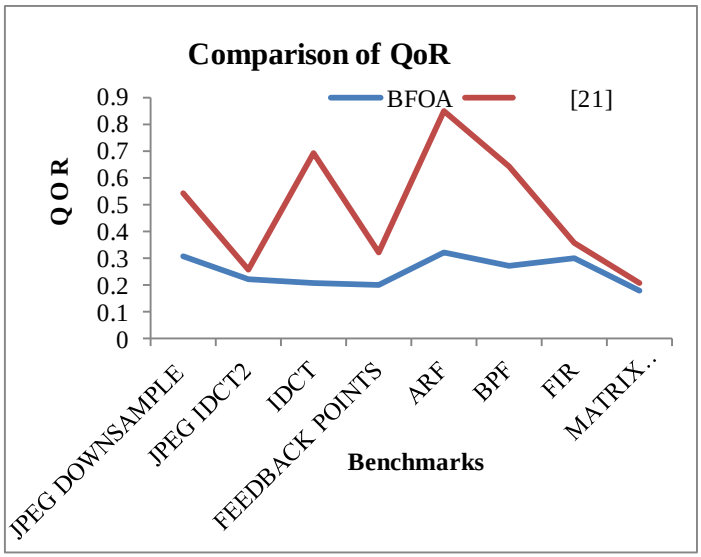


Figure 8.4 Comparison of QoR between BFOA-DSE and [21] Approach

the proposed approach. However, there are some cases where a higher exploration time is encountered. For JPEG IDCT2 the proposed approach has higher exploration time compared to [20]. This behavior is due to the large size of the benchmark. Though the exploration time for JPEGIDCT2 is slightly higher than the existing approach [20], nevertheless, the QoR cost of the obtained solution (7(*), 1(+)) obtained through the proposed approach is significantly better than [20]. This trend of better QoR has been observed for all the tested benchmarks.

Figure 8.3 shows the comparison of QoR (cost units) between the proposed approach (BFOA-DSE) and the [20] approach. After experimentation, it has been found that there is an average improvement in QoR of 35% and in exploration time of 4% as shown in Table 8.4.

8.1.2.2 Comparison with [21]

Table 8.5 shows the comparison of [21] with the proposed BFOA driven DSE approach. From the Table 8.5 it is evident that the exploration time of [21] is multiple times higher than the proposed approach. Also, proposed approach achieves a better QoR factor in comparison to [21] for most of the benchmarks. The average improvement in QoR is more than 48% and an average reduction of 90% is attained in exploration time as shown in Table 8.5.

Figure 8.4 shows the graphical representation of the comparison of QoR (cost units) between proposed methodologies and [21].

8.1.2.3 Comparison Based on Performance Metrics.

Table 8.6 presents the analysis of the proposed and existing approaches ([20] and [21] driven DSE) on the performance metrics. To evaluate the effectiveness of multi objective

Table 8.6 Comparison Of Proposed DSE Approach With [20] and [21] in Terms of Quality Metrics and QoR

Approach	GD	MFE	Spacing (S)	Spread (Δ)	Weighted Metric (W_m)
JPEG DOWNSAMPLE					
Proposed	0.000	0.599	0.052	0.937	0.408
[20]	0.571	1.050	0.000	0.742	0.656
[21]	0.198	0.636	0.000	0.636	0.417
JPEG IDCT2					
Proposed	0.000	0.984	0.133	0.869	0.430
[20]	0.031	0.954	0.000	0.710	0.370
[21]	0.004	1.007	0.000	0.830	0.417
IDCT					
Proposed	0.000	0.627	0.027	0.830	0.415
[20]	0.097	0.452	0.241	0.869	0.483
[21]	0.689	1.743	0.000	0.866	0.778
FEEDBACK POINTS					
Proposed	0.000	0.855	0.110	0.785	0.392
[20]	0.087	0.718	0.000	0.795	0.441
[21]	0.025	0.951	0.041	0.894	0.460
ARF					
Proposed	0.000	0.604	0.008	0.585	0.292
[20]	0.021	0.670	0.000	0.916	0.468
[21]	0.510	1.574	0.122	0.918	0.714
BPF					
Proposed	0.000	0.47	0.107	0.670	0.335
[20]	0.199	0.722	0.000	0.996	0.598
[21]	0.599	1.448	0.007	0.867	0.733
FIR					
Proposed	0.000	0.707	0.072	0.586	0.293
[20]	0.044	0.986	0.000	0.618	0.331
[21]	0.057	0.890	0.000	0.903	0.480
MESA MATRIX MULTIPLICATION					
Proposed	0.000	0.887	0.058	0.774	0.387
[20]	0.033	0.959	0.091	0.941	0.487
[21]	0.002	0.931	0.045	0.903	0.452

optimization algorithms, the metrics viz. GD, MFE, S, D and Wm are required to demonstrate how close the obtained solutions have converged to the true Pareto-optimal front [88]. It can be stated that a good optimization algorithm generates solutions close to the true

Pareto-optimal front as well as solutions that span the entire Pareto-optimal region uniformly. The GD metric is used to measure the convergence of solutions towards the Pareto-optimal front. Further, metrics which quantify the diversity among obtained non-dominated solutions are spacing and spreading. Spacing is the measure of relative distance between consecutive nondominated solutions. On the other hand, spread accounts to the diversity of the non-dominated solutions with respect to the extremities of the Pareto-solution set. An algorithm, finding smaller values of both is able to find better diverse set of nondominated solutions. In addition, Wm provides a combined qualitative measure of both closeness and diversity of the solutions. An algorithm having an overall small value of Wm is good in both aspects. As seen, the GD is zero for almost all benchmarks revealing that the proposed approach lies on the true Pareto front compared to [20] and [21]. In some cases, spacing is either zero, or very lower indicating that there is a uniform distribution of Pareto point on the curve. Also, the weighted metrics is lower for the proposed approach compared to [20] and [21] indicating better results obtained for all benchmarks.

8.2 Experimental Results: Automated Design Space Exploration of Multi-Cycle Transient Fault Detectable Datapath based on Multi-Objective User Constraints for Application Specific Computing

This section describes the experimental results of the proposed approach explained in Chapter 4 and the improvements obtained compared to recent approach [28, 30]. The proposed MCFD-DSE as well as [28, 30] has been implemented in java and run on Intel Core-i5-3210M CPU with 3MB L3 cache memory, 4GB DDR3 primary memory and processor frequency of 2.5 GHz. An average of 10 runs was reported for the proposed DSE with equal weightage to both user objectives of power and delay ($Ø_1$= $Ø_2$= ½) during experimentation. Various HLS benchmarks were chosen for experimentation such as JPEG Downsample [1, 45], JPEG IDCT2 [98], IDCT [99], Feedback Points [1, 98], ARF [45, 100], BPF [1], FIR [1, 98, 86], and MESA Matrix Multiplication [86, 45]. The proposed approach can handle problems of any size. The library is given in chapter 3 Table 3.1. This section discusses the following:

- Results of proposed approach for multi-cycle fault values in terms of delay and power user constraints.
- Comparative results of the proposed methodology and existing fault detectable approach [28, 30] in terms of resource solution found and cost of solutions. The user specified weightage of both metrics viz. power and execution time are both kept at ½ during exploration to provided equal preference.

8.2.1 Results of Proposed Approach for k_c = 10

Table 8.7 illustrates the results obtained for our proposed DSE of fault detectable datapath based on 10-cycle faults i.e. k_c = 10. It can be seen from results, the proposed approach comprehensively meets the user constraints of delay and power (and minimizes cost) for all benchmarks. This section provides the capability of the proposed approach to reach high quality solutions for transient fault of high strengths (k_c = 10) that satisfy the conflicting multi-objective user constraints as well as minimizes the hybrid cost function. There have been no previous works which report the results of exploration of power-execution time constraint driven fault detectable datapath system for single and multi-cycle fault strength. Further, the solutions obtained for the tested benchmarks are real optimal solutions which were verified by comparing with the golden solutions found by exhaustive analysis.

Table 8.7 Results of Proposed Fault Secure DSE approach for k_c = 10

Benchmark [45, 86, 98, 99, 100]	Resource set	T_{cons} (us)	T_E^{DMR} (us)	P_{cons} (mW)	P_T^{DMR} (mW)
IIR	1(+), 2(*)	70	57.1	0.3	0.238
BPF	1(+),2(*)	175	138.7	0.3	0.238
MPEG MV	1(+), 4(*)	170	82.67	0.7	0.405
ARF	1(+), 2(*)	220	178.7	0.45	0.23
DCT	4(+), 2(*)	210	143.8	0.5	0.45
FIR	3(+), 2(*)	100	78.6	0.6	0.46
WDF	2(+), 2(*)	172	126.1	0.4	0.3

Table 8.8 Comparison of Proposed Approach with Approach [28] and [30] for k_c=1

Benchmark [45, 86, 98, 99, 100]	Resource Set (proposed)	Resource Set [30]	Resource Set [28]	Cost (proposed)	Cost [30]	Cost [28]
IIR	1(+), 2(*)	1(+), 3(*)	2(+), 4(*)	-0.12	-0.092	-0.010
BPF	1(+),2(*)	2(+), 2(*)	4(+), 4(*)	-0.137	-0.084	0.0467
MPEG MV	1(+), 4(*)	3(+), 7(*)	6(+), 14(*)	-0.238	-0.168	0.056
ARF	1(+), 2(*)	4(+), 2(*)	4(+), 4(*)	-0.181	-0.061	-0.094
DCT	4(+), 2(*)	4(+), 2(*)	7(+), 4(*)	-0.14	-0.122	-0.045
FIR	3(+), 2(*)	4(+), 4(*)	6(+), 6(*)	-0.116	-0.115	-0.051
WDF	2(+), 2(*)	2(+), 2(*)	4(+), 3(*)	-0.193	-0.161	-0.141

8.2.2 Comparison of Proposed Approach

As seen from Table 8.8, the proposed approach when compared with existing approaches gave better results. The existing approaches provide fault security however with no provision of guaranteeing that the solution abides the user budget of power and delay. This is due to the fact that [28], [30] are not able to generate a fault secured schedule for any number of resource instance (say single instance of each resource type). They at least need two instances of a resource type due to the compulsion of distinct hardware allocation. So, for a user which requires a transient fault secured datapath at the lowest hardware area (say single instance), approach [28] and [30] both will not be able to design one. Table 8.8 indicates the cost improvement of the proposed approach over [28] and [30] for various benchmarks for k_c = 1 (as multi cycle transient faults are not handled by [28] & [30]). As evident, the cost of final solution through proposed approach is significantly lower than [28] & [30].

8.3 Experimental Results: Multi-Cycle Single Event Transient Fault Security Aware MO-DSE for Single loop CDFGs in HLS

This section describes the experimental results of the proposed approach explained in Chapter 5 and the improvements obtained compared to recent approach [30]. The proposed approach has been implemented in java language on Intel core i5-2450M processor with 3MB L3 cache memory and 4GB DDR3 primary memory. The processor frequency is 2.5 GHz. Various HLS benchmarks were chosen for experimentation such as FIR [45], FFT [86, 100], DIFFERENTIAL EQUATION [45, 96, 100], MPEG MV[98, 99] , ARF [45, 86], WDF[45, 86]. The proposed approach can handle problems of any size. This section discusses the results in four phases:

- Variation of exploration time with change in swarm size
- Variation of exploration time with inertia weight
- Results of the proposed approach in terms of area occupied and execution delay of the final solution along with its associated final cost
- Comparison of proposed approach with [30] in terms of solution explored and final cost.

8.3.1 Effect of swarm size (*p*) Variation on Exploration Time

A larger swarm size covers larger design space during one iteration step (with a chance to get a better result) but is simultaneously subjected to increase in exploration time because of larger number of particles as well as greater computational complexity per iteration. On the contrary, a smaller swarm size needs more iteration to explore a better result for larger problem size. Therefore, three different swarm sizes have been analyzed and their impacts on exploration time are reported. *(Note:-based on this analysis, the selected swarm sizes for benchmarks used as our base line parameter are underlined).*

Table 8.9, presents the increase in exploration time with the increase in swarm size at the cost of no improvement in the final explored solution. In other words, final solution explored is optimal for all different swarm sizes. However, exploration time increases due to increase in computation complexity per iteration. As evident from Table 8.9, the best tradeoff between fast exploration and searching optimal solution can be obtained by setting p =3. For example, in case of MPEG MV, p = 3 gives a minimum exploration time of 16482 ms in comparison to p =5 and 7.

Table 8.9 Variation of Exploration Time with Swarm Size (p) in ms

Benchmark [45, 86, 98, 99, 100]	p =3	p =5	p =7
FIR	1216	1621	1853
FFT	3999	6370	7496
Differential	1819	1924	2415
MPEG MV	16482	24624	31604
ARF	17666	29993	43515
WDF	10911	17299	24781

Table 8.10 Exploration Time vs. Inertia Weight (at p =3)

Benchmark [45, 86, 98, 99, 100]	Linearly decreasing (ms)	ω=0.5 (ms)	ω =1.0 (ms)
FIR	1216	1259	1514
FFT	3999	4810	4827
Differential	1819	1776	1872
MPEG MV	16482	17194	18484
ARF	17666	19670	18881
WDF	10911	11736	11624

8.3.2 Results of Variation of Exploration Time with Inertia Weight

Inertia weight controls to the exploration drift process of the particle by weighing the involvement of the previous exploration drift. During the experiment, the following three variations of 'ω' have been analysed and its impact on the performance of exploration process has been reported:

- Linearly decreasing 'ω' in every iteration between [0.9- 0.1] throughout the exploration process.
- b) A constant value of ω = 1 throughout the exploration process.
- c) A constant value of ω = 0.5 throughout the exploration process.

Table 8.11 Experimental Results of the Proposed Approach for k_c = 1

Benchmark [45, 86, 98, 99, 100]	Final solution	A_{cons}	A_T^{DMR}	T_{cons} (us)	T_E^{DMR} (us)	Cost
FIR	2(+),3(*),1(<),UF=2	23058	16506	78.54	46.24	-0.208
FFT	2(+),2(-),4(*),1(<),UF=1	53739	36638	273.14	184.4	-0.218
Differential	1(+),2(-),5(*),1(<),UF=1	36379	24976	308.7	137.6	-0.262
MPEG MV	1(+), 4(*)	24000	13776	170	82.67	-0.240
ARF	1(+), 2(*)	15500	8092	220	178.7	-0.179
WDF	2(+), 2(*)	14000	10500	172	125.3	-0.196

Table 8.12 Experimental Results of the Proposed Approach for k_c = 4

Benchmark [45, 86, 98, 99, 100]	Final solution	A_{cons}	A_T^{DMR}	T_{cons} (us)	T_E^{DMR} (us)	Cost
FIR	1(+),4(*),1(<),UF=2	23058	16940	78.54	49.6	-0.189
FFT	3(+),2(-),3(*),1(<),UF=2	53739	36638	278.78	186.08	-0.222
Differential	1(+),2(-),6(*),1(<),UF=1	36379	27440	308.7	139.84	-0.238
MPEG MV	1(+), 4(*)	24000	13776	170	82.67	-0.24
ARF	1(+), 2(*)	15500	8092	220	178.7	-0.179
WDF	2(+), 2(*)	14000	10500	172	125.3	-0.196

As evident from Table 8.9, for all benchmarks the exploration time of proposed fault detectable DSE process is generally better with linearly decreasing value of 'ω'. For instance in Table 8.9, exploration time for FFT in case of linearly decreasing inertia weight (from 0.9 to 0.1) is 3999 ms and is much less as compared to exploration time of 4810mS and 4827 ms attained in constant inertia weight (ω =0.5 and ω =1). Similar trend is observed for other benchmarks. Further, in case of ARF, the exploration time for finding the optimal solution is 17666 ms when 'ω' is linearly decreased between [0.9 – 0.1] compared to 19670ms and 18881ms when 'ω' = 0.5 and 'ω' = 1 respectively as shown in Table 8.10.

8.3.3 Results of the proposed approach

As evident from Table 8.11 and 8.12 the solution explored by the proposed approach comprehensively meets the user defined constraints for power and execution time as well as

Table 8.13 Variation of Proposed Approach with [28]
Note: For proposed approach $\Phi_1 = \Phi_2 = 0.5$ in the fitness function

Benchmark [45, 86, 98, 99, 100]	Final solution (proposed)	Final solution [28]	Cost proposed	Final Cost [28]
FIR	2(+),3(*), 1(<),UF=2	2(+), 4(*), 1(<),UF=8	-0.208	-0.121
FFT	2(+),2(-), 4(*),1(<), UF=1	4(+), 2(-),4(*), 1(<),UF=3	-0.218	-0.15
Differential	1(+),2(-), 5(*),1(<), UF=1	2(+), 2(-),4(*), 1(<),UF=4	-0.262	-0.123
MPEG MV	1(+), 4(*)	3(+), 7(*)	-0.24	-0.168
ARF	1(+), 2(*)	4(+), 2(*)	-0.179	-0.061
WDF	2(+), 2(*)	2(+), 2(*)	-0.196	-0.161

minimizes the hybrid cost. The proposed approach was evaluated both for single cycle faults and multi cycle faults. (i.e. k_c = 1and k_c = 4). Further, the solutions obtained for the tested benchmarks are real optimal solutions which were verified by comparing with the golden solutions found by exhaustive analysis. For exploration, a swarm size of 3 was used also the acceleration coefficients were initialized to 2.0.

8.3.4 Comparison of proposed approach

The proposed approach has been compared with fault secured approach [28] in terms of final solution for faults secured DMR and its associated cost. Table 8.13 indicates the improvement in final solution cost of the proposed approach obtained over [28] for various benchmarks at k_c = 1 (*Note: k_c = 1 is only considered during comparison as multi cycle transient faults are not handled by [28]*). As evident, the cost of final solution found through proposed approach is significantly lower than [28].

8.4 Experimental Results: Bacterial Foraging Driven Exploration of Multi Cycle Fault Tolerant Datapath based on Power-Performance Tradeoff in High Level Synthesis

This section describes the experimental results of the proposed approach explained in Chapter 7 and the improvements obtained compared to recent approaches [28, 32]. The proposed MCFT BFOA-DSE and the approaches compared with [28, 32] all have been implemented in java and run on Intel Core-i5-3210M CPU with 3 MB L3 cache memory and 4 GB DDR3 primary memory during experimentation. The processor has frequency of 2.5 GHz. Experimentation was done for various HLS benchmarks such as IIR Butterworth filter [86], BPF[98,86], MPEG MV [45], ARF [45, 100], DCT [99], FIR [45, 100], WDF [86, 45]. The proposed approach can handle problem of any size. The library is given in chapter 3 Table 3.1.

From the results it has been observed that, the proposed approach has always yielded optimal results for all tested applications. Also, the results generated are fault tolerant in nature. This chapter covers the following details:

- Results and comparison of proposed algorithm and existing approach [28, 32], generated by varying the k_c fault value.
- Comparison of MCFT-BFOA-DSE with previous Fault tolerant approaches in terms of Quality of Results and resource set utilized.

The QoR is calculated as:

$$QoR = \frac{1}{2}\left(\frac{P_T^{DMR}}{P_{\max}^{DMR}} + \frac{T_E^{DMR}}{T_{\max}^{DMR}} \right) \tag{8.2}$$

8.4.1 Pre-tuning of Parameters

During experimentation, following settings were made for fault tolerant approach of design exploration: $\varphi_1 = \varphi_2 = 0.5$, $p = 3$, $N_c = 120$, $N_{ed} = 5$.

8.4.2 Comparison of Proposed Approach with [32]

This section describes the results obtained by comparing the proposed approach with existing approach [32].

Table 8.14 Results of Proposed Fault Tolerant DSE Approach for $k_c = 1$

Benchmark [45, 86, 98, 99, 100]	T_{cons} (us)	T_E^{DMR} (us)	P_{cons} (uW)	P_T^{DMR} (uW)	Cost
IIR	66.00	55.00	308.90	205.96	-0.151
BPF	173.00	149.70	301.19	205.59	-0.155
MPEG_MV	165.00	82.90	727.78	352.25	-0.276
ARF	220.00	179.50	450.97	205.93	-0.217
DCT	207.00	154.54	450.37	384.65	-0.132
FIR	108.00	89.60	596.19	325.20	-0.180
WDF	172.00	137.60	330.00	265.19	-0.145

Table 8.15 Comparison of Proposed Approach with [32] in Terms of Resource (Hardware) Utilized for Fault Tolerant Datapath for ($k_c = 1$)

Benchmark [45, 86, 98, 99, 100]	Resource found [32] $k_c = 1$	Resource utilized $k_c = 1$	% reduction in area
IIR	3(+), 5(*)	1(+), 2(*)	62.5
BPF	6(+), 5(*)	1(+),2(*)	72.7
MPEG_MV	9(+), 14(*)	1(+), 4(*)	78.2
ARF	5(+), 6(*)	1(+), 2(*)	72.7
DCT	12(+),6(*)	4(+), 2(*)	66.6
FIR	9(+), 8(*)	3(+), 2(*)	70.5
WDF	6(+), 4(*)	2(+), 2(*)	60.0

8.4.2.1 Analysis of Results by Varying k_c Value

The proposed approach when compared with [32] for different k_c values, gave substantially better results in terms of cost of solutions and the resource solutions obtained. Table 8.14, illustrates the results obtained by making the design system tolerable to 1-cycle faults i.e., for $k_c = 1$. As evident from Table 8.15, the applications when tested through the proposed approach require less hardware usage than the existing approach [32]. In [32], the hardware usage is almost (sometimes more than) tripled for most of the applications. For instance, in BPF application, the proposed approach generated 1(+), 2(*) as the final solution, which has

Table 8.16 Results of Proposed Fault Tolerant DSE Approach for $k_c = 2$

Benchmark [45, 86, 98, 99, 100]	T_{cons} **(us)**	T_E^{DMR} **(us)**	P_{cons} **(uW)**	P_T^{DMR} **(uW)**	**Cost**
IIR	66.00	66.00	308.94	205.65	-0.101
BPF	179.00	149.70	301.11	205.59	-0.168
MPEG_MV	165.00	93.60	727.78	351.82	-0.258
ARF	221.00	179.70	450.95	205.93	-0.218
DCT	213.00	165.50	450.26	384.52	-0.124
FIR	108.00	89.80	596.19	325.20	-0.180
WDF	172.00	137.60	330.00	265.19	-0.145

Table 8.17 Results of Proposed Fault Tolerant DSE Approach For $k_c = 3$

Benchmark [45, 86, 98, 99, 100]	T_{cons} **(us)**	T_E^{DMR} **(us)**	P_{cons} **(uW)**	P_T^{DMR} **(uW)**	**Cost**
IIR	66.00	66.00	350.00	205.65	-0.148
BPF	173.00	150.00	301.18	205.58	-0.152
MPEG_MV	165.00	93.60	727.78	351.82	-0.258
ARF	221.00	180.00	450.95	205.92	-0.217
DCT	213.00	165.80	450.26	384.52	-0.124
FIR	108.00	90.10	596.19	325.20	-0.177
WDF	172.00	137.60	330.00	265.19	-0.145

a cost of -0.155. While, [32] yielded a solution set of 6(+), 5(*) which is much higher (more than triple) compared to proposed. This shows that since the hardware usage is much greater in [32], therefore, the solutions obtained do not satisfy the power budget or the time budget, thereby, generating higher cost of the solution (design). On the other hand, the final solutions of the proposed approach involving DMR are fault tolerant and satisfy the user specified power and time constraints.

Table 8.18 Comparison of Proposed Approach With [32] in Terms of Resource (Hardware) Utilized for Fault Tolerant Datapath for (k_c = 2 and 3)

Benchmark [45, 86, 98, 99, 100]	**Resource found [32] k_c = 2**	**Resource utilized k_c = 2**	**% reduction in area**	**Resource found [32] k_c = 3**	**Resource utilized k_c = 3**	**% reduction in area**
IIR	3(+), 5(*)	1(+), 2(*)	62.5	2(+), 5(*)	1(+), 2(*)	57.1
BPF	6(+), 4(*)	1(+),2(*)	70.0	6(+), 4(*)	1(+),2(*)	70.0
MPEG_MV	9(+),14(*)	1(+), 4(*)	78.2	8(+), 14(*)	1(+), 4(*)	77.2
ARF	5(+), 6(*)	1(+), 2(*)	72.7	6(+), 6(*)	1(+), 2(*)	75.0
DCT	12(+),6(*)	4(+), 2(*)	66.6	12(+),6(*)	4(+), 2(*)	66.6
FIR	8(+), 8(*)	3(+), 2(*)	68.7	8(+), 8(*)	3(+), 2(*)	68.7
WDF	6(+), 5(*)	2(+), 2(*)	63.6	6(+), 5(*)	2(+), 2(*)	63.6

Table 8.19 Results of Proposed Approach (for k_c = 1) in Terms of Optimality

Benchmarks [45, 86, 98, 99, 100]	**GD**	**MFE**	**Spacing (*S*)**	**Spread (Δ)**	**Weighted Metric (W_m)**
IIR	0.00	0.32	0.00	0.60	0.30
BPF	0.00	0.26	0.11	0.84	0.42
MPEG_MV	0.00	0.54	0.18	0.89	0.45
ARF	0.00	0.65	0.07	0.78	0.39
DCT	0.00	0.14	0.00	0.66	0.33
FIR	0.00	0.33	0.01	0.63	0.31
WDF	0.00	0.46	0.00	0.74	0.37

Similar results are observed for the approaches when tested for multi-cycle faults in the system. Table 8.16 and Table 8.17, shows the results of the proposed for 2-cycle (k_c = 2) and 3-cycle faults (k_c = 3). As seen from Table 8.18, the proposed approach generates more efficient results than the [32] approach against faults with k_c = 2 and 3. The solutions generated through proposed approach have much lower cost, then the [32] approach employing TMR scheme.

Table 8.20 Results of Proposed Approach (for k_c = 3) in Terms of Optimality

Benchmarks [45, 86, 98, 99, 100]	GD	MFE	Spacing (S)	Spread (Δ)	Weighted Metric (W_m)
IIR	0.00	0.17	0.00	0.66	0.33
BPF	0.00	0.27	0.11	0.85	0.42
MPEG_MV	0.00	0.98	0.03	0.82	0.41
ARF	0.00	0.68	0.00	071	0.35
DCT	0.00	0.34	0.10	0.81	0.40
FIR	0.00	0.42	0.00	0.70	0.35
WDF	0.00	0.47	0.04	0.73	0.36

Table 8.21 Comparison of Proposed Approach With [32] Fault Tolerant Approach

Benchmark [45, 86, 98, 99, 100]	k_c = 1			k_c = 2		
	QoR proposed	QoR [32]	% Improve-ment	QoR proposed	QoR [32]	% Improve-ment
IIR	0.46	0.76	39.4	0.51	0.86	40.5
BPF	0.55	1.01	45.5	0.55	1.00	44.9
MPEG_MV	0.26	0.66	60.6	0.28	0.69	58.8
ARF	0.38	0.65	41.5	0.38	0.68	42.9
DCT	0.51	0.91	43.9	0.53	0.95	44.1
FIR	0.40	0.65	38.4	0.40	0.68	40.5
WDF	0.58	0.91	36.2	0.58	0.98	40.5

For k_c = 1 average improvement in QoR w.r.t [32] = 43.4 %
For k_c = 2 average improvement in QoR w.r.t [32] = 44.3 %

8.4.2.2 Results of Proposed Approach in Terms of Optimality

Table 8.19 and Table 8.20 show the analysis of proposed approach in terms of quality metrics such as generational distance (GD), maximum pareto-optimal front error (MFE), spacing (S), spread (Δ) and weighted sum (W). Table 8.19 illustrates the results of proposed approach in terms of optimality for k_c = 1, while Table 8.20 is for k_c = 3.

Table 8.22 Comparison of Proposed Approach With [32] Fault Tolerant Approach

Benchmark [45, 86, 98, 99, 100]	k_c = 3		
	QoR proposed	QoR [32]	% Improve-ment
IIR	0.51	0.80	36.2
BPF	0.55	1.05	47.6
MPEG_MV	0.28	0.67	58.2
ARF	0.38	0.73	47.9
DCT	0.53	0.99	46.4
FIR	0.40	0.70	42.8
WDF	0.58	1.03	43.6

For k_c = 3 average improvement in QoR w.r.t [32] = 45.8 %

As seen from Table 8.19, the GD is zero for all the benchmarks, indicating that the solutions generated through the proposed approach lie on true pareto front. A spacing of zero (or a little higher than zero) for an application states that, the proposed approach is able to have uniform distribution of Pareto points on the curve. Similar pattern of results is evident from Table 8.20 where optimality of proposed approach for k_c = 3 is evaluated. It can also be seen that, the results obtained by the proposed approach are real optimal solution as discovered by verifying with the golden solution found through exhaustive analysis.

8.4.3 Comparison of Proposed Approach in Terms of Quality of Results

8.4.3.1 Comparison with [32]

Table 8.21 and 8.22 shows the comparison of [32] with the proposed MCFT-BFOA driven DSE approach. In approach [32] there was no concept of exploration of a fault tolerant datapath based on power-performance constraint presented in the paper, unlike the proposed approach. Further, the authors did not provide any concept of multi-cycle faults. Moreover, the approach presented a TMR (triple modular redundant) system for *k*-cycle fault tolerance for single event transient (SET). The outputs of the units were voted upon by the help of voter, to mask the errors. Additionally, comparators were used to detect the difference in the outputs of the units. However, the proposed approach uses Double Modular Redundancy (DMR) scheme to explore a fault tolerant design without using voters to extract the correct

Table 8.23 Comparison of Proposed Approach with [28] Fault Secured Approach

Benchmark [45, 86, 98, 99, 100]	QoR proposed	QoR [28]	Resource Set (proposed)	Resource Set [28]	% reduction in area
IIR	0.46	0.49	1(+), 2(*)	1(+), 3(*)	25.0
BPF	0.55	0.58	1(+),2(*)	2(+), 2(*)	25.0
MPEG_MV	0.26	0.33	1(+), 4(*)	3(+), 7(*)	50.0
ARF	0.38	0.50	1(+), 2(*)	4(+), 2(*)	50.0
DCT	0.51	0.51	4(+), 2(*)	4(+), 2(*)	0.0
FIR	0.40	0.40	3(+), 2(*)	4(+), 4(*)	37.5
WDF	0.58	0.57	2(+), 2(*)	2(+), 2(*)	0.0

Average improvement in QoR = 7.10%

output. Therefore, [32] involved higher degree of redundancy in their system which sometimes involved a TMR system almost tripling the resource usage.

Therefore, from Table 8.21 and 8.22 it is evident that proposed approach achieves a better QoR in comparison to [32] for all the benchmarks. The average improvement in QoR is more than 43%. Also, an average reduction of 70% is attained in the hardware usage of proposed approach as observed in Table 8.21 and 8.22.

8.4.3.2 Comparison with [28]

Table 8.23 shows the comparison of QoR (cost units) between the proposed approach (MCFT-BFOA-DSE) and fault secured approach [28]. It should be noted that [28] is just a fault secured approach and does not have ability to mask the fault. Therefore it has ability to only detect the fault but not correct it. Moreover, [28], does not have ability to explore a datapath circuit based on conflicting user constraint. After experimentation, it has been found that there is an average improvement in QoR of 7% and a reduction of 29.1 % in hardware usage through proposed approach. For example, in case of DCT and WDF, there is no reduction in hardware area observed compared to [28], however, the proposed approach with the same resource achieves fault tolerance as well as minimizes the hybrid cost of power and execution time which [28] is unable to perform.

8.5 Experimental Results: Untrusted Third Party Digital IP cores: Power-Delay Trade-off Driven Exploration of Hardware Trojan Secured Datapath during High Level Synthesis

The proposed approach as well as [35] both have been implemented in java and run on Intel Core-i5-3210M CPU with 3MB L3 cache memory, 4GB DDR3 primary memory and processor frequency of 2.5 GHz. An average of 10 runs was reported for proposed BFOA DSE with equal weightage to both user objectives of power and delay (W_1 = W_2 = ½). Various HLS benchmarks were chosen for experimentation such as JPEG Downsample, JPEG IDCT2, IDCT, Feedback Points, ARF, BPF, FIR, and MESA Matrix Multiplication. As found during the experiments, the proposed approach is scalable and is able to handle problems of any size. The results are divided into three phases.

- Sensitivity Analysis
- Results of proposed approach
- Comparison of proposed approach with existing approaches.

8.5.1 Sensitivity Analysis

8.5.1.1 Pre-tuning

During experimentation, following settings were kept for proposed approach: p= 3, 5 and 7, N_c = 120.

8.5.1.2 Bacterium Size, *p*

Table 8.24 shows the effect of bacterium size 'p' on the exploration time of proposed DSE method. As evident, it indicates that for all benchmarks with the increase in bacterium size, the exploration time of the proposed approach to find the final solution increases (with the cost of the final solution remaining the same for all bacterium size). The exploration time increase is because of increase in computational complexity per iteration (i.e. the total number of positions evaluated in a run increases with the increase in 'p'). Figure 8.5 and 8.6 shows a graphical representation of the variation of exploration time with respect to increase in the bacterium size 'p'.

Table 8.24 Comparison of Exploration Time with respect to Bacterium size '*p*' for Proposed Approach

Benchmark [45, 86, 98, 99, 100]	Bacterium Size	Exploration time (ms)	Cost of final solution
IIR	3	640	-0.125
	5	703	-0.125
	7	1250	-0.125
MPEG MV	3	7043	-0.251
	5	7657	-0.251
	7	11422	-0.251
ARF	3	1907	-0.192
	5	2156	-0.192
	7	3786	-0.192
IDCT	3	7156	-0.154
	5	7998	-0.154
	7	8328	-0.154
DCT	3	3516	-0.106
	5	3891	-0.106
	7	5977	-0.106
FIR	3	6500	-0.245
	5	7282	-0.245
	7	12532	-0.245

8.5.2 Results of proposed Approach

As shown Table 8.25, the proposed approach comprehensively meets the user constraints of delay and power (and minimizes the hybrid cost for all benchmarks. For example, in case of IIR benchmark, the proposed approach generates the final optimal solution with power (P_T^{DMR}) = 0.58mW and L_T^{DMR} = 23080ns, which is with the specified user constraints of power and delay (P_{cons} = 0.55mW and L_{cons} = 38945 ns). Also, the proposed approach is able

to achieve the real optimal solution for all benchmarks as verified with the golden solution found through brute force.

8.5.3 Comparison of proposed approach

Metric such as QoR indicating the quality of final solution (lower cost solution explored)

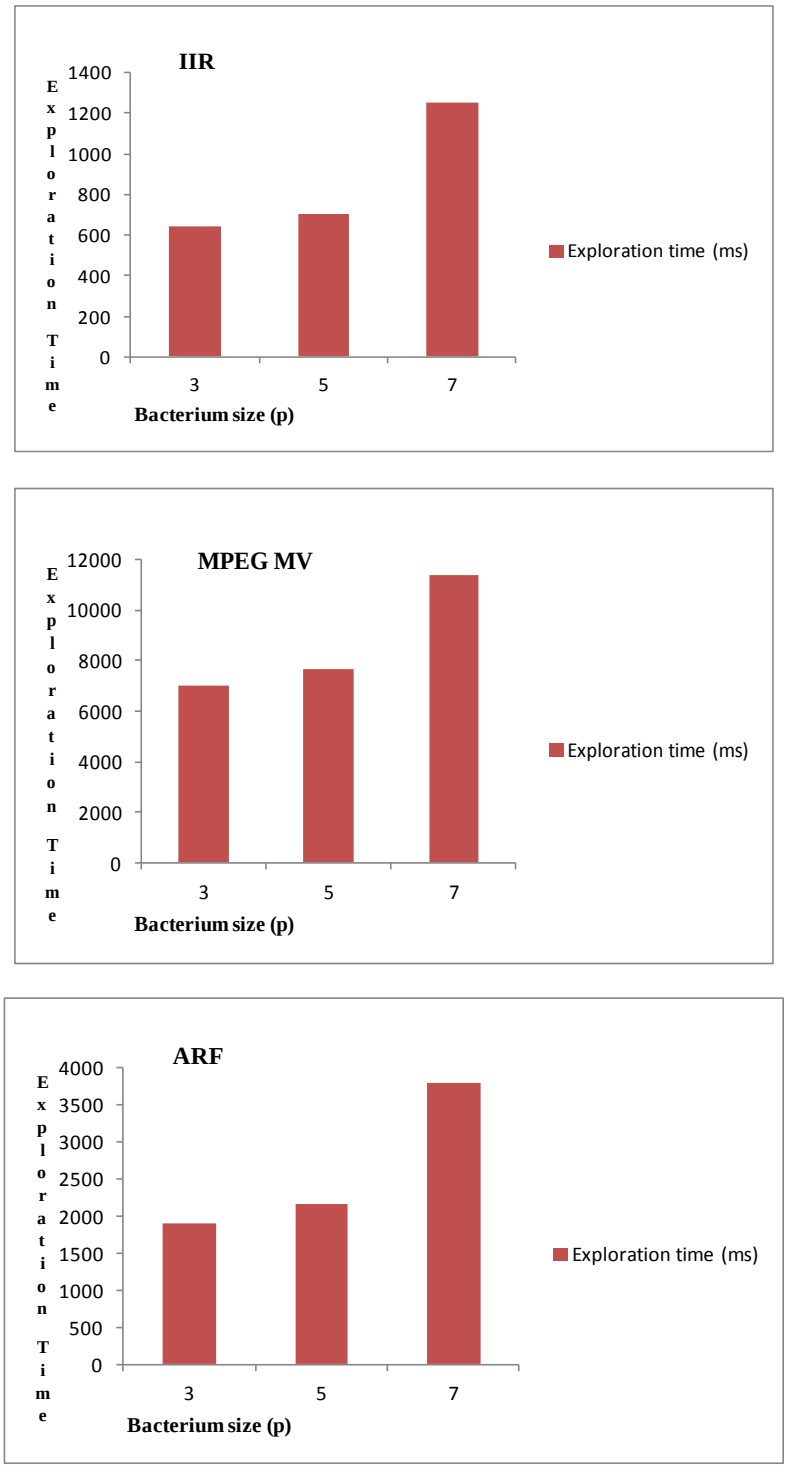


Figure 8.5 Graphical Representation of Variation of Exploration Time (in ms) with respect to Change in Bacterium size (*p*)

yielded by both approaches (proposed and [35]) is an important tool for comparison. The QoR for both the approaches (proposed and [35]) is evaluated by the following function:

$$QoR = \frac{1}{2}\left(\frac{P_T^{DMR}}{P_{\max}^{DMR}} + \frac{L_T^{DMR}}{L_{\max}^{DMR}}\right) \tag{8.3}$$

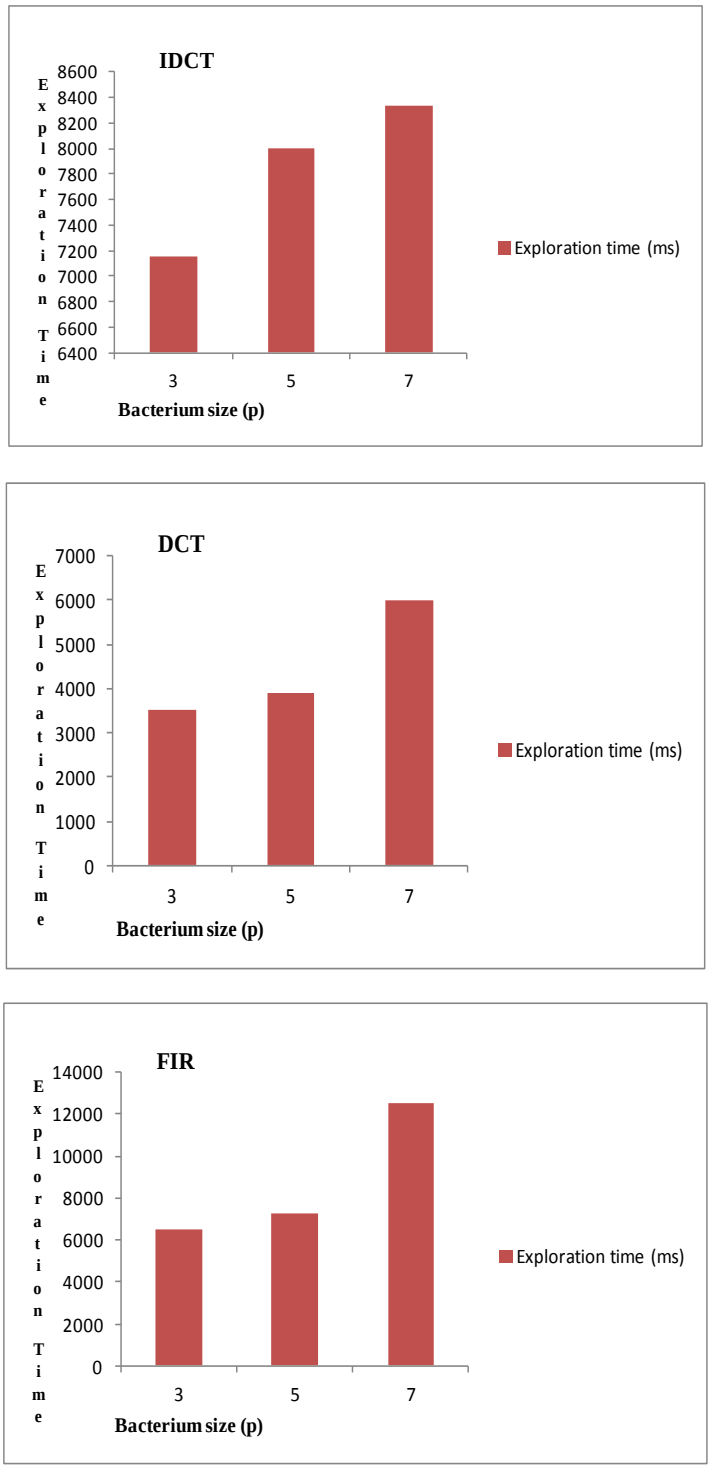


Figure 8.6 Graphical Representation of Variation of Exploration Time (in ms) with respect to Change in Bacterium size (*p*)

Table 8.25 Results of Proposed Trojan Secured Approach

Benchmark [45, 86, 98, 99, 100]	L_{cons} (ns)	L_T^{DMR} (ns)	P_{cons} (mW)	P_T^{DMR} (mW)	Cost
IIR	38945	23080	0.58	0.58	-0.125
IDCT	119160	77080	1.02	0.93	-0.154
ARF	130810	89890	0.63	0.48	-0.192
MPEG MV	88307	36240	1.48	1.03	-0.251
DCT	175442	153540	0.77	0.59	-0.106
FIR	76387	34890	1.22	0.85	-0.245

Note: mW = miliwatt, ns = nanoseconds

Table 8.26 Comparison of Proposed Approach With [35]

Benchmark [45, 86, 98, 99, 100]	Final solution for Trojan Secured datapath (proposed)	Final solution for Trojan Secured datapath [35]	Cost of final solution (proposed)	Cost of final solution [35]	QoR in cost units (proposed)	QoR in cost units [35]
IIR	2(+), 5(*), 0	2(+), 3(*),1	-0.125	-0.016	0.53	0.64
IDCT	6(+), 4(*), 0	5(+), 3(*),1	-0.154	-0.027	0.50	0.63
ARF	2(+), 4(*),0	3(+), 3(*),1	-0.192	-0.056	0.49	0.63
MPEG MV	2(+), 10(*), 0	3(+), 8(*),1	-0.251	-0.226	0.30	0.33
DCT	4(+), 4(*), 0	5(+), 3(*),1	-0.106	-0.064	0.50	0.54
FIR	6(+), 6(*), 0	5(+), 5(*),1	-0.245	-0.209	0.33	0.36

The area of the single comparator/error detection block responsible to runtime Trojan detection at the final output is also considered in the above QoR function when evaluating its magnitude for both [35] and proposed approach. However, since only a single comparator/error detection block is used in both approaches, hence it has no impact on the QoR of both approaches. However, during QoR comparison, power overhead due to internal buffering (temporary storage of operation output), has been considered for both proposed approach and [35].

Table 8.26, illustrates the comparative results of the proposed approach and [35] when evaluated on the standard benchmarks. As seen from the results in Table 8.26, with the introduction of exploration for vendor allocation procedure type 'A_v' and user constraint

driven exploration, the proposed approach generates better results in comparison to [35]. For example, in ARF benchmark, the proposed approach generates 2(+), 4(*), 0 as the solution (cost of -0.192) which is lesser than the cost of [35] (cost = -0.056). This is because, in previous approach

there is no provision of exploring an optimal ‘vendor allocation procedure’ during scheduling in DMR as well as no optimization scheme based on user power- delay constraint for finding a better alternative solution. Figure 8.7 shows the comparison of the QoR (in cost units) of the proposed approach with [35].

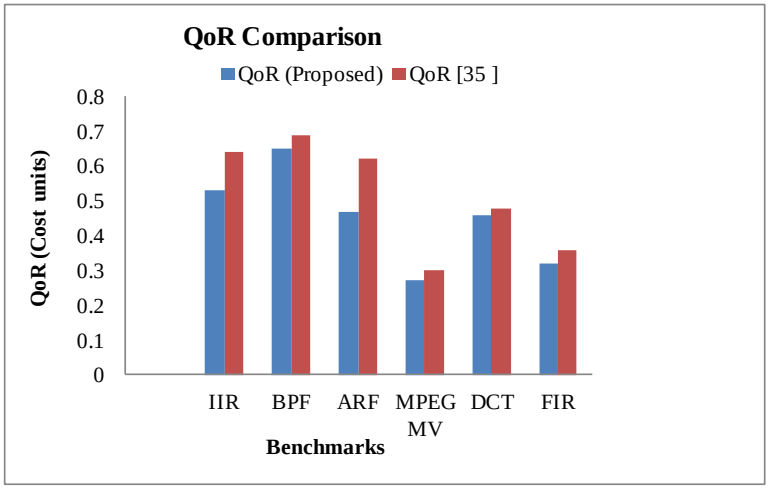


Figure 8.7 Comparison of QoR (cost units) of proposed and [35] approach

Chapter 9

Conclusion and Future work

9.1 Conclusion

This thesis presented novel methodologies for designing reliability aware and hardware security aware designs at behavioural level for data intensive and control intensive applications during design of application specific datapath processor. Therefore, the following objectives were accomplished in this thesis work:

- Proposed a methodology to solve the problem of DSE during power-performance trade-off for data intensive applications that produces high quality design solutions. The proposed approach provided an average improvement in QoR of > 35% and reduction in runtime of > 4% compared to recent approaches.
- Proposed an approach to solve the problem of exploration of low cost optimal *k*-cycle transient fault secured datapath during power-performance trade-off for data intensive applications. Results of comparison of proposed approach with recent approaches indicated significant reduction of final cost.
- Proposed an automated approach to solve the problem of simultaneous exploration of low cost optimal *k*-cycle transient fault secured datapath and unrolling factor for control intensive applications during area-delay trade-off. Results of proposed approach when compared to similar approach indicated better quality solution within acceptable runtime.
- Proposed an execution time prediction model for faster exploration process in case of single loop based CDFGs without tediously unrolling CDFG loop completely.
- Proposed an approach to solve the problem of exploration of low cost optimal *k*-cycle transient fault tolerant datapath based on power-performance tradeoff for data intensive applications. The results in chapter 8 showed that the proposed MCFT-BFOA based DSE provided higher an average reduction of 7% in final cost and 29% in hardware utilization compared to recent approaches.
- Proposed an approach that solves the problem of exploration of low cost optimal Trojan secured datapath during behavioural synthesis for data intensive applications.

The proposed approach achieves an improvement of 14.1% in QoR in comparison to existing approaches.

Therefore, this thesis presented multiple novel methodologies for designing reliability aware and hardware security aware designs at behavioural level for data intensive and control intensive applications during design of application specific datapath processor. The proposed methodologies can be efficiently applied for any exploration problem in HLS based on any user criteria.

9.2 Future Work

However, generating highly reliable and secured HLS designs for application specific processors still requires a lot of effort in the future. Some of the important aspects which require future attention by the researchers area as follows:

- Consideration of multi-checkpointing technique can be considered during transient fault security in HLS.
- Development of low cost Trojan secured schedule for nested-loop based applications.
- Consideration of other class of Trojans than the one handeled in this thesis, during development of Trojan security aware HLS methodology.

References

[1] Mohanty S. P., Ranganathan N., Kougianos E., Patra, P. (2008). Low-power highlevel synthesis for nanoscale CMOS circuits. Springer Science & Business Media.

[2] Marwedel P. (1984). The MIMOLA design system: Tools for the design of digital processors. In Proceedings of Design Automation Conference, pp. 587–593.

[3] Granacki J., Knapp D., Parker A. (1985). The ADAM advanced design automation system: Overview, planner and natural language interface. In Proceedings of Design Automation Conference, pp 727–730.

[4] Jain R., Kucukcakar K., Mlinar M., Parker A. (1989). Experience with the ADAM synthesis system. In Proceedings of Design Automation Conference, pages 56–61.

[5] Paulin P. G., Knight J. P., Girczyc E. F. (1986). Hal: a multi-paradigm approach to automatic data path synthesis. In Proceedings of Design Automation Conference, pp. 263– 270.

[6] Chandrakasan A. P., Potkonjak M., Rabaey J., Brodersen R. W. (1992). HYPERLP: a system for power minimization using architectural transformations. In Proc. Int. Conf. on Computer-Aided Design, pp. 300–303.

[7] Micheli G. De, D. Ku. (1988). HERCULES-a system for high-level synthesis. In Proceedings of Design Automation Conference, pp. 483–488.

[8] Micheli G. De, Ku D., Mailhot F., Truong T. (1990, Oct). The Olympus synthesis system. IEEE Design & Test of Computers, 7(5):37–53.

[9] Rabaey J. M., Chu C., Hoang P., Potkonjak M. (1991, April). Fast prototyping of datapathintensive architectures. IEEE Design & Test of Computers, 8(2):40–51.

[10] Elliott J. P.(1999). Understanding Behavioral Synthesis: A Practical Guide to High-Level Design. Kluwer Academic Publishers.

[11] Knapp D. W. (1996). Behavioral synthesis: digital system design using the synopsys behavioral compiler. Prentice-Hall, Inc.

[12] Kress R., Pyttel A., Sedlmeier A. (2000). FPGA-based prototyping for product definition. In Proceedings of Int. Conference on Field Programmable Logic and Applications, pp. 78–86.

[13] Gerez, S.H. (2004) Algorithms for VLSI Design Automation. Wiley

[14] Paulin P.G., Knight J.P. (1989) Force directed scheduling for the behavioral synthesis of ASICs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 8(6), pp.661–679.

[15] Mohanty, S.P. (2003) Energy and Transient Power Minimization During Behavioral Synthesis. Ph.D. thesis, University of South Florida.

[16] Mohanty, S.P., Rangnathan, N., Chappidi, S.K. (2003) An ILP-based scheduling scheme for energy efficient high performance datapath synthesis. In: Proceedings of the International Symposium on Circuits and Systems (ISCAS), pp. 313–316.

[17] Ram, D. S., Bhuvaneswari, M. C., & Prabhu, S. S. (2012). A novel framework for applying multi objective GA and PSO based approaches for simultaneous area, delay, and power optimization in high level synthesis of datapaths. VLSI design, 2012, pages-12.

[18] Heijlingers MJM, Cluitmans LJM, Jess JAG. (1995). High-level synthesis scheduling and allocation using genetic algorithms. In: Proceedings of Asia South Pacific design automation conference, pp. 61–66.

[19] Palermo G, Silvano C, Zaccaria V. (2008). Discrete particle swarm optimization for multi-objective design space exploration. In: Proceedings of the 11th EUROMICRO IEEE digital system design architectures, methods and tools, pp. 641–644.

[20] Krishnan V, Katkoori S. (2006). A genetic algorithm for the design space exploration of datapaths during high-level synthesis. IEEE Trans. on Evolutionary Computation, Vol.10, No.3.

[21] Sengupta A, Sedaghat R, Sarkar P. (2012). A multi structure genetic algorithm for integrated design space exploration of scheduling and allocation in high level synthesis for DSP kernels. Elsevier Journal on Swarm and Evolutionary Computation, Vol. 7, pp. 35–46.

[22] Haubelt C, Schlichter T, Keinert J, Meredith M. (2008). SystemCoDesigner: Automatic design space exploration and rapid prototyping from behavioral models. In: Proceedings of the 45th annual ACM/ IEEE Design Automation Conference, pp. 580-585.

[23] Coussy P, Chavet C, Bomel P, Heller D, Senn E, Martin E. (2008). GAUT: A High-Level Synthesis tool for DSP applications. In High-Level Synthesis From Algorithm to Digital Circuits, Springer Netherlands, pp. 147-69.

[24] Canis A, Choi J, Aldham M, Zhang V, Kammoona A, Czajkowski T, Brown S D, Anderson J H. (2013). LegUp: An open-source high-level synthesis tool for FPGA-

based processor/accelerator systems. In: Proceedings of ACM Trans. Embedd. Comput. Syst., Vol.13, No. 2, Article 24, 27 pages.

[25] Villarreal J, Park A, Najjar W, Halstead R. (2010). Designing modular hardware accelerators in C with ROCCC 2.0. In: Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines, pp. 127–34.

[26] CatapultC : *//calypto.com/en/products/catapult/overview*

[27] Cong J., Bin L., Neuendorffer S., Noguera J., Vissers K., Zhang Z. (2011, April) High-Level Synthesis for FPGAs: From Prototyping to Deployment. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 30, No.4, pp. 473-91.

[28] Karri R., Orailoglu A. (1996). Time-constrained scheduling during high-level synthesis of fault-secure VLSI digital signal processors. In Proceedings of IEEE Transactions on Reliability, Vol.45, No.3, pp.404-412.

[29] Karri R., Orailoˇglu A. (1992). Transformation-based high-level synthesis of fault tolerant ASICs. In IEEE Design Automation Conference, pp. 662–665.

[30] Wu K., Karri R. (2004). Fault Secure Datapath Synthesis Using Hybrid Time and Hardware Redundancy. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol.23, No.10, pp.1476-1485.

[31] Wu K., Karri R. (2001, Nov). Algorithm level recomputing—a register transfer level concurrent error detection technique. In Proc. IEEE/ACM Int. Conf. Computer-Aided Design, pp. 537–543.

[32] Inoue T., Henmi H., Yoshikawa Y., Ichihara H. (2011). High-level synthesis for multi-cycle transient fault tolerant datapaths. In Proceedings of the 17th IEEE international on-line testing symposium, pp 13–18.

[33] Bhunia S., Abramovici M., Agrawal D., Bradley P., Hsiao M., Plusquellic J., Tehranipoor M. (2013). Protection against hardware Trojan attacks: Towards a comprehensive solution. In Proceedings of IEEE Design & Test, Vol. 99, pp. 1–1.

[34] Zhang X., Tehranipoor M. (2011). Case study: Detecting hardware Trojans in third-party digital IP cores. In Proc. of IEEE International Symposium on Hardware-Oriented Security and Trust, pp. 67–70.

[35] Rajendran J., Huan Z., Sinanoglu O., Karri R. (2013). High-level synthesis for security and trust. In Proceedings of 19th IEEE Intl On-Line Testing Symposium (IOLTS), pp. 232-233.

[36] Sengupta A, Sedaghat R. (2011). Integrated scheduling, allocation and binding in high level synthesis, using multi structure genetic algorithm based design space exploration system. In: Proceedings of the 12th IEEE/ ACM international symposium on quality electronic design (ISQED), California, USA, pp. 486–94.

[37] Passino K. M. (2002). Biomimicry of Bacterial Foraging for Distributed Optimization and Control. IEEE Control Systems Magazine, pp. 52-67.

[38] Pierucci O. (1972, Feb). Chromosome replication and cell division in Escherichia coli at various temperatures of growth. J. Bacteriol, Vol.109, No.2, pp. 848–854.

[39] Maeda K, Imae Y, Shioi J I, Oosawa F. (1976, Sept). Effect of Temperature on Motility and Chemotaxis of Escherichia coli. J. Bacteriol. Vol. 127, No. 3, pp. 1039-46.

[40] Doyle M.P., Shoeni J.L. (1984). Survival and growth characteristics of Escherichia coli associated with hemorrhagic colitis. Applied Environmental Microbiology, Vol.48, pp. 855-856.

[41] Adler J., Templeton B. (1967, Feb). The Effect of Environmental conditions on the Motility of Escherichia coli. Microbiology. Vol. 46, No. 2, pp. 175-84.

[42] 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, Vol.50, No.3, pp. 424–37.

[43] Chang, Jui-Ming, Massoud P. (1997) Energy minimization using multiple supply voltages, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.5, No. 4, pp. 436-443.

[44] Das S., Biswas A., Dasgupta S., Abraham A. (2009). Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications. Studies in Computational Intelligence, Vol. 203, pp. 23-55.

[45] University of California, Santa Barbara, Express Benchmarks: http://express.ece.ucsb.edu/benchmark/.

[46] Gajski D., Dutt N.D., Wu A., Lin S. (1992). High level synthesis: introduction to chip and system design. Norwell (MA): Kluwer Publishers, pp. 52-67.

[47] Lakshminarayana G., Raghunathan A., Jha N. K. (2000, Sep). Behavioral synthesis of fault secure controller/datapaths based on aliasing probability analysis. IEEE Trans. Comput., Vol. 49, pp. 865–885.

[48] Normand E. (1996, April). Single-event effects in avionics. IEEE Trans. Nucl. Science, Vol. 43, pp. 461–474.

[49] Lala P.K. (1985). Fault-Tolerant and Fault-Testable Hardware Design, Prentice-Hall.

[50] Choi Y.H., Malek M. (1988, May). A fault-tolerant FFT processor. IEEE Trans. Computers, Vol 37, pp 617-621.

[51] Jou J.Y., Abraham J.A. (1988, May). Fault-tolerant FFT networks", IEEE Trans. Computers, vol 37, 1988 May, pp 548-561.

[52] R. Geist, Trivedi K. (1990, July). Reliability estimation of fault-tolerant systems: Tools and techniques. IEEE Computer, Vol 23, pp 52-61.

[53] Eberhart R.C, Shi Yuhui. (2001). Particle swarm optimization: developments, applications and resources. In Proceedings of the 2001 Congress on Evolutionary Computation, pp. 81-86.

[54] Mishra V.K., Sengupta A. (2014, Jan). MO-PSE: Adaptive Multi Objective Particle Swarm Optimization Based Design Space Exploration in Architectural Synthesis for Application Specific Processor Design. Elsevier Journal on Advances in Engineering Software, Vol. 67, pp. 111 - 124.

[55] Sengupta A., Mishra V.K. (2014, March). Integrated Particle Swarm Optimization (i-PSO): An Adaptive Design Space Exploration Framework for Power-Performance Tradeoff in Architectural Synthesis. In Proceedings of IEEE 15th International Symposium on Quality Electronic Design (ISQED 2014), California, USA, pp.60 – 67.

[56] Trelea, Cristian I. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. Elsevier Information processing letters, Vol. 85 No. 6, pp. 317-325.

[57] Sengupta A., Mishra V.K. (2014, Aug). Automated Exploration of Datapath and Unrolling Factor during Power-Performance Tradeoff in Architectural Synthesis Using Multi-Dimensional PSO Algorithm. Elsevier Journal on Expert Systems With Applications, Vol. 41, No. 10, pp. 4691- 4703.

[58] Lisboa C. A., Carro L. (2007, May). System Level Approaches for Mitigation of Long Duration Transient Faults in Future Technologies. In Proceedings of 12th IEEE European Test Symposium – ETS 2007, in Freiburg, Germany, pp. 165 – 172.

[59] Dodd P. E, et al. (2004). Production and propagation of Single-Event Transients in High-Speed Digital Logic ICs. IEEE Transactions on Nuclear Science, Vol. 51, No 6, Part 2, pp. 3278-3284.

[60] Coussy P., Morawiec A. (2008). High-Level Synthesis: From Algorithm to Digital Circuits. Springer, Berlin, Germany.

[61] Micheli G. D. (1994). Synthesis and optimization of digital circuits. McGraw-Hill Higher Education. New York.

[62] Yassa F. F., Jasica J. R., Hartley R. I., Noujaim S. E. (1987). A silicon compiler for digital signal processing: Methodology, implementation, and applications. In Proceedings of IEEE, 75(9), pp. 1272-1282.

[63] Weste N., Harris D. (2010). CMOS VLSI Design: A Circuits And Systems Perspective Author: Neil Weste, David Harris, Publisher: Addison We, pp. 1-5.

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

[65] Torbey E., Knight J. (1998, May). High-level synthesis of digital circuits using genetic algorithms. In Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, pp. 224-229.

[66] Torbey E., Knight J. (1998, August). Performing scheduling and storage optimization simultaneously using genetic algorithms. In Proceedings of the Midwest Symposium on Circuits and Systems, pp. 284-287.

[67] Wang G., Gong W., DeRenzi B., Kastner R. (2006, July). Design space exploration using time and resource duality with the ant colony optimization. In Proceedings of the 43rd annual Design Automation Conference, pp. 451-454.

[68] Kopuri S., Mansouri N. (2004) Enhancing scheduling solutions through ant colony optimization. In: Proceedings of the International Symposium on Circuits and Systems (ISCAS), pp. 257–260.

[69] Mohanty S.P., Velagapudi R., Kougianos E. (2006) Physical-aware simulated annealing optimization of gate leakage in nanoscale datapath circuits. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE), pp. 1191–1196.

[70] Al-Mouhamed M., Al-Massarani A. (2000). Scheduling optimization through iterative refinement. Journal of systems architecture, 46(10), pp. 851-871.

[71] Springer D.L., Thomas D.E. (1994) Exploiting the special structure of conflict and compatibility graphs in high-level synthesis. IEEE Transactions on CAD of Integrated Circuits and Systems 13(7), pp.843–856.

[72] Pilato C., Loiacono D., Ferrandi F., Lanzi P. L., Sciuto D. (2008, June). Highlevel synthesis with multi-objective genetic algorithm: A comparative encoding analysis. In Proceedings of the IEEE World Congress on Computational Intelligence. pp. 3334-3341.

[73] Ferrandi F., Lanzi P. L., Loiacono D., Pilato C., Sciuto D. (2008, April). A multiobjective genetic algorithm for design space exploration in high-level synthesis. In

Proceedings of the IEEE Computer Society Annual Symposium on VLSI, ISVLSI'08. pp. 417-422.

[74] Yang H., Wang C., Du N. (2012). High Level Synthesis using Learning Automata Genetic Algorithm. Journal of Computers, 7(10), pp. 2534-2541.

[75] Micalewicz Z. (1996). Genetic Algorithms + Data Structures = Evolution programs, 3rd ed., Springer-Verlag.

[76] Patel B., Pradhan D. K., Koren I. (1991, Sep). High level synthesis of data driven ASICs. In Proceedings of Fourth Annual IEEE International ASIC Conf. and Exhibit, Vol. 13, No. 3, pp 1-4, 23-27.

[77] Dhodhi M. K., Hielscher F. H., Storer R. H., Bhasker J. (1995, Aug). Datapath synthesis using a problem-space genetic algorithm. IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, Vol. 14, No.8, pp. 934–944.

[78] Nestor J. A, Krishnamoorthy G. (1993). SALSA: A new approach to scheduling with timing constraints. IEEE Trans. on Comput. Aided Design of Integrated circuits and Systems, Vol. 12, No.8, pp. 1107–1122.

[79] Mandal C., Chakrabarti P. P., Ghose S. (2000, Dec). GABIND: a GA approach to allocation and binding for the high-level synthesis of data paths. In Proceedings of IEEE Trans. on VLSI Systems, Vol. 8, No. 6, pp. 747-750.

[80] Zhang Z., Fan Y., Jiang W., Han G., Yang C., Cong J. (2008). AutoPilot: a platform-based ESL synthesis system. In High-Level Synthesis Springer, pp. 99-112.

[81] Williams A.C., Brown A.D., Zwolinski M. (2008). Simultaneous optimisation of dynamic power, area and delay in behavioural synthesis. In Proceedings of IEEE computers and digital techniques, Vol. 147, No. 6, pp. 383–90.

[82] Lebreton L., Coussy G., Martin P. (2010). Hierarchical and Multiple-Clock Domain High-Level Synthesis for Low-Power Design on FPGA. In Proceedings of International Conference on Field Programmable Logic and Applications, pp.464-468.

[83] Liu, Hung-Yi, Carloni L. P. (2013). On learning-based methods for design-space exploration with high-level synthesis. In Proceedings of Design Automation Conference, pp. 1-7.

[84] Kennedy J., Eberhart R. C. (1995). Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948.

[85] Engelbrecht A.P. (2005). Fundamental of computational swarm intelligence. John Wiley and sons limited, England.

[86] http://www.cbl.ncsu.edu/benchmarks/, 2015.

[87] Oikonomakos P., Zwolinski M., (2006, Nov). An Integrated High-Level On-Line Test Synthesis Tool. In Proceedings of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.25, No.11, pp.2479 – 2491.

[88] Deb K. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, ISBN 047187339.

[89] Agrawal D., Baktir S., Karakoyunlu D., Rohatgi P., Sunar B. (2007). Trojan Detection using IC Fingerprinting. In Proceedings of IEEE Symposium on Security and Privacy, pp. 296-310.

[90] Tehranipoor M., Koushanfar F. (2010). A Survey of Hardware Trojan Taxonomy and Detection. IEEE Design & Test of Computers, pp. 10-25.

[91] Narasimhan S., Dongdong Du., Chakraborty R.S., Paul S., Wolff F., Papachristou C., Roy K., Bhunia S. (2010). Multiple-parameter side-channel analysis: A non-invasive hardware Trojan detection approach. In Proceedings of IEEE International Symposium on Hardware-Oriented Security and Trust, pp. 13-18.

[92] Karri R., Rajendran J., Rosenfeld K., Tehranipoor M. (2010). Trustworthy Hardware: dentifying and Classifying Hardware Trojans. Computer, pp. 39-46.

[93] Yier J., Makris Y. (2008). Hardware Trojan detection using path delay fingerprint. In Proceedings of IEEE International Workshop on Hardware-Oriented Security and Trust, pp. 51 - 57.

[94] Xiaoxiao W., Salmani H., Tehranipoor M., Plusquellic J. (2008). Hardware Trojan Detection and Isolation Using Current Integration and Localized Current Analysis. In Proceedings of IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, pp. 87-95.

[95] Mohanty S. P., Gomathisankaran M., Kougianos E. (2014). Variability-Aware Architecture Level Optimization Techniques for Robust Nanoscale Chip Design. Elsevier International Journal on Computers and Electrical Engineering (IJCEE), Vol. 40 No. 1, pp. 168-193.

[96] Cui X., Ma K., Shi L., Wu K. (2014). High-level synthesis for run-time hardware Trojan detection and recovery. In Proceedings of 51st IEEE Design Automation Conference (DAC), pp. 1 – 6.

[97] Banga M., Hsiao M.S. (2009). A Novel Sustained Vector Technique for the Detection of Hardware Trojans. In Proceedings of 22nd International Conference on VLSI Design, pp. 327-332.

[98] Texas Instruments: 'Benchmarks - C674x Low Power DSP - TI.Com', Aug 2014, http://www.ti.com/lsds/ti/dsp/c6000dsp/c674x/benchmarks.page.

[99] Nikara, J., Takola, J., Akopian, D., & Saarinen, J. (2001, May). Pipeline architecture for DCT/IDCT. In Proceedings of the IEEE International Symposium on Circuits and Systems, pp. 902-905.

[100] Elgamel, M., & Bayoumi, M. A. (2002). On low power high level synthesis using genetic algorithms. In Proceedings of the 9th International Conference on Electronics, Circuits and Systems, pp. 725-728.

[101] Kumar, A.K., Somasundareswari, D., Duraisamy, V. and Pradeepkumar, M. (2010). Low power multiplier design using complementary pass-transistor asynchronous adiabatic logic. In Proceedings of International Journal on Computer Science and Engineering, Vol. 2 No. 7, pp. 2291–2297.

[102] Crop, J., Fairbanks, S., Pawlowski, R., and Chiang, P. (2010). 150mV subthreshold Asynchronous multiplier for low-power sensor applications. In Proceedings of the International Symposium on VLSI Design Automation and Test (VLSI-DAT), pp. 254–257.

[103] Reynders, N., and Dehaene, W. (2011). A 190mV supply, 10MHz, 90nm CMOS, pipelined sub-threshold adder using variation-resilient circuit techniques. In Proceedings of the IEEE Asian Solid State Circuits Conference (A-SSCC), pp. 113–116.