AUTONOMOUS RECOVERY OF RECONFIGURABLE LOGIC DEVICES USING PRIORITY ESCALATION OF SLACK

by

NAVEED IMRAN
M.S. Electrical Engg. University of Central Florida, Orlando, 2010

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering
in the College of Electrical Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2013

Major Professor: Ronald F. DeMara

© 2013 Naveed Imran

ABSTRACT

Field Programmable Gate Array (FPGA) devices offer a suitable platform for survivable hardware architectures in mission-critical systems. In this dissertation, active dynamic redundancy-based fault-handling techniques are proposed which exploit the dynamic partial reconfiguration capability of SRAM-based FPGAs. Self-adaptation is realized by employing reconfiguration in detection, diagnosis, and recovery phases.

To extend these concepts to semiconductor aging and process variation in the deep submicron era, resilient adaptable processing systems are sought to maintain quality and throughput requirements despite the vulnerabilities of the underlying computational devices. A new approach to autonomous fault-handling which addresses these goals is developed using only a uniplex hardware arrangement. It operates by observing a health metric to achieve *Fault Demotion using Reconfigurable Slack (FaDReS)*. Here an autonomous fault isolation scheme is employed which neither requires test vectors nor suspends the computational throughput, but instead observes the value of a health metric based on runtime input. The deterministic flow of the fault isolation scheme guarantees success in a bounded number of reconfigurations of the FPGA fabric.

FaDReS is then extended to the *Priority Using Resource Escalation (PURE)* online redundancy scheme which considers fault-isolation latency and throughput trade-offs under a dynamic spare arrangement. While deep-submicron designs introduce new challenges, use of adaptive techniques are seen to provide several promising avenues for improving resilience. The scheme developed is demonstrated by hardware design of various signal processing circuits and their implementation on a Xilinx Virtex-4 FPGA device. These include a Discrete Cosine Transform (DCT) core, Motion Estimation (ME) engine, Finite Impulse Response (FIR) Filter, Support Vector Machine (SVM), and Advanced Encryption Standard (AES) blocks in addition to MCNC benchmark circuits. A

significant reduction in power consumption is achieved ranging from 83% for low motion-activity scenes to 12.5% for high motion activity video scenes in a novel ME engine configuration. For a typical benchmark video sequence, PURE is shown to maintain a PSNR baseline near 32dB. The diagnosability, reconfiguration latency, and resource overhead of each approach is analyzed. Compared to previous alternatives, PURE maintains a PSNR within a difference of 4.02dB to 6.67dB from the fault-free baseline by escalating healthy resources to higher-priority signal processing functions. The results indicate the benefits of priority-aware resiliency over conventional redundancy approaches in terms of fault-recovery, power consumption, and resource-area requirements. Together, these provide a broad range of strategies to achieve autonomous recovery of reconfigurable logic devices under a variety of constraints, operating conditions, and optimization criteria.

To all those whose innovations and efforts make the world a better place to live

ACKNOWLEDGMENTS

I am deeply grateful to my advisor Dr. Ronald F. DeMara whose continuous guidance, encouragement, and support made this dissertation possible. I really appreciate his exceptional advice, valuable time and very generous attitude throughput my doctorial studies.

I would like to thank Drs. Wasfy B. Mikhael, Mingjie Lin, Jiann-Shiun Yuan, and Christopher D. Geiger for their great suggestions and for their time to evaluate this dissertation as my advisory committee members.

I would also like to thank Dr. Jooheung Lee at Hongik University, Seoul, South Korea, for his valuable input during my collaborative work with him. I thank my friends for my unforgettable happy memories during my stay in Orlando.

Finally, I would like to thank my parents and sisters for their endless love, support and passionate encouragement throughout my life. I am greatly thankful to my wife and sons for their unremitting love, patience and support throughout this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACE Advanced Configuration Environment

AE Active Element

AEs Active Elements

AES Advanced Encryption Standard

APE Amorphous Processing Element

APEs Amorphous Processing Elements

BIST Built-in Self Test

CDM Comparison Diagnosis Model

CED Concurrent Error Detection

CLBs Configuration Logic Blocks

CRR Competitive Runtime Reconfiguration

CUT Circuit Under Test

DCT Discrete Cosine Transform

DMR Dual Modular Redundancy

DRFI Distance-Ranked Fault Identification

DR Dynamic Replica

DRs Dynamic Replicas

DSP Digital Signal Processing

ECC Error Correcting Code

EM Electromigration

FaDReS Fault Demotion using Reconfigurable Slack

FIAT Fault Injection and Analysis Toolkit

FPGAs Field Programmable Gate Arrays

FD Fault Detection

FE Functional Element

FI Fault Isolation

FR Fault Recovery

FT Fault Tolerance

FH Fault Handling

FHME Fault-Handling Motion Estimation

GA Genetic Algorithm

GAs Genetic Algorithms

hCED Heterogeneous CED

HDL Hardware Description Language

ISE Integrated Software Environment

ICAP Internal Configuration Access Port

ME Motion Estimation

MV Motion Vector

NUT Node Under Test

NMR N-Modular Redundancy

PE Processing Element

PEs Processing Elements

PSNR Peak Signal-to-Noise Ratio

PLB Programmable Logic Block

PR Partial Reconfiguration

PRR Partial Reconfiguration Region

PRRs Partial Reconfiguration Regions

PURE Priority Using Resource Escalation

QoS Quality of Service

QP Quantization Parameter

RN Reconfigurable Node

RS Reconfigurable Slack

SEU Single Event Upset

SET Single Event Transient

SA Stuck At

SCDR Self-Configuring Discrepancy Resolution

SNR Signal-to-Noise Ratio

SEUs Single-Event Upsets

SCDR Self-Configuring Discrepancy Resolution

STARs Self-Testing AReas

TDDB Time-Dependent Dielectric Breakdown

TMR Triple Modular Redundancy

LIST OF NOTATIONS

$\mathbf{G}(V, E)$	An undirected graph, where V is the set of all nodes, E is the set of edges
$\mathbf{C}$	Connectivity matrix
$C(t)$	Connectivity $\mathbf{C}$ at time instant t
$\mathbf{\Psi}$	Syndrome Matrix
$\hat{\Phi}$	Estimated Fitness State Vector
$\mathbf{P}$	Priority Vector
$t(G)$	Diagnosability of $\mathbf{G}$
$d(G)$	Average degree of a node in $\mathbf{G}$
V_a	Set of active nodes
V_s	Set of Reconfigurable Slack (RS) to diagnose the active nodes by comparison-based diagnosis
V_h	Set of healthy nodes
V_{NMR}	Set for N-Modular Redundancy checking
M	Number of PRRs
N_a	Number of nodes in the datapath (i.e., $\|V_a\|$)
N_s	Number of Reconfigurable Slacks (i.e., $\|V_s\|$)
N_d	Number of defectives
r	Testing arrangement instance (may involve multiple reconfigurations)
s	Slack update instance (a slack is reconfigured with some function)
t	Time instant
T_{recon}	Reconfiguration Time
T_{eval}	Evaluation window period

F	Functions assignments vector
F^*	Solution vector F after recovery
T_d	Latency of fault detection
T_{diag}	Latency of fault diagnosis
T_{rec}	Recovery time
N_r	Number of testing arrangement instances before the diagnosis completes
N_{sup}	Number of slack updates
N_{AE}, N_{DR}, N_{RS}	Number of AEs, DRs, and RSs, respectively
N	Total number of APEs defined in the reconfigurable device
N	Macroblock's size (N $\times$ N)
Φ	Predicted fitness status (0:healthy, 1:faulty, or x:suspect)
V_T	APEs in the pool under test
S	Search Range
n	Number of SAD's computed by an APE per clock cycle
$TCAE$	Current Testing Candidate APE in the active datapath (i.e., AE)
T_r	Time to reconfigure an APE
$\hat{S}$	Search range anticipated
μ	Average magnitude of motion vectors over a video frame
τ_{SAD}	Threshold to increased search range
Δ_{SAD}	Difference in SAD values

CHAPTER 1: INTRODUCTION

Survivability, reliability, and availability are indispensable characteristics of mission critical digital systems. To achieve these characteristics in electronics systems used in space, satellite, or other difficult to access environments where the manual intervention may not be feasible, autonomous repair capability becomes a desirable property. This chapter highlights the significance of the problem, and provides an overview of the techniques widely used in fault-tolerant designs on reconfigurable platforms. Afterwards, innovations of the proposed resilience approaches are identified and listed in the Contribution of Dissertation section.

Need for Reliability and Survivability

With the continued reducing feature size of semiconductor technology, device reliability and system survivability for mission-critical systems poses increasingly significant challenges [5][6][7] [8]. Error-resiliency and self-adaptability of future electronic systems are subjects of growing interest [5][9]. In some situations, even survivability in the form of graceful degradation is desired if a full recovery cannot be achieved. Transient, so called *soft*, errors as well as permanent, *hard*, errors in electronic devices caused by aging or radiation in space environment require autonomous mitigation as manual intervention may not be feasible [10]. The reliability problem of highly complex VLSI systems in sub-90 nanometer process, caused by soft and hard errors, is increasing. Therefore, the importance of addressing reliability issues is growing to sustain a high level of integration, performance, and transistor density on chip.

The self reconfiguration capability of Field Programmable Gate Arrays (FPGAs) is appealing for building fault-tolerant circuits. Various configurations of a design can be studied for throughput,

power, and reliability analysis. Fault recovery of FPGA-based designs can be realized by employing fault-free logic resources at runtime. Given some faulty resources in a particular region in an FPGA chip, the circuit can be repaired by assigning its functionality to a pristine area in the chip. Equivalently, if a circuit realized by a particular configuration-bitstream manifests faults, an alternate configuration-bitstream utilizing only the fault-free resources can be downloaded into a chip.

Another reason for research interest in fault-tolerance of FPGA based design is due to their popularity in mission critical systems [11]. The regular structure of an FPGA-fabric is amenable to reconfiguration-based recovery. A high regularity of FPGA logic resources allows movement of a function implemented over a defective region to a fault-free region [12] [13] [14] [15]. FPGAs are popular among space exploration community for its reconfigurability [10] [16]. On the other hand, FPGAs are also susceptible to transient as well as permanent faults, for example Single Event Upset (SEU) in the configuration memory, and Stuck At (SA) faults in the logic resource [17]. These errors can occur while operating in deep space environments when FPGAs are subjected to cosmic rays and high energy radiations. For fault detection capability, a duplex of the design can be instantiated on the chip and a discrepancy in the output can be monitored via a discrepancy detector [18].

At the *circuit level*, scaling the supply voltage V_{dd} and threshold voltage V_t have been effective methods to drastically reduce power consumption in digital circuits [19]. However, a mere scaling of operating voltage can lead to output degradation as it can result in increased delays of critical paths. For example, a low voltage operation of Discrete Cosine Transform (DCT) in a video encoder impacts Peak Signal-to-Noise Ratio (PSNR) quality metric due to erroneous output [20]. Without controlling the supply voltage adaptively, the power gains are diminished due to increased sensitivity of the circuits to manufacturing variations and longer critical path delays. Otherwise, choosing a $(V_{dd}$,$V_t)$ combination becomes necessary for error-free computation and power gains

become limited accordingly [21].

In addition to voltage scaling-induced errors and manufacturing *Process Variations (PV)*, other types of hardware faults include aging-induced degradations [22] and radiation-created permanent faults. Systems in which some manual intervention remains no longer a feasible option after deployment, the absence of provision of an autonomous fault-handling capability may lead to a catastrophic system failure [23]. Thus, the provision of capability to adapt the hardware is an essential characteristic for self-organizing hardware systems and reconfigurable hardware paradigm [24][25][26] is favorable in such a scenario. A unified scheme of mitigating hardware errors irrespective of their underlying causes is desirable to achieve output at sufficient quality levels. Nevertheless, hardware faults necessitate mitigation to sustain computational correctness.

While low power Digital Signal Processing (DSP) designs focus on reducing redundancy in computations, conventional error mitigation techniques rely on introducing some form of redundancy in computations. Therefore, for low power designs intended for modern or future nano-scale hardware platforms, the design poses a dilemma. Achieving power efficiency as a design objective seeks those algorithms which can be efficiently mapped to fewer computational units. However, such a reduction makes every computational unit more critical and hence more susceptible to errors. This becomes a significant concern when the intended algorithm is mapped to unreliable hardware fabrics. At the other end of the spectrum, to achieve robustness as a design objective, architectures are sought which exploit or even introduce redundancy in the design. As this type of approach brings in new alternatives to distribute the reliance to multiple hardware units, overall fault-tolerance is enhanced. However, it can realize power hungry designs as the redundancy to mask errors becomes a significant overhead. To combat these challenges in a unified manner, we propose a design framework which introduces the concept of dynamic modular redundancy utilizing computational priorities. We utilize a reconfigurable approach to avoid redundancy during normal operation of the Circuit Under Test (CUT) and dynamically introduce it at run-time to

mitigate fault scenarios.

Characteristics of Fault-Tolerant Systems

Fault tolerant systems are characterized by the reliability and dependability they provide in mission critical systems. The fault detection capability, or *detectability*, is an important attribute of the fault tolerant systems by which faults or a system failure can be detected [27]. An indication is required in situation of faults when the output of the system deviates from its desired operation. While the detectability can be implemented by observing the behavior of a system through certain variables, another way is to replicate the system to realize a duplex configuration. A disagreement in output of the two instances indicates the faulty nature of operation as an error in at least one of the two instances.

A survivable system is defined as one that, when enabled by likely regeneration strategy, can operate without substantial depreciation throughout its expected lifetime even when subjected to multiple internal or external fault-invoking conditions. Specifically within the domain of DSP, a device is said to be survivable if it is capable of handling imminent failures throughout its lifetime by taking the actions necessary to maintaining desired signal processing performance above some minimum threshold. The threat of diminished component reliability becomes more unpredictable due to escalating thermal profiles, process-level variability, and harsh DSP environments such as deep-space and high-altitude flight. Furthermore, increasing density and complexity renders preventing or eliminating all possible design faults to be increasingly infeasible. All these factors pose renewed challenges to designing signal processing circuits resistant to unpredictable damage or malfunction.

Two conventional approaches to handle permanent faults in FPGAs are through Triple Modular

Redundancy (TMR) via tools such as Xilinx XTMR, or progressive resolution via distinct *detection*, *diagnosis*, *isolation*, and *recovery* processes. Typically, recovery relies upon reconfiguring the impaired functional block in a different fault-free portion of the fabric. Partial Reconfiguration (PR) capability is beneficial in achieving runtime adaptability with reduced time and space overhead. Compared to resource-oriented Built-in Self Test (BIST) based schemes [28], *functional testing* techniques offer model-free fault-diagnosis [18]. Moreover, autonomous survivability is desired to continue operation without halting service. Ideally, an online fault handling scheme would not interrupt the continuous throughput of the system while only temporarily degrading the spatial/temporal resolution, or PSNR quality. Nonetheless, minimal impact on PSNR is desirable and rapid fault recovery becomes an important design objective.

Dynamic redundancy techniques have been widely used to increase reliability of critical systems in which reconfiguration is employed at runtime to utilize spare units in response to failures [29] [30] [31]. While some techniques rely on pre-allocation of dedicated spare units, a dynamic spare pool sharing approach can be favorable in terms of extending fault-capacity [8]. Redundancy enables fault-tolerance, however, how wisely redundancy is employed at runtime determines the sustainability of the system exposed to cumulative failures. Fig. 1.1 illustrates these characteristics of an ideal autonomous recovery technique and the techniques developed in this dissertation in the context of exiting approaches towards achieving these goals. A favorable fault-tolerance technique minimally impacts the throughput datapath. In addition, the area-overhead of a fault-handling controller, δ should be very low as compared to the baseline area where δ is fixed, independent of the datapath complexity, and a fraction of the size of the datapath.

Biological systems have inherent self-repairing capabilities which have inspired signal processing research to mimic these natural adaptive processes in silicon-based systems. Thus, research interest has been increasing toward electronic systems which can sustain considerable damage, yet still remain operational or at least partially operational. Consequently, self-repair and self-healing

mechanisms have been proposed for digital hardware by various researchers [32][33][34]. These mechanisms rely on identifying or employing some form of redundancy, reconfiguration, or both. To realize these properties in a signal processing system, it is useful to identify how a layered model emphasizing the impact of signal processing tasks on output correctness and the runtime reconfiguration of FPGA resources based on Evolvable Hardware can be leveraged.

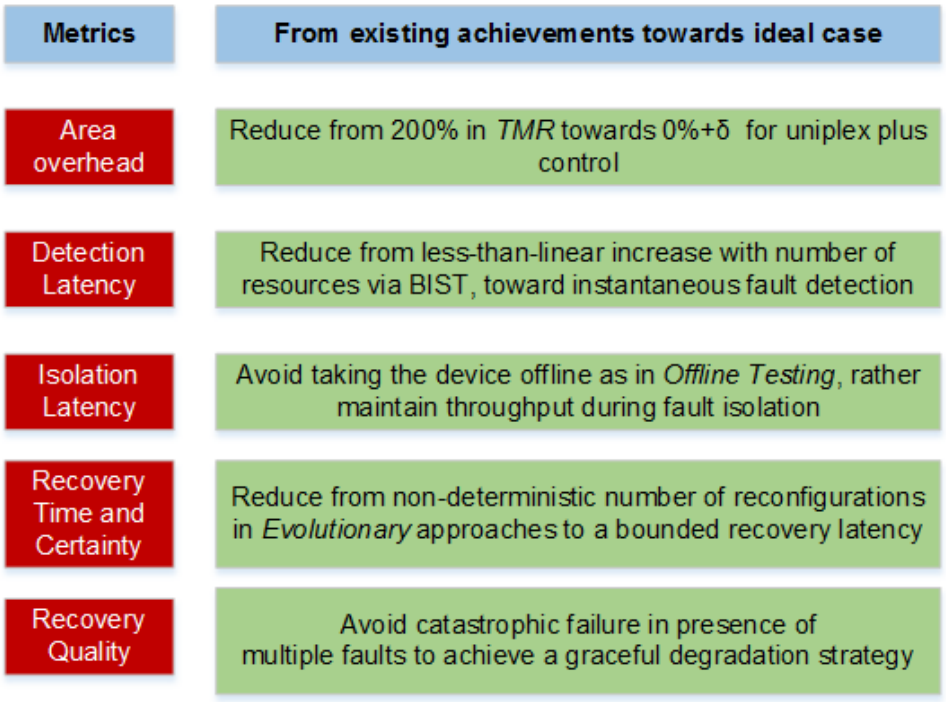


Figure 1.1: Characteristics of an ideal autonomous recovery technique

Evolvable Hardware has been proposed in literature as a reconfiguration-based approach to achieve fault tolerance in electronic designs. These methods extend static fault tolerance techniques at design-time which attempt to make designs which are more robust to faults [35][36]. In particular, runtime evolvable hardware techniques reconfigure hardware resources at runtime to refurbish the circuit [18]. Previous works establish the successful use of Evolutionary Algorithms for adaptive self-recovery of hardware systems based on reconfigurable logic platforms, especially in FPGA-

based systems [18][37][38]. A survey of techniques ranging from passive to dynamic in classification are presented in [39] to tackle hard faults in SRAM-based FPGAs for small circuit case studies. For example, modular redundancy is exploited in [40] for achieving fault recovery of a 4-bit x 4-bit multiplier. Moreover, novel techniques are sought which are scalable to large modular signal processing systems.

Researchers have devised runtime evolutionary techniques to realize fault-resilient electronics through iterative selection [41][42][43]. The fault-detection technique in [43] employs redundant cells in the reconfigurable fabric to check the operating resources by detection discrepancies among replicated outputs. Although, previous attempts have been made to combine architecture and algorithm level knowledge [44][45], there remains a need to develop frameworks utilizing cross-layer information in a way that leverages the soft-resilience present in signal processing applications.

Soft Resilience of Signal Processing Systems

In the domain of DSP, a system is said to be *resilient* if it is capable of handling failures throughout its lifetime to maintain the desired signal processing performance within some tolerance. The threat of diminished component reliability becomes more critical to maintaining these tolerances due to process-level variability, as well as escalating thermal profiles which can accelerate aging effects [46] [47]. Additionally, harsh DSP environments such as deep-space and high-altitude flight can further exacerbate lifetime reliability concerns. Meanwhile, increasing device density and system complexity can make the use of design margins and timing guard-banding techniques more difficult [48]. All these factors pose renewed challenges to designing signal processing systems resistant to process variation and aging-induced malfunction.

Dynamic redundancy techniques based on reconfiguration have been widely used to increase re-

liability [30][31]. While traditional fault-handling techniques rely on pre-allocation of dedicated spare units, more recent approaches based on dynamic spare pool sharing can be favorable in terms of reducing area overhead[46] [49]. Resiliency is achieved when a regeneration strategy allows a system to operate without substantial depreciation throughout its lifetime, even when subjected to multiple internal or external fault-invoking conditions. Redundancy enables fault-tolerance, however, how wisely redundancy is employed at runtime determines the sustainability of the system after exposure to cumulative failures. Adaptive reconfiguration can reduce the size of a sustainable spare pool, and it also enables the novel resiliency strategies developed herein.

FPGAs offer two important features towards resilient signal processing architectures. First, FPGAs have been used to achieve significant acceleration of DSP applications over conventional computing platforms [50][51]. Second, FPGAs provide hardware support for adaptive reconfiguration. From a reliability perspective, the regular structure of an FPGA-fabric is amenable to reconfiguration-based recovery. A high regularity of FPGA logic resources allows movement of a computational function implemented over a defective region to a fault-free region [15][52]. This characteristic has already made FPGAs popular for application in the space exploration community[10][16][53]. On the other hand, SRAM-based FPGAs are also susceptible to soft (transient) errors as well as hard (permanent) faults [54] that can be addressed using the techniques developed in this work.

Aggressive scaling of semiconductor technology to cope with today's intensive processing demands leads to seeking new autonomous reliability approaches for logic devices. In particular, the reliability concern of VLSI signal processing systems implemented in a sub-32 nanometer process, caused by soft and hard errors, is increasing. Therefore, the importance of providing resiliency is increasing in order to achieve a high level of integration, throughput performance and quality, and the classical trends of transistor density per chip.

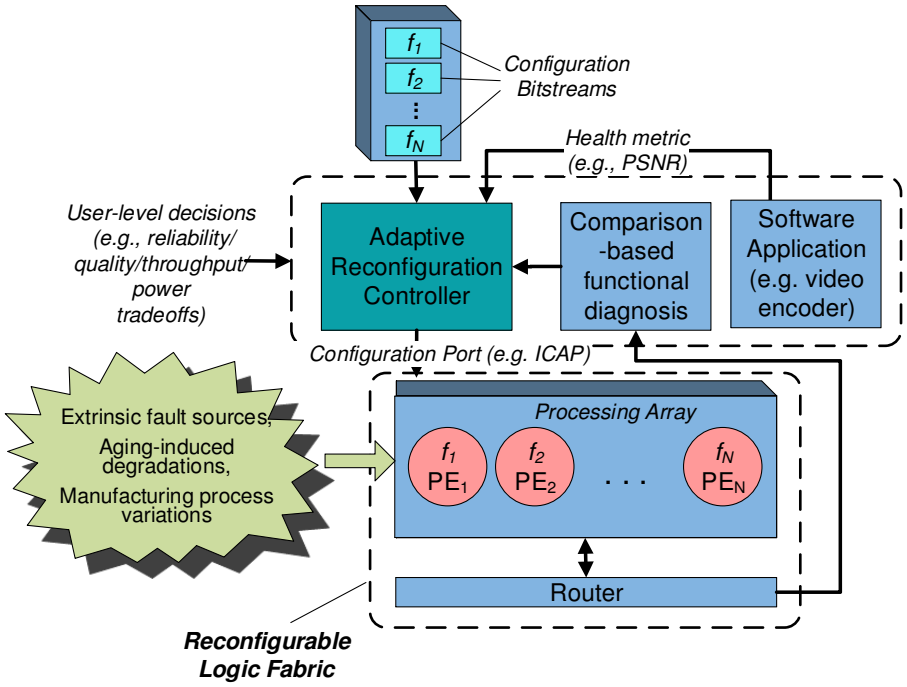


Figure 1.2: System block diagram illustrating the scope of reconfiguration techniques

In this dissertation, a strategy is presented for autonomously mitigating permanent faults in order to improve system availability and mission lifetime. The scheme is advantageous in terms of continuous operation, power consumption, and area-overhead while improving reliability. We consider a Functional Element (FE) which has been decomposed into various Processing Elements (PEs) for throughput enhancement. Such a distributed implementation is also beneficial in terms of fault-tolerance. Some of the PEs can be used at runtime to perform diagnosis while others can be configured as operational resources to compensate for failures and maintain performance requirements. Without loss of generality, we term each of these PEs as Reconfigurable Slack (RS). Each RS region denotes a contiguous 2-dimensional reconfigurable region of FPGA logic resources used to diagnose the active PEs. An RS has size and shape which is identical to an active PE in throughput datapath, yet is not currently configured to contribute to the throughput. Multiple

RS's are not required for the techniques herein, but are shown to decrease the fault diagnosis latency. A system level block diagram to illustrate the framework used in this work for evaluating the techniques developed herein is given in Fig. 1.2. The reconfiguration management of logic resources for fault-handling purposes is performed by reconfiguration controller. The configuration bitstreams are stored in an external memory module. Results from comparison-based diagnosis are monitored at software level. These reconfiguration bitstreams represent alternate configurations to recover from faults. They are loaded from the storage memory as needed dynamically, under control of autonomous algorithms executing on the reconfiguration controller. Thus, from a high-level viewpoint, the objective of this dissertation is to develop algorithms for the reconfiguration controller that best address the fault-handling characteristics identified in Figure 1.1

Quality-Oriented Architectural Adaptations

To deal with susceptibility to aging and process variation in the deep submicron era, signal processing systems are sought to maintain quality and throughput requirements despite the vulnerabilities of the underlying computational devices. The Priority Using Resource Escalation (PURE) online resiliency approach is developed herein to maintain throughput quality based on the output PSNR or other health metric. PURE is evaluated using an H.263 video encoder and shown to maintain signal processing throughput despite hardware faults. Its performance is compared to two alternative reconfiguration algorithms which prioritize the optimization of the number of reconfiguration occurrences and the fault detection latency, respectively. For a typical benchmark video sequence, PURE is shown to maintain a PSNR baseline near 32dB. Compared to the alternatives, PURE maintains a PSNR within a difference of 4.02dB to 6.67dB from the fault-free baseline by escalating healthy resources to higher-priority signal processing functions. The diagnosability, reconfiguration latency, and resource overhead of each approach is analyzed. The results indicate

the benefits of priority-aware resiliency over conventional redundancy in terms of fault-recovery, power consumption, and resource-area requirements.

Voltage scaling has been an effective approach to reduce the power consumption in DSP systems due to the quadratic dependence of power on operating voltage. However, variations in the fabrication process can manifest soft errors in devices built with deep submicron technology [47][55]. The reliability issues of modern signal processing architectures due to voltage scaling are being addressed in recent research [56][57]. Many of these works take various approaches to leverage the role of priority in the signal processing computation to improve resiliency, along with its area and energy costs. For example, the general concept of asymmetric reliability is developed in [46] to prioritize the protection of higher order bits in error resilient architectures supporting probabilistic applications. Algorithmic level properties are utilized to realize area efficient replicas of motion estimation blocks to achieve reliable operation under energy efficiency constraint in [58]. Likewise, to minimize the power overhead of error resilience while maintaining signal quality, the scheme proposed in [47] exposes only less crucial blocks to process variation and channel noise.

In this dissertation, the developed techniques exploit health metric based feedback to perform reconfiguration in order to meet resiliency, availability, energy efficiency, and survivability objectives. Various applications are considered as case studies. These health metrics include PSNR of video encoder, bitrate of the compressed bitstream, and measure of confidence from DCT, Motion Estimation (ME), and SVM modules, respectively. In other case studies where a readily available health metric is not feasible, the discrepancy information is used to assess the erroneous behavior of the hardware fabric. Fault-diagnosis algorithms developed herein engage the priority of processing blocks in order to sustain partial throughput during the recovery period. The recovery strategy also considers functional priorities when mitigating hard faults. Furthermore, the tradeoffs of quality and energy efficiency are explored by a multi-objective formulation of the reconfiguration problem.

The primary focus of this work is to develop novel and effective techniques for autonomous fault-handling in digital systems. To this end, reconfiguration based diagnosis and recovery techniques are proposed to effectively utilize redundancy needed for fault-tolerance.

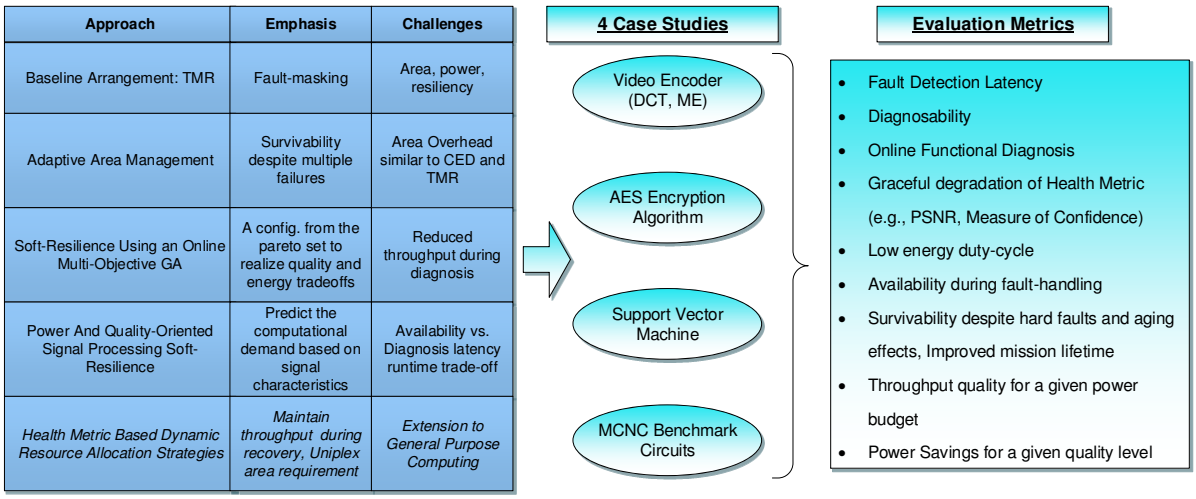


Figure 1.3: A roadmap diagram illustrating the techniques evaluated herein

Fig. 1.3 illustrates the emphasis of various techniques presented in this dissertation and evaluation metrics in the context of some signal processing case-studies. Here, the five novel approaches for autonomous fault-handling developed in this dissertation are listed. The last approach listed is the capstone work in the dissertation using an observable health metric. In the next column, each approach has a technical focus or objective related to the desirable characteristics for autonomous fault handling mentioned earlier ranging from simple fault-masking to sophisticated maintenance of throughput during recovery while incurring negligible area overhead. To achieve these objective, each approach faces technical challenges ranging from reduced throughput during diagnosis to recovery latency. Next, four case studies were selected to evaluate each approach. These case

studies were chosen due to their popularity in signal processing, communication systems, machine learning, and computer architecture. Each case study was then evaluated using one or more metrics as listed on the right side of Figure 1.3. While progress was made on improving many of these, a focus on fault-handling latency and recovery quality proved especially effective as will be shown in Chapter 6. The main contributions of the dissertation are listed in the following, while the chapters following a chapter on the related work describe these techniques in detail.

- Developed area management techniques for the fault handling problem in reconfigurable logic devices with δ area-overhead, less than 5% throughput degradation, and ability to sustain multiple failures in the hardware resources. These are *A Self-Configuring TMR Scheme Utilizing Discrepancy Resolution (SCDR)*, *Heterogeneous Concurrent Error Detection (hCED)*, *Amorphous Slack (AS) Fault-Handling Methodology*, and *Distance-Ranked Fault Identification (DRFI)* presented in Chapter 3.

- Formulated the objective of maintaining the quality-of-service and power consumption into a generalizable runtime mapping problem based on the underlying resource performance and operating workload. A multi-objective GA approach is developed for this mapping optimization problem in which a population of solutions is guided by a novel adaptive guidance function as presented in Chapter 4.

- Demonstrated use of PSNR as a health metric to achieve autonomous monitoring of operation for graceful degradation, low-power operation, and survivability in presence of multiple failures. These techniques are *Fault-Handling Motion Estimation (FHME) Engine*, *Fault Demotion Using Reconfigurable Slack (FaDReS)*, and *Priority Using Resource Escalation (PURE) Escalation* presented in Chapters 5 and 6, respectively

CHAPTER 2: RELATED WORK

Fault Handling (FH) systems typically employ a sequence of resolution phases including Fault Detection (FD), *Fault-Diagnosis*, and Fault Recovery (FR). A system can be considered to be fault-tolerant if it continues operation in the presence of failures, perhaps in a degraded mode with partially restored functionality [59]. Reliability and availability are desirable qualities of a system, which are measured in terms of service continuity and operational availability in presence of adverse events, respectively [60]. In this work, reliability is attained by employing the reconfigurable modules in the fault-handling flow, whereas availability is maintained by minimum interruption of the main throughput datapath.

The redundancy based FD methods are popular among fault-tolerant systems community, with costs of area and power overhead. In the *Comparison Diagnosis Model* [61][62], a pair of units is evaluated subjected to the same inputs and a discrepancy indicates failure. For example, a Concurrent Error Detection (CED) arrangement utilizes either two concurrent replicas of a design [18], or a diverse duplex design to reduce common mode faults [30]. Its advantage is a very low fault detection latency. A TMR system [63][64] utilizes three instances of a datapath module. The outputs of these three instances become inputs to a majority voter, which in turn, provides the main output of the system. In this way, besides fault detection capability, the system is able to mask its faults in the output if distinguishable faults occur within one of three modules. However, this incurs an increased area and power requirement to accommodate three replicated datapaths. It will be shown that these overheads can be significantly reduced by either considering the instantaneous PSNR measure obtained within video encoder as a precipitating indication of faults or periodic checking of the logic resources.

The *Fault Diagnosis* phase consists of distinguishing properly-functioning components from some

larger set of suspect components. Traditionally, in many fault tolerant digital circuits, the components are diagnosed by evaluating their behavior under a set of test inputs. This *test vector strategy* can isolate faults while requiring only a small area overhead, yet incurs the cost of evaluating an extensive number of test vectors to diagnose the functional blocks as they increase exponentially according to the number of inputs. The proposed active dynamic redundancy approach combines the benefits of redundancy with a negligible computational overhead. *Static redundancy* techniques reserve dedicated spare resources for fault-handling. In contrast, in the presented approach, the redundant modules are continually utilized in the datapath during the normal mission operation.

While reconfiguration and redundancy are fundamental components of a FR process, both the choice of reconfiguration scheduling policy and the granularity of recovery affect the *availability* during *recovery phase* and *quality of recovery* after fault-handling. Here, it is possible to exploit the algorithm's properties so that the reconfiguration strategy is constructed taking into account varying priority-levels associated with required functions.

Various methods of achieving fault-tolerance in FPGAs include device level, passive redundancy, resource-oriented testing, functional testing, GA, configuration level, multiple configurations, and scrubbing techniques. The relevant work is described in detail in the following.

Static Redundancy

A passive *redundancy* based scheme, TMR is popular in FPGA-based reliable designs for protection against permanent as well as transient faults. As a matter of fact, a vendor's tool *XTMR* is available to triplicate the user logic [63]. The errors can be masked in the output to some extent. A Dual Modular Redundancy (DMR) arrangement is also commonly used for fault detection in situations in which area and power overhead of a TMR is unaffordable.

A research of different forms of the CED setup was presented by Mitra et. al [30]. The CED schemes rely on some form of redundancy for fault detection purpose. Sometimes, the functional modules are implemented in the form of a diverse duplex system which involves two alternative implementations of the same design. This helps error detection in case of common mode failures. Besides duplex systems, CED schemes also realize parity based systems [30]. For example, an even/odd parity can be used to ensure the correctness of an output sequence of a digital system.

Temporal redundancy techniques have been explored in fault-tolerant microprocessor systems. A typical error detection scheme involves running a duplicate thread for the comparison purpose on a chip multiprocessor (CMP). Hyman et. al [5] proposed an extension to the scheme by exploiting various redundancies in instructions in multi-core processors framework. Thus, if an instruction is affected by transient errors in the execution path, the duplicate execution would provide a fault detection capability. However, if the faults are of permanent nature, the re-execution of an instruction would have the same result and the error detection becomes impossible. This is because a given input data will exercise the logic resource in the same way no matter how many times an instruction is executed. In the proposed Heterogeneous CED (hCED) approach, we apply the desired function to the input data at one instant, and the redundant computation is performed on the difference data in the second instant. Thus, the logic resource is exercised with different input data in each case. In this way, the approach is able to detect errors in case of permanent faults also, in addition to transient errors.

In the context of previous work, the followings are the key-points of the hCED approach: FD in spatial *h*CED mode with resource saving, FD in temporal *h*CED mode with uniplex chip area requirement at the cost of reduced throughput, and the coverage of transients as well as permanent faults in FD using temporal *h*CED.

The Algorithmic Noise Tolerance (ANT) technique [58] offers area efficiency which consolidates

an application oriented approach to achieve fault-resilience. In the ANT scheme, a reduced precision replica of CUT is employed which is less energy hungry; then CED is performed to check the CUT for output discrepancy. Alternatively, in [45], fault-handling is demonstrated in uniplex mode of operation using the runtime reconfiguration property of FPGAs. Fault-detection was performed by observing the behavior of the PSNR metric while diagnosis was performed by re-mapping the APEs contained in the DCT core itself with significantly less area overhead as compared to ANT. Varying priorities of DCT coefficients were exploited to recover from faults scenarios.

Resource Testing by BIST

Resource-based Testing techniques rely on testing the logic resources using some test vectors. The output response of the logic resources is analyzed to identify their health. Online BIST and Roving Self-Testing AReas (STARs) are based on the principle that part of a chip is subjected to test inputs, and the test area is moved around while keeping the system online. The techniques proposed by Emmert et al. [6], Dutt et al. [65], and Gericota et al. [66], are some examples of resource oriented techniques. The heterogeneous nature of FPGA resources (e.g., LUTs, FFs, BRAMS, DSP Blocks) makes it intractable to come up with a generic testing methodology. Moreover, the scalability of resource testing techniques with huge growth of on-chip resources is also a concern. Therefore, functional testing is an appealing alternative to resource testing.

Resource testing techniques for fault isolation of FPGA resources have been proposed in literature in the form of either offline testing or online testing [67]. In an offline BIST method, all the active resources are released from their active functionality and a testing sequence is conducted to verify the correctness of these resources. However, this method is less practical for real-time systems having specific timing deadlines, or mission critical systems in which device outage may be problematic to the mission. On the other hand, *Online Testing* schemes may employ the dy-

namic reconfiguration capability of FPGA and tests can be performed during runtime. Online BIST techniques [6] check a small area of the chip in concurrence while keeping the remaining non-tested regions in operation. Resource testing typically involves pseudo-exhaustive input-space testing of the physical resources to identify faults, while functional testing methods check the fitness of the datapath functions [65]. In Gericota et. al's approach [66], the active logic resources are concurrently replicated to support a runtime testing procedure. Their active replication technique concurrently creates replicas of Configuration Logic Blocks (CLBs) to improve the reliability. Dutt et. al [65] extended the BIST method to offline as well as online testing modes. In their approach, the output of a Programmable Logic Block (PLB) is compared to that of an identically configured PLB. A discrepancy in the output flags the PLB as faulty. The exhaustive evaluation of all the resources through test vectors may be a long process. Our scheme can be conceptualized as resource testing through actual inputs of the circuit.

BIST techniques are characterized by the fact that fault detection latency may be long depending upon the chip area. Moreover, transient errors are not detectable in these schemes. In our approach, by introducing some redundancy for error checking purpose, the transient errors are also detectable with a negligible fault detection latency. Gao et. al [68] proposed a resource testing scheme using time multiplexing of different components through the reconfiguration capability of FPGA. As the reconfiguration time is a considerable entity in current FPGA technology, the Self-Configuring Discrepancy Resolution (SCDR) technique multiplex the inputs to a fixed hardware fabric instead of reconfiguring the resources with alternating functions.

System-Level Diagnosis

There is a body of research dealing with the problem of identifying faulty components by employing system diagnosis theory. A pioneer work in diagnosis theory is by Preparata et al. [62] in

which the problem of identifying faulty nodes in a digital system is formulated as a connection assignment procedure. Various components of a digital system are represented by nodes in a graph described by a *connection matrix*. A given edge in the graph connects two nodes, one being the node *under test* and the other being *testing* node. The diagnosibility of digital systems containing faulty modules has been studied by various researchers [69][70]. In the proposed scheme, reconfigurable hardware's bitstreams can be conceptualized as nodes of a graph representing a digital system.

In general, the process of identifying faulty nodes in a system $\mathbf{G}$ is called *Fault Diagnosis*. The maximum number of faulty nodes which a scheme guarantees to identify is known as *diagnosability* of the system. *System-Level Diagnosis* is a widely used technique for fault resilience in multiprocessor systems. In Comparison Diagnosis Model (CDM) [61] [71], [72], [73], a pair of units is evaluated subjected to the same inputs and a discrepancy indicates some failure. The impact of a topology on diagnosability of a network is thoroughly discussed in [74] [75]. In the proposed adaptive reconfiguration schemes, we will consider a fully connected topology so that the diagnosis can be performed between any pair of nodes. Then, after identifying a faulty node, it can be replaced by any of the available healthy nodes. It will be shown that fault-handling can be performed online by time varying topology of the PEs.

Evolvable Hardware Techniques

Evolvable hardware techniques focus on adapting hardware to achieve fault tolerance. These methods rely on finding a configuration which meets fitness criteria under a given fault scenario. A Competitive Runtime Reconfiguration (CRR) [18] scheme uses evolution to repair the faulty resources of a CED arrangement. We can divide these evolutionary schemes into two types 1) Design-time fault tolerance 2) Runtime fault tolerance. The focus in design time fault tolerance

is to build circuits which are robust to faults in different components, yet the disadvantage is that fault recovery is limited to only anticipated faults. On the other hand, the focus in runtime fault tolerance schemes is to recover during runtime operation. Using the dynamic partial reconfiguration capability and the presented recovery sequence, PURE approach is able to achieve a high degree of runtime fault tolerance.

Keymeulen et al. [36] proposed an *evolutionary* approach to circumvent the faults in reconfigurable digital circuits. They proposed genetic algorithms to evolve the population of fault tolerant circuits by applying genetic operators like mutation and crossover over the circuit representation. Another technique [76] is based upon bitstream manipulation by evolutionary algorithms to recover from faults.

Heng and DeMara [77] developed a *Multilayer Runtime Reconfiguration Architecture* for autonomous fault handling in FPGA. They split the task into logic, translation and reconfiguration layers and manipulate the hardware configurations using on-chip resources for autonomous repair. Lach et al. [78] split the design into tiles and calculate the reliability of each tile instantiated into the design implemented in FPGA. On identifying the tile that uses faulty resources, they assign the allocated spare resources to that element. *Multiple configurations* are generated by [18] for fault handling purpose whereas the configurations are repaired using evolutionary algorithms.

Sharma et al. [79][80] used *group testing* techniques to isolate fault locations in FPGA. Once the resources are isolated, the recovery is made by utilizing alternative logic resources. The method presented in this dissertation does not require explicit fault isolation phase, while the configurations are only generated at design time thereby not necessitating the vendor's synthesis and implementation tool at runtime. Fault handling is accomplished by promoting the hardware configurations which utilize fault-free resources [12]. The proposed Distance-Ranked Fault Identification (DRFI) approach is a system-level fault-diagnosis technique by which healthy configurations are identified

in a configuration pool, while the instantiation of two healthy configurations in a duplex manner completes the fault-recovery process.

Reconfiguration Techniques

Hardware autonomy is desirable in space systems because manual intervention may be an infeasible option. Steiner et. al [81] proposed an autonomous system in which hardware computational resources are managed at runtime. Their demonstrated system can dynamically parse and synthesize digital circuit netlists, place-and-route on FPGA at a very fine granularity. It relies on a customized implementation tool. Our method operates at a coarse granularity of block-level in the circuit corresponding to pipelined stages of hardware core. The autonomous operation can be realized using dynamic reconfiguration capability, an on-chip microprocessor and the internal access port for reconfiguration. An autonomous operation of hardware is also desirable for other applications involving certain objectives such as power optimization.

FPGAs are widely used in signal processing, image processing and video applications [82] due to their parallel nature. In addition, the reconfiguration capability [83] provides flexibility in exploring different hardware architectures. The dynamic reconfiguration of FPGA resources can be performed in a fault handling scheme to avoid the faulty resources.

Dynamic partial reconfiguration capability of FPGAs has been explored for useful tasks by various researchers [84], [85], [86]. Fault recovery methods of FPGA-based designs usually exploit the reconfigurable nature of the device. After completion of the fault isolation phase, the faulty resources are avoided by reconfiguring the chip so that the design is relocated to a fault-free area. On the other hand, evolutionary techniques such as Genetic Algorithms (GAs) have been employed to generate circuits at design-time which are robust to faults [36]. In the current work, the circuit

is evolved at runtime to reach the desired level of functionality. The GAs being soft computing stochastic search process, the bounds on search time are not achievable; however, related work using GAs demonstrated very acceptable recovery time for circuits with various number of LUTs.

Comparison of Techniques

Table 2.1: Comparison of fault-tolerance techniques for SRAM-based FPGAs

Approach	**Area requirement**	**Basis for Recovery**	**Detection Latency**	**Number of Reconfigs**	**Additional Components Required**	**Granularity**	**Guarantee of improvement**
TMR	3 fold	Requires 2 datapaths are operational	Negligible	Not Applicable	2 of 3 Majority Voter	Function or Resource level	100% for single fault, 0% thereafter
Evolutionary Hardware	Not Applicable	Redundancy and Competitive Selection	Not Applicable	Only when fault is present; Non-deterministic	GA Engine	Logic Blocks	No
CRR	Duplex	Recovery complexity	Negligible	Only when fault is present; Varies	CRR controller	Function level	No
Online Recovery (Roving STARs, Online BIST)	Roving Area	Available Spares	Significant: linear in number of PLBs	Continuous reconfiguration	Test vector generator, Output response analyzer	Logic Blocks	Yes
PURE (the approach proposed herein)	*Uniplex*	*Priority of functionality*	*Negligible*	*Only when fault is present; Linear in number of functions*	*Reconfig. Controller*	*Computational Functions*	*Yes*

Reliability of FPGA based designs [87] can be achieved in various ways. Table. 2.1 provides a comparison of previous approaches towards fault-handling in FPGA based systems. *Passive recovery* techniques, such as TMR, are popular but incur significant area and power overheads. The TMR technique involves a triplication of the design where the three copies of system components are active simultaneously. The fault recovery capability is limited to the faults within one instance only. This limitation of TMR can be overcome using *self-repair* [68] [88] approaches to increase sustainability, such as refurbishing the failed instance using *jiggling* [76]. Other *active recovery*

techniques incorporate control schemes which realize intelligent actions to cope with a failure. Evolutionary techniques [36] avoid the area overhead of pre-designed spares and can repair the circuit at the granularity of individual logic blocks, yet lack a guarantee that a recovery would be obtained within a certain number of generations. They may require hundreds of Genetic Algorithm (GA) iterations before finding an optimal solution, thus undesirably extending the recovery time. On the other hand, PURE operation is bounded in terms of maximum number of evaluations required. Many evolvable hardware techniques have been presented in literature that rely on modifications in current FPGA device structure. In addition, a fitness evaluation function must be defined a-priori to select the best individuals in a population, which may in turn necessitate knowledge of the input-output truth table. PURE avoids both of these complications. Altogether, they allow PURE to evaluate to the actual inputs, instead of exhaustive or pseudo-exhaustive test vectors, on any commercial off-the-shelf FPGA with PR capability.

One approach to reducing overheads associated with TMR is to employ the Comparison Diagnosis Model with a pair of units in an adaptable CED arrangement subjected to the same inputs. For example, the CRR [18] scheme uses an initial population of functionally identical (same input output behavior), yet physically distinct (alternative design or place-and-route realization) FPGA configurations which are produced at design time. At run-time, these individuals compete for selection to a CED arrangement based on a fitness function favoring fault-free behavior. Hence, any physical resource exhibiting an operationally-significant fault decreases the fitness of those configurations which use it. Through runtime competition, the presence of the fault becomes occluded from the visibility of subsequent operations.

Other runtime testing methods, such as online BIST techniques [6] offer the advantages of a roving test, which checks subset of the chip's resources while keeping the remaining non-tested resources in operation. Resource testing typically involves pseudo-exhaustive input-space testing of the FPGA resources to identify faults, while functional testing methods check the fitness of

the datapath functions [65]. In [89], a pair of blocks configured with identical operating modes are subjected to resource-oriented test patterns. This STARs approach keeps a relative small area of the device off-line and being tested, while the rest of the device is online and continues its operation. STARs compares the output of each Programmable Logic Block (PLB) to that of an identically configured PLB. This utilizes the property that a discrepancy between the output flags the PLB as suspect as outlined by Dutt et. al's *Roving Tester (ROTE)* technique [65] and used in Gericota et. al's active replication technique [66] which concurrently creates replicas of CLBs. In STARs approach, each block-under-test is successively evaluated in multiple reconfiguration modes, and when a block is completely tested then the testing area is advanced to the next block in the device. To facilitate reconfigurability to relocate the system logic, there is a provision to temporarily stop the system operation by controlling the system clock. The recovery in STARs is achieved by remapping lost functionality to logic and interconnect resources which were diagnosed as healthy.

In contrast, Fault Demotion using Reconfigurable Slack (FaDReS) and PURE perform functional testing of the resources at higher granularity by comparing outputs of PEs which execute functions that comprise a signal processing algorithm. This allows resources to be tested implicitly within the context of their use, without requiring an explicit model of each PE's function. In the proposed dynamic resource allocation schemes, the testing components remain part of the functional datapath until otherwise demanded by the fault-handling procedure. Upon fault detection, these resources are designated for fault diagnosis purposes. Later, upon the completion of fault diagnosis and recovery, the reconfigurable slacks may then be recommissioned to perform priority functions in the throughput path.

CHAPTER 3: ADAPTIVE AREA MANAGEMENT FOR LOCAL PERMANENT DAMAGE

In the conventional TMR arrangement, if one module becomes faulty, two other healthy modules are able to provide throughput as long as no discrepancy is observed in their output. Yet, an additional fault in one of these two healthy modules can result in a mission's failure, or equivalently a discrepancy in the output of an operational pair (or CED pair) flags one of these two instances as *faulty* [90]. The observable faults are those which manifest their behavior during an evaluation period. The effect of evaluation period on fault detection in CED pair was examined in [91].

Under these conditions, repair schemes are presented to mitigate permanent faults in which the discrepancy between the output of a CED pair initiates the genetic recovery process. Upon fault detection in a CED pair, or two instances of a TMR arrangement, the system is recovered through reconfiguration by repairing faulty instances.

A Self-Configuring TMR Scheme utilizing Discrepancy Resolution

Evolutionary techniques have been proposed in literature for achieving fault tolerance in FPGAs. The logic resources or interconnects are somehow represented by individuals of a population to be evolved. The genetic operators are performed to either repair the circuit or to create a robust circuit at design-time. These techniques require the knowledge of input-output values to evaluate the fitness criteria. However, for large circuits having many possible input-output values, the evaluation of every possible combination of input-output becomes an infeasible option. We employ a fitness criterion in which an absolute knowledge of the correctness of output values is not required. The followings are the advantages of the proposed repair mechanism:

- A self-healing TMR system is proposed with improved fault capacity. Initially, there are three instances of a module in operation, then in event of faults in two modules, the repair is performed to switch the system into duplex mode. Thus, in effect, the fault capacity of a TMR is improved.
- The fitness evaluation is performed using the actual inputs of the system, avoiding any test vectors. Thus, an optimal solution is sought in the relevant input subspace.
- The fault handling process does not require an explicit fault isolation phase. The faulty modules are consequently avoided by the evolutionary recovery process.
- Fault handling of sequential circuits is demonstrated. A considerable throughput is maintained during the fault recovery process.

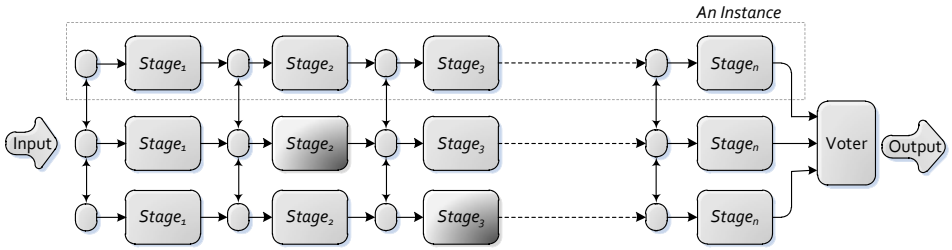
The SCDR Approach

The TMR realization of a sequential circuit is illustrated in Fig. 3.1(a). A CUT is replicated 3 times resulting into $Instance_1$, $Instance_2$ and $Instance_3$ of the CUT. The majority voter computes the system's main output by enabling the majority output value. Faults are thus masked in the output if a single instance is affected.

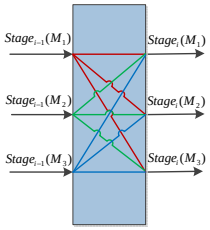
To improve the reliability, in the presence of multiple faults which may occur during long mission durations, adaptation of the datapaths can be applied. For this purpose, switches are inserted between various pipeline stages of the circuit to steer the datapath. Combinational circuits can also be partitioned into blocks. The configuration of these switches determine which module is selected as an active element of each instance. The SCDR concept is to avoid the faulty blocks and utilize the healthy blocks in the processing datapath.

In Fig. 3.1(a), the shaded blocks represent faulty stages. A *Module Switch (MS)* element, which

can be realized as a multiplexer or a custom-routing circuit, is connected between every pipeline stage of a digital sequential circuit (or a block more generally) to steer the output of a block to one of three blocks in the following stage. A *Router Box* (RB) is a realization of the three MS elements, one for each instance of a TMR pathway. It supports six possible input-output pairnings (i.e. 3!=6) as shown in Fig. 3.1(b). It can be implemented in hardware by either routing wires through partial reconfiguration or using multiplexers. The former has the advantage that no delay is introduced by the multiplexers in the datapath. However, it comes at the cost of reconfiguration latency during the fault recovery process. On the other hand, fault recovery time can be improved by introducing dedicated multiplexers at design time, but this introduces a drawback that the MS elements remain in the active throughput path at all times.



(a) The TMR realization of a sequential circuit



(b) A Router Box

Figure 3.1: Circuit realization to employ the SCDR recovery mechanism

Encoding Representation of the TMR Pathways

In the presented SCDR scheme, an *evolutionary fault recovery* process is used to explore the search space of RB settings to obtain a more suitable TMR realization. An individual of the population represents one possible configuration of the TMR system. The number of variables in a GA individual is equal to the number of pipeline stages or number of blocks n in an instance of the TMR arrangement. Therefore, the length of a chromosome is n. Each variable can assume one of 6 possible values while each value corresponds to a unique configuration of the RB. The GA representation of the routing between each stage is exemplified in Fig. 3.2 where a chromosome's values correspond to different configurations of the RB. The representation of the GA problem is such that it exploits the dynamic partial reconfiguration capability of FPGA. The faulty blocks of a duplex can be swapped with healthy blocks of the third instance of TMR arrangement during runtime. In the following, the scheme is presented in the context of a standard GA [1] to facilitate conventional analysis.

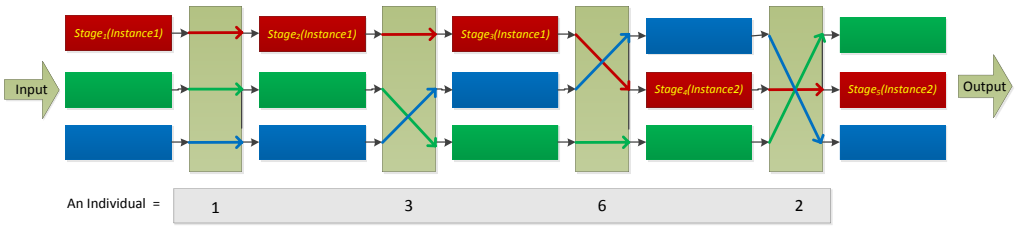


Figure 3.2: Mapping between an individual and configuration

Fitness Function

Given many possible configurations of the system, it is difficult to ascertain which is superior in terms of desired functionality of the system. Many previous approaches store the input-output test

vectors or truth-tables of the circuit. The SCDR avoids this approach because of two drawbacks: 1) the storage requirement would be prohibitive for large circuits 2). The application of test vectors may require the system taking offline. Therefore, we employ a previously developed *Competitive Runtime Reconfiguration* (CRR) approach [90] in which individuals are selected based upon their behavior consistency with the consensus of the other population members.

A *Discrepancy Value* (DV) is defined as the Euclidean distance between the outputs of two instances of a TMR in a given evaluation window. The objective is to minimize the DV between $Instance_1$ and $Instance_2$.

$$DV = \frac{||Y_1 - Y_2||}{\sqrt{E}} \tag{3.1}$$

where

Y_1 = Output of the *Instance*$_1$

Y_2 = Output of the *Instance*$_2$

E = *Evaluation Window*

which is used to access the fitness of competing TMR pathways.

Fitness Evaluation

For the purpose of evaluation, each individual of the population is instantiated on the chip and its fitness is updated after a temporal sliding window of input samples. In the SCDR approach, the individuals are evaluated subjected to the actual inputs of the system rather than synthetic test vectors. The input is applied to the TMR arrangement and the output of the individual instances is observed during the evaluation window period, E. Upon the completion of the evaluation window,

a new configuration of the TMR is introduced into operation. Once all the individuals have been evaluated, a generation of the GA is complete. Thus, fitness scores of the individuals which is based upon their agreement/disagreement history, becomes available at the end of a generation. The SCDR proceeds next to select which configurations should be retained for subsequent operations.

Fitness Selection

Fitness Selection is the strategy of choosing individuals from the population to be parents which produce offspring that realize new arrangements. The fitness selection strategy in this section is based on the rank selection [92]. The fitness score of the individuals is converted into the respective rank, then the probability of an individual being selected as a parent is inversely proportional to the square root of its rank. Recall that we want to minimize the fitness score which is the DV. Thus, the use of a rank selection strategy makes it more probable that the configurations having small DV would survive to produce new configurations in the next generation.

Genetic Operators

A *crossover* operator interchanges the segments of chromosomes between two parents to produce offspring. A crossover rate specifies how many individuals go through the crossover operation in one population generation. The *mutation* operator relocates the chromosome parts within an individual to diversify the population.

In summary, the evolutionary process for the recovery mechanism is shown in Fig. 3.3. The exit criteria of the algorithm is the realizing of at least one configuration able supporting duplex mode with a full consensus, i.e., discrepancy-free outputs. The implication is that these configurations are those precisely which avoid known faulty resources as we discuss in the next section.

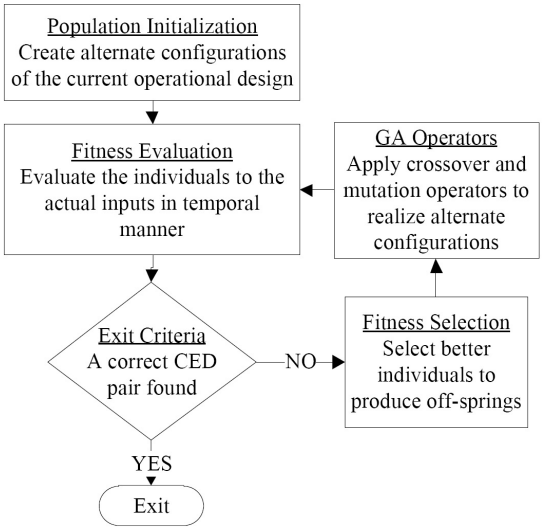


Figure 3.3: The evolutionary recovery process in the context of a standard GA [1]

Experiment Design

A 25 stage FIR filter is implemented in Verilog HDL using the Xilinx ISE 9.2i development tool. An ML410 development board is used which contains Virtex-4 FPGA chip, Compact Flash interface, DDRAM, and UART. For experimental purpose, the GA was simulated on a desktop PC rather than using the PowerPC on-chip processor at this time. Although the PowerPC is a hardware overhead of the GA-based SCDR recovery process, yet it is not on the critical throughput path. Thus, the use of PowerPC for just recovery purpose is likely to be amenable to most applications. The logic resources utilized by one instance of the TMR arrangement are 2810 number of LUTs and 500 FFs excluding those needed by the bus macros which are needed for partial reconfiguration flow in Xilinx ISE 9.2i.

Then, the circuit is replicated three times to realize a TMR system. The effect of random *Stuck-At (SA)* faults for injection into different stages of the FIR filter was realized by modifying the LUT contents in the post place-and-route simulation model. This simulates faults in different pipeline

stages of a sequential circuit. An instance of the experiment is shown in Fig. 3.4, where faults are injected in the three stages of the $Instance_1$ and 5 stages of the $Instance_3$. The experiment objective is to achieve fault recovery of two instances constructing a CED pair.

	The 25 stages of an FIR filter																								
$Instance_1$					x	x	x																		
$Instance_2$																									
$Instance_3$															x	x	x	x	x						

x indicates a faulty stage

Figure 3.4: A faulty TMR configuration

Simulation Results

Intrinsic Hardware Evaluation using SCDR

The fitness history of the recovery process is shown in Fig. 3.5. These plots depict the average behavior and best behavior of the arrangements in each generation. Even with use of this realistic case study consisting of 25 stage FIR filter utilizing 2810 LUTs and with 8 faulty stages simultaneously present, the GA is able to converge to a minimum score within 70 generations. Here, the population size used was 50, the crossover rate was 0.8, and the size of the evaluation window was 100 input samples.

The zero value fitness score of an individual corresponds to a TMR configuration in which two instances completely agree in terms of their output. A repaired instance after the genetic recovery process is shown in Fig. 3.6 where arrows depict the selected data pathway. It can be observed that this instance avoids any faulty resource, although any knowledge of faulty behavior of the resources was made unavailable. This demonstrates that consensus-based fitness evaluated over a sufficient window can provide a good approximation of the actual fitness of the system thus

identifying faulty modules.

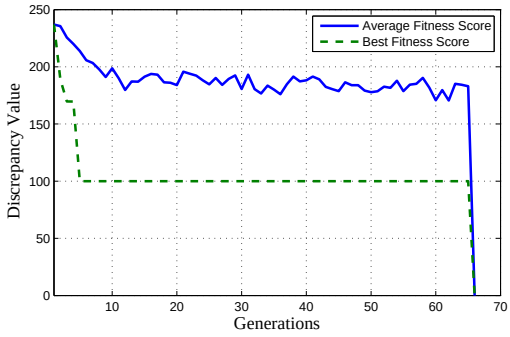


Figure 3.5: The consensus fitness history of the population

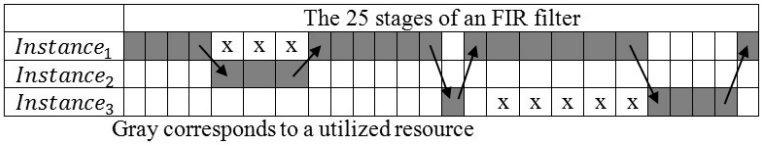


Figure 3.6: A repaired instance in the new configuration

The amplitude spectrum of the input signal contains two periodic sine waves of frequency 50 Hz and 100 Hz. The cut-off frequency of the low-pass FIR filter is set to 75Hz. The spectrum of the output signal from the filter is shown in Fig. 3.7. After fault injection, the output spectrum is also given in Fig. 3.7 which is different than what is desired for the filter functionality. In addition, Fig. 3.7 depicts the output of the preferred recovered TMR arrangement after completing the SCDR fault recovery process. To quantify the recovery quality, the *Signal-to-Noise Ratio* (SNR) is defined by the ratio of signal power, P_x to the noise power, P_e. In this case, the SNR is computed by:

$$SNR = 10 * log_{10} \frac{P_x}{P_e} \tag{3.2}$$

The difference signal is defined by:

$$e(t) = x(t) - y(t)$$

where

$x(t)$ = Input signal to the filter

$y(t)$ = Output signal from the filter

It is evident from Fig. 3.7 that the SNR measure of the signal after fault recovery process of the system is identical to the original fault-free system. Thus, the desired functionality of the FIR filter is retained.

Faults-Aware Simulation Paradigm

To further evaluate the SCDR scheme, a simulation was performed in which the absolute fitness was assessed in the presence of knowledge about the faulty modules locations. The objective cost to be minimized is the utilization of a minimum number of faulty blocks on data pathway , and is defined by:

$$cost = \sum_{i=1}^{3} \sum_{j=1}^{n} Stage_j(Instance_i).FS \tag{3.3}$$

where

n = Number of pipeline stages in the circuit

$Stage_j(.).FS$ fitness state $\in \{0, 1\}$

In the above set, '0' corresponds to *Healthy* condition, and '1' corresponds to *Faulty* condition of a module.

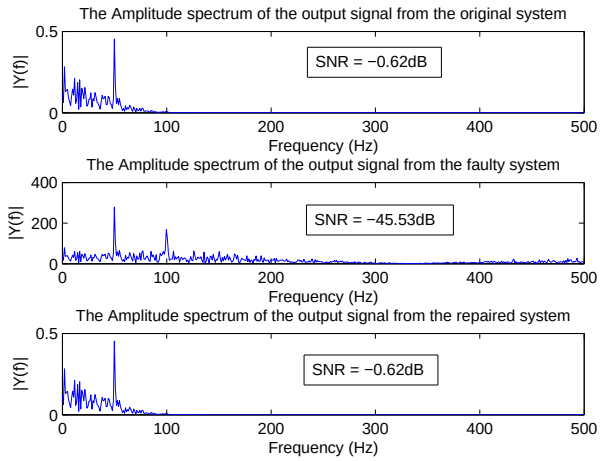


Figure 3.7: The amplitude spectrum of the output signal

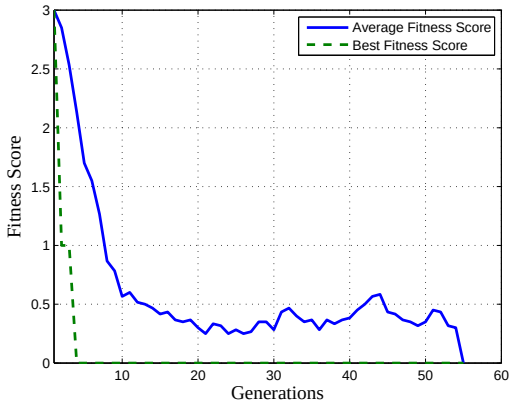


Figure 3.8: The absolute fitness history of the population

Fig. 3.8 shows that although average behavior of the population for various generations is same in both the SCDR and the faults-aware simulation cases (i.e., Fig. 3.5 and Fig. 3.8), the best individual is found in smaller number of generations in the later case. It is due to the fact that the knowledge about faulty nature of the modules is made available to the fitness function and the GA readily evolves towards the objective of minimum number of faulty resources utilization. This

case, however, utilizes ideal information which may not be available during fielded missions.

Performance Bound Comparison to Exhaustive Search

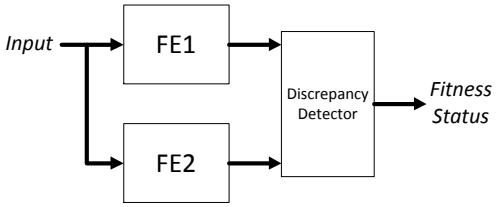
Finally, we compare with a case when all the possible configurations are evaluated exhaustively instead of employing a GA. As there are 3 instances, each with n variables, the total number of configurations combinations to be evaluated would be $N_E = n^3$.

For n=25 stage circuit, the upper bound on the required number of evaluations N_E becomes 15,625 which is much larger than the number of evaluations taken by the GA to reach the solution. As shown in Fig. 3.5, the population size of 50 is able to bring the correct configuration in 66 generations, thereby necessitating only 3,300 evaluations indicating effective recovery for the 25 stage FIR design. Future work will be to assess and improve recovery time which may be significant for mission-critical and safety-critical circuits. In addition, the focus will be to conduct generic analysis in presence of variable number of pipeline stages and granularity of fault handling.

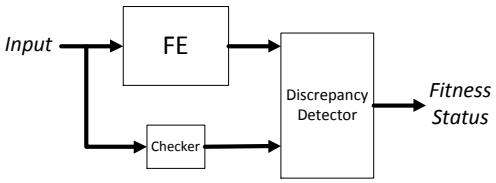
Heterogeneous Concurrent Error Detection (*h*CED) Based on Output Anticipation

The redundancy based schemes have overhead of resource and power. A TMR and higher-MR systems [13] have better fault coverage and fault masking capabilities than a CED based system, at the expense of increased power consumption and the number of components requirement. In this work, our focus is CED setup and thus, fault detection in a fault tolerant system. A conventional CED setup is shown in Fig. 3.19, in which two exact replicas of a given module concurrently operate to compute for the same input. In the figure, a FE is the main required component of the system which provides the system's throughput. FE1 and FE2 are two replicas and their output is monitored by a discrepancy detector. In this way, one of the FEs can provide the system's output,

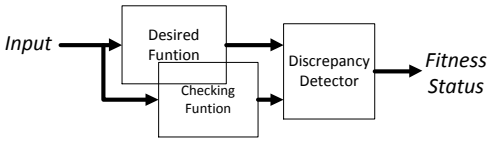
whereas the other FE can be thought of as a *Checker* for the main FE.



(a) Conventional Concurrent Error Detection arrangement



(b) Heterogenous Spatial Concurrent Error Detection arrangement



(c) Heterogenous Temporal Concurrent Error Detection arrangement

Figure 3.9: Various CED configurations

Because the sole purpose of a CED setup is the error detection, we can reduce the checker size compromising some of its capabilities. For example, by marginalizing the checker's throughput, a considerable number of resources can be saved. Thus, the checker does not need to be an replica of the functional module. We present two forms of heterogeneous CED exploiting some properties of the functional module. In the spatial CED type (Fig. 3.9(b)), the checker size is reduced by marginalizing its throughput at the cost of some increased fault detection latency. In the temporal CED form (Fig. 3.9(c)), the same hardware fabric is alternately switched between the actual function and the checker function. Thus, error detection is possible with uniplex resource requirement at the cost of some reduction in throughput.

Alternate CED Arrangements

As a case study, we consider a DCT hardware core to evaluate two forms of the heterogeneous CED. DCT function is widely used in image/video compression applications, and hardware implementation is highly desired in many applications due to parallel nature of the image processing related tasks and their throughput requirements. One example is video encoder application in which an image frame is first divided into macroblocks and the transform operation is performed on these macroblocks.

The DCT was introduced by Ahmed et. al [93] in 1974, when an efficient computation of Fourier Transform (FT) was desired. They defined the DCT of a data sequence as:

$$G_x(0) = \frac{\sqrt{2}}{M} \sum_{m=0}^{M-1} X(m)$$

$$G_x(k) = \frac{2}{M} \sum_{m=0}^{M-1} X(m) cos \frac{(2m+1)k\pi}{2M} \tag{3.4}$$

where $G_x(k)$ is the k-th DCT coefficient [93]. As a digital image is represented in computers via 2-D matrix notation, it is often desired to define 2-D transforms for image processing tasks. The DCT is used to represent an image by sum of varying sinusoidal amplitudes and frequencies. After the transform operation, the information content of a natural image is usually concentrated in only first few coefficients of the DCT. Due to this property, the DCT can be used 1) for image compression because the information can be contained in fewer coefficients. 2) for pattern recognition as the feature size to be used in classification can be reduced considerably. Moreover, scale and rotation variance are easier to handle in frequency domain than that in pixel domain. Gonzalez and Woods

[94] represent the 2-D DCT operation as:

$$r(x, y, u, v) = \alpha(u)\alpha(v)cos\frac{(2x+1)u\pi}{2n}cos\frac{(2y+1)v\pi}{2n} \tag{3.5}$$

where

$$\alpha(u) = \begin{pmatrix} \sqrt{\frac{1}{n}} & foru = 0 \\ \sqrt{\frac{2}{n}} & foru = 1, 2, ..., n-1 \end{pmatrix}$$

The details are given in their book [94].

The DCT matrix is commonly calculated for 2-D input data samples using the equation, and is given in Fig. 3.10.

$$\phi = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}$$

Figure 3.10: The DCT matrix

Our current focus is fault detection of the DCT block. In the following, we derive a sufficient condition to mark a DCT hardware block as *faulty*.

The 1-dimensional DCT operation on a row of pixels in a macroblock is defined by:

$$Y = \Phi.X \tag{3.6}$$

Where

X = A row of input pixels macroblock.i.e., $\{x_1, x_2, ..., x_8\}$

Φ = Matrix of DCT kernels

Y = A row of output DCT coefficients.i.e., $\{y_1, y_2, ..., y_8\}$

As the DCT operation is a linear transformation from input pixels space to output coefficients space, we can make the following derivations:

$$(Y - \grave{Y}) = \Phi.(X - \grave{X}) \tag{3.7}$$

where $\grave{X}$ and $\grave{Y}$ are the vectors for the previous time instant. Let's define:

$$\Delta X = X - \grave{X}\ ,\ \Delta Y = Y - \grave{Y}$$

then

$$\Delta Y = \Phi.\Delta X$$

The current DCT coefficients can be estimated from the previous coefficients using the difference values. i.e.,

$$\hat{Y} = \grave{Y} + \Delta Y \tag{3.8}$$

Thus, a necessary condition for a fault free DCT block can be written as:

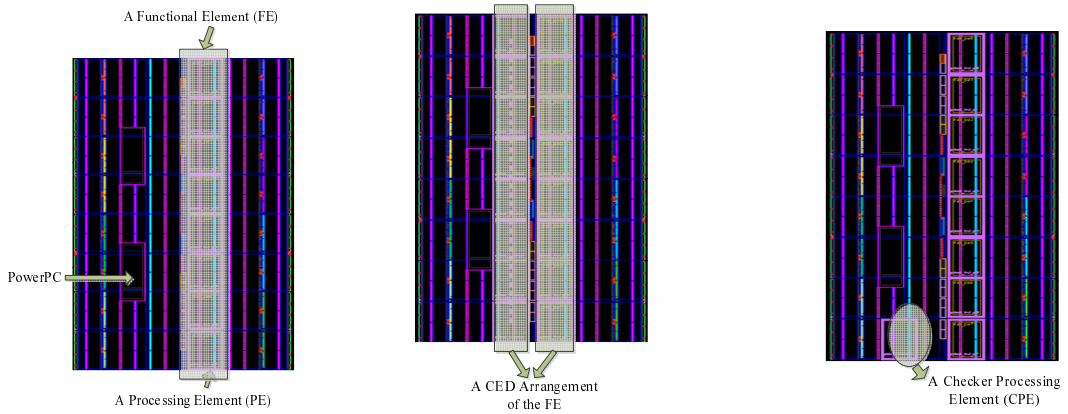
$$Y = \hat{Y} \tag{3.9}$$

If the equality is not met, the DCT block may be flagged as faulty. It may be noted that this is a necessary condition, not sufficient. Thus, even if the equality in the above expression is met, it does not imply fault-free nature of the module. Manifested fault in the logic or routing resource of the FE renders it faulty. Manifested faults are those faults which affect the output of any FE. We compute the output Y of DCT module through the FE, and the predicted output $\hat{Y}$ through the *Checker*. Thus, the checker serves as a predictor of the output that is desired at the module output for its correct operation.

The Baseline Setup

An 8-point 1-D DCT is implemented in Verilog HDL using Xilinx ISE 9.2i development environment. The place-and-route report shows that the design can run up to a clock frequency of 108 MHz. An FE consists of 8 *Processing Elements* (PEs), where each PE computes one DCT coefficient of a row of input pixels. Through the pipelining scheme, we are able to output one DCT coefficient every clock cycle. The DCT kernels are stored inside these PEs using the Look-up Tables (LUTs) of the FPGA chip. For example, PE_1 contains the DC-kernel, PE_2 has the AC_0-kernel and so on. Internally, the PEs use Multiply-Accumulate (MAC) units to perform the dot product of input data with the kernels. To provide scalability in the implementation [95], different Partial Reconfiguration Regions (PRRs) are defined to accommodate the PEs. Xilinx PlanAhead is used for the partial reconfiguration design, and ExploreAhead is used to generate partial bitstreams. The design is tested on a Xilinx development board ML410 which has a Virtex-4 FPGA. We report the

utilized resources in the next section when the resource requirement in two forms of the CED is compared.



(a) An FE realizing the DCT function

(b) CED arrangement

(c) Spatial *h*CED arrangement

Figure 3.11: Floorplan of various FE configurations realizing a DCT module

The DCT core is interfaced with on-chip PowerPC microprocessor through Xilinx provided GPIO core. The PowerPC writes the fixed-point format pixels data to the transposition memory. Also, it generates the control signals for the hardware DCT controller to sequence reads and writes operations from the frame buffer (transposition memory). The DCT controller manages to perform the 1-D operation on the pixels data row-wise in the first stage of the DCT, whereas the 2-D operation is accomplished by repeating the DCT operation on the input data column-wise. The completion of DCT operation on a macro-block is acknowledged to the PowerPC through GPIO core after which the processor can read the DCT coefficients from the transposition memory. A dual port RAM is instantiated to serve as the frame buffer. The physical layout of the design is shown in Fig 3.11(a). An FE realizing the DCT function and containing 8 PEs, is highlighted. A conventional arrangement of the CED is realized by the placement illustrated in Fig 3.11(b).

Spatial Heterogeneous CED

The architecture of the spatial *h*CED is given in Fig. 3.12. In the context of Fig. 3.9(b), PE1 through PE8 form an FE computing the DCT of input data. On the other hand, the PE containing MAC9 serves as a *Checker Processing Element* (CPE). The CPE performs MAC operation on the difference input instead of the input pixel values. The CPE utilizes the previous output from FE to predict the current output of the FE. If a discrepancy is observed between the current output from FE and the predicted output from CPE, it gives an indication of fault(s) in one of the two elements.

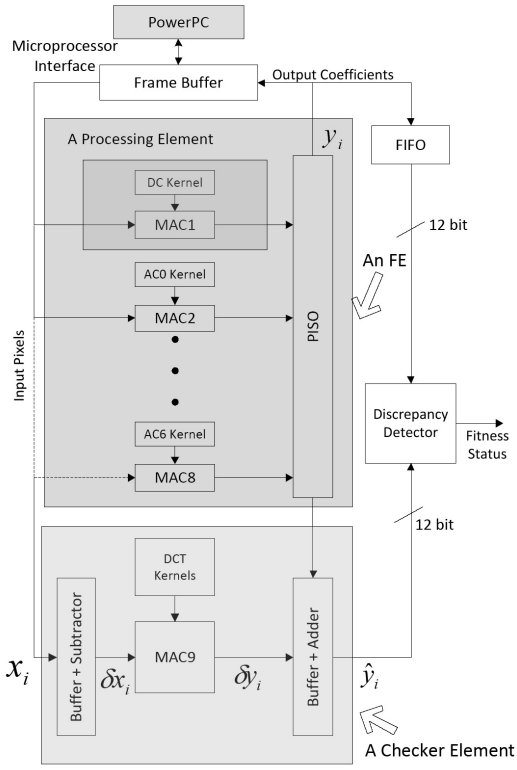


Figure 3.12: Spatial heterogeneous CED arrangement realizing a DCT module

All the DCT kernels for the checker unit are stored inside the CPE. The placement layout of the processing units is given in Fig. 3.11(c). The CPE computes the DCT coefficients sequentially unlike the FE which computes all the 8 coefficients in parallel owing to the contained multiple

PEs. Thus, one DCT coefficient is available at the checker's output after every 8 clock cycles which can be compared with that residing in the output buffer of the FE. A discrepancy between the predicted coefficient and the actual coefficient indicates an error.

Table 3.1 lists the resource requirement; the chip resources utilized by a PE are given in the first column, the second column provides a resource count of the 8 PEs in the DCT module. A conventional CED would require two times of this resource count. By using a CPE instead of a replica, a considerable amount of chip resources is saved as evident in the third column, which in turn saves power. The 8 PEs utilize 728 Configuration Logic Block (CLB) slices of the chip, while one CPE implementation requires only 167 slices, saving overall 77% of the FPGA resource. The resource requirement of a CPE is not $\frac{1}{8}$th of that of an FE as all the DCT kernels are stored in the CPE. On the other hand, each PE of the FE contains one kernel only.

It may be noted here that this is not very area-efficient implementation of the DCT core. Our objective is to evaluate the fault detection methodology. We avoid the Xilinx built-in hard multipliers known as DSP48 blocks, and employed LUTs while synthesizing the design as a generic implementation is desired to be analyzed for error detection purpose. Moreover, we can inject SA faults at LUT inputs and analyze their behavior in post place-and-route simulation model.

Table 3.1: Resource utilization for spatial hCED arrangement

Component	**PE**	**FE**	**CPE**
Number of slices	91	728	167
Number of slice FFs	46	368	75
Number of 4 input LUTs	161	1288	308

An analysis of the design by using Xilinx XPower tool reveals that one PE requires an estimated power of 12.39 mW. On the other hand, a CPE's estimated dynamic power consumption is 13.33 mW using the clock frequency of 100 MHz. As a CED arrangement will require a total of 8 PEs to serve as a checker, therefore the power requirement for the checker is approximately $\frac{1}{8}$th which

uses a CPE.

This saving in resource and power, however, comes at a cost. Because of the sequential nature of the CPE, number of comparisons for the discrepancy check that can be made in a given time-period, are reduced by a factor of 8. The frequency of comparisons defines the latency of detection in case of fault occurrence. By increasing the processing units for the checker module, the fault latency can be improved. It may be a worth to be noted here that even a detection latency of 64 clock cycles may be negligible as the design is running at 108 MHz, especially when the resource saving is considerably large. The latency of fault detection also depends upon the location of faults and input data. Although, Single-Event Upsets (SEUs) could be missed when using reduced sampling rate for error detection, they can incur only transient noise in the output image.

Temporal Heterogeneous CED

In this form of the *h*CED, temporal redundancy is used for fault detection in DCT module. Instead of applying a repeated function on the same input, we alternately apply the same function on two different types of inputs. In the first instant, the DCT computation is performed on the input pixels set. In the next time slot, the same computation is performed on the difference values. In this way, the hardware fabric is time multiplexed between two types of inputs. In this two shot operation mode, even permanent faults may be manifested as the diverse inputs exercise the underlying logic resources in a different way.

Fig. 3.13 shows the temporal *h*CED setup. At the first time instant (t=1), FE computes the DCT coefficients ($\grave{Y}$) for the given input row of pixel values ($\grave{X}$). At the second instant (t=2), the DCT coefficients (Y) are computed for the next row of pixels. At the instance (t=3), same FE is used for the DCT computation of the ΔX, which is a difference between two consecutive rows of input pixels. The output ΔY is used as an alternative computation (i.e., prediction $\hat{Y}$), which can be

used for the discrepancy check.

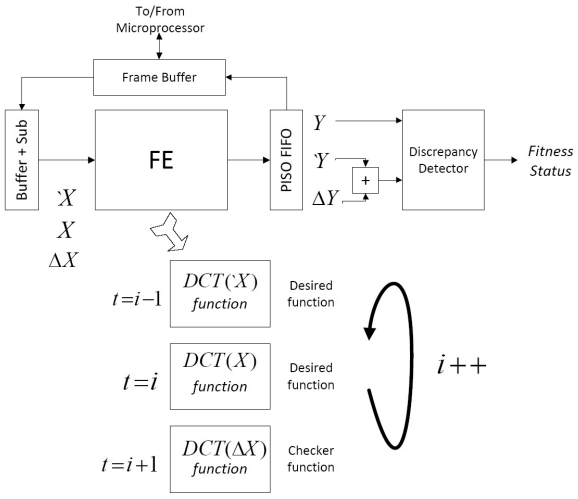


Figure 3.13: Temporal heterogeneous CED arrangement of the DCT module

If we dedicate one time instant out of 3 instants for the prediction computation, the *effective* throughput would reduce to 66.7%. It means that 2 of the three computations are providing the actual desired output of the DCT module. By performing the redundant computation of difference input data less frequently, the useful throughput can be improved from the worst case. Fig. 3.14 shows the overhead in terms of throughput reduction. The plot provides the frequency of comparisons in terms of *Comparisons per Row* (CPR) vs. throughput reduction. As it is evident, reducing the frequency of redundant computations for the comparison purpose, the throughput reduction can be improved.

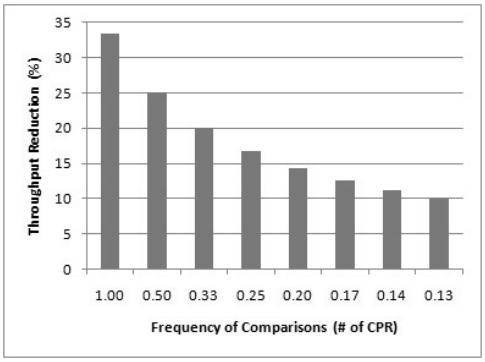


Figure 3.14: Throughput reduction of temporal Heterogeneous CED arrangement

Amorphous Slack (AS) Fault-Handling Methodology

To achieve fault-handling operation, we propose an *Amorphous Slack* (AS) technique to time-multiplex the processing PRRs for different functions and compare their outputs with those from the active modules in the logic datapath. A discrepancy between the outputs of two modules results in them remaining in the *Suspect* pool, whereas the agreement marks them as *Healthy* after the evaluation window elapses. This diagnosis procedure runs concurrently with DSP processing, without decreasing signal processing throughput. Each processing slack can check multiple distinct functional blocks, therefore being area efficient, by leveraging the FPGA's inherent property of reconfiguration.

We consider a typical DSP application which can be pipelined into multiple stages to accelerate the throughput. Consider a FE which can be partitioned into multiple PEs. Some of the PEs operate as Reconfigurable Checker Elements (RCEs) for discrepancy checking purposes while others are kept in the throughput datapath for computation purposes. The total number of checker elements, designated as *slack* denoted by N_s, available for comparison purposes can be varied depending upon input signal characteristics, area margin, and power budget. These RCE can either be spares reserved at design-time, temporarily vacated PEs during runtime, or part of another FE performing

some other task of lower priority. The term RS is used for the PEs corresponding to the first two cases. Algorithm 1 is used for fault isolation purpose in a core containing N PEs. Upon identifying faulty PEs, their functionality is assigned to healthy PEs which may either be slacks reserved at design time or some PEs computing lower priority-functions. In case of a DCT, the DC-Coefficient computation function is more significant than AC-Coefficients computing functions since the DC-Coefficient contains the most content information about a natural image.

Algorithm 1 Fault-isolation algorithm

Require: N, N_s
Ensure: $\hat{\Phi}$

1: Initialize $\hat{\Phi} = [x\ x\ x\ ...\ x]^T$, $i = 1$
2: **while** $(\{k|k \in \hat{\Phi}, k = 0\} = \phi)$ **do**
3: Designate PE$_s$ as checker(s) $(N + 1) \leq s \leq (N + N_s)$
4: **while** $i \leq N$ **do**
5: Reconfigure RCE(s) with the same functionality as PE$_i$
6: Perform N-Modular Redundancy (NMR) majority voting to identify at least one healthy RCE, $\hat{\phi}_i \leftarrow 0$ for PE$_i$ which shows no discrepancy then go to step-11, $\hat{\phi}_i \leftarrow x$ otherwise
7: $i \leftarrow i + 1$
8: **end while**
9: Move the RCE by updating $N = N - N_s$, Re-initialize $i = 1$
10: **end while**
11: Use a healthy RS to check all other PEs

The AS fault handling scheme identifies the faulty PE(s) by employing the RCE(s) as follows. `Step-1` of Algorithm 1 initially labels all PEs as Suspect. An entry $\hat{\phi}_i = 1$ in a vector $\hat{\Phi}$ of length $(N + N_s)$ stands for faulty nature of the PE$_i$, $\hat{\phi}_i = 0$ for healthy PE$_i$, and $\hat{\phi}_i = x$ for suspected PE$_i$. Initially, the set containing tested and verified fault-free *Healthy* PEs is an empty set (ϕ) as labelled `step-2`. The RCE can either be the blank PEs available in the system, some low-priority PEs, or PEs temporarily decommissioned from another FE. Initially, the RCE (or multiple RCEs) is reconfigured with the same functionality as that of the most important functional PE, for example, the module for computing DC-coefficient (`step-3` and `step-5`).

The location of a faulty PE is detected by performing the discrepancy check in an NMR arrange-

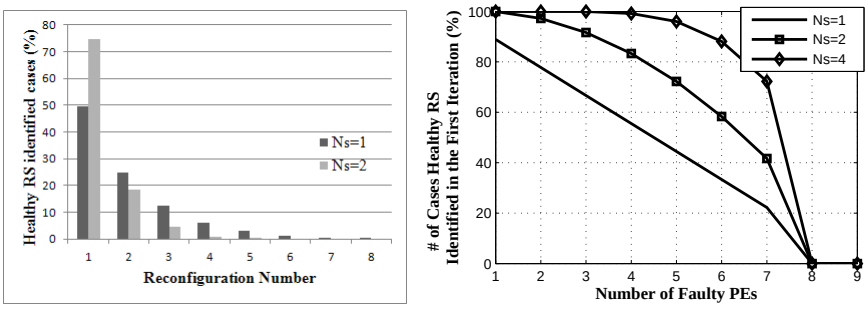
ment (`step-6`). For DMR, faulty status of one of the modules whereas for NMR faulty status of more than $N - 2$ modules marks each of the instances as *Suspect*. Therefore, we proceed to reconfigure the RCE with the second priority function and so on (`step-3`). Once an agreement between two modules over a complete evaluation window is observed, the two modules are declared as *Healthy* and their fitness state is updated (`step-6`). The identification of a healthy RCE implies we do not need to reconfigure the PEs as checkers further. A healthy RCE can be used to check the fitness of all the modules (`step-11`). The discrepancy of a suspected module in pair with a healthy module reveals its *Faulty* nature. On the other hand, an observed discrepancy between suspected modules does not provide any information and keeps them marked *Suspect*. If a *Healthy* RCE is not identified in the first iteration even after reconfiguring with all of the functions in the datapath, it is moved to the next PE, and so on (`step-9`). Upon the completion of fault isolation, the priority functions are moved to the *Healthy* PEs, achieving recovery.

The number of RCE to be employed depends upon the design margin desired within the resources available. If higher quality is to be maintained during the fault handling process yet no more PEs are available to be spare to serve as checker, then a smaller N_c should be selected. On the other hand, to speed-up the fault isolation process, a larger number of checkers are desirable. We will discuss in later chapters that input signal characteristics can be exploited to free some PEs to deploy them as checkers.

Simulation Results

First, we validate the proposed fault handling flow for the DCT core. The core consists of 8 PEs, each PE computing one coefficient of the 8×8 DCT. Fig. 3.15(a) shows the number of group reconfigurations required to isolate the faulty modules. For example, if the current DCT mode is 8×8 and one RS is available, approximately 50% of the fault scenarios are successfully isolated

in the first iteration. As there are total 9 modules, there are 512 possible combinations of the faulty/healthy modules. Out of these 512 cases, about half of the cases require only one iteration to find a healthy RS. Fig. 3.15(a) shows that increasing the number of RS, fault isolation process is accelerated. Fig. 3.15(b) shows the probability of isolating the faulty modules in the first iteration for different number of RS. Compared to a single slack, more failure scenarios are resolved with two slacks.



(a) Number of Reconfigurations required for fault isolation

(b) Probability of finding a Healthy RS in the first iteration

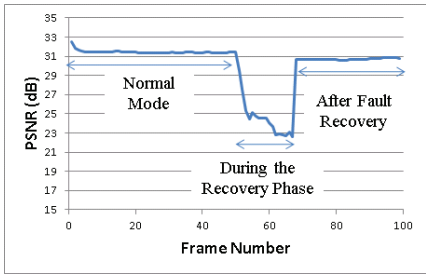
Figure 3.15: Fault isolation using AS technique

Case Study-1: Video Encoder

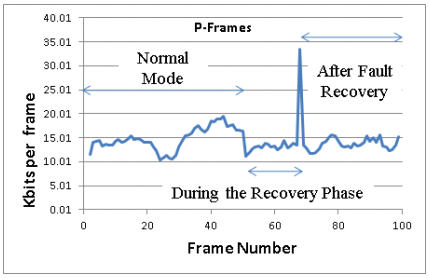
To analyze the quality degradation of a faulty DCT core during the AS fault-handling process, the H.263 video encoder application is executed on the on-chip PowerPC processor of a Virtex-4 FPGA provided on Xilinx ML-410 development board. The DCT module is implemented in hardware. A 256 MB memory module is used which can hold the code as well as it provides the data memory required to hold the images.

The data from the first stage of the DCT is not overwritten, rather it is kept in its own space in the frame buffer. Xilinx PlanAhead is used for PR flow while the software and hardware system is built

using Xilinx Platform Studio (XPS). Various PRRs are defined where each PRR corresponds to a PE of the DCT core. Xilinx Internal Configuration Access Port (ICAP) is used for downloading the partial bitstreams from external compact flash. The Xilinx System Advanced Configuration Environment (ACE) is a controller to manage configuration data [96]. It provides an interface between CompactFlash and the FPGA. This controller is connected in slave mode over the PLB bus and the embedded processor can read the bitstreams stored on the Compact Flash. The combined ACE file consisting of full system reconfiguration file (`.bit`) and the executable file (`.elf`) can be stored on Compact Flash. The FPGA chip is configured with the stored ACE file upon a power-ON event.



(a) Graceful degrdation of PSNR during and after fault-handling



(b) The bitrate of the compressed bit-stream

Figure 3.16: An operational example in a faulty scenario of video encoder

An example of the video encoder in faulty scenario is presented in Fig. 3.16(a). The faulty situation of PE_1 and PE_4 is examined here. The healthy nature of the RS makes it possible to isolate the faulty PE in the first iteration in which two reconfigurations are involved. As soon as the RS output is compared with PE_2 which is healthy, the RS is identified as healthy. As the faulty PE_1 was performing an important function, that is, the computation of the DC coefficient, therefore a healthy PE is assigned to this functionality. Fig. 3.16(a) illustrates that the quality degradation is spanned to a few frames in this case. The bit rate for the fixed Quantization Parameter (QP) QP=10 is also shown in Fig. 3.16(b).

The quality degradation of images in the frame buffer of the video encoder depends upon the location of faults. For example, the faulty behavior of a PE computing the DC coefficient impacts the video quality more than that of a PE computing the higher frequency AC coefficients. Therefore, recovery of an important function PE provides more improvement in the quality of results. Fig. 3.17 shows improvement in PSNR in case of different faulty PEs location, for three QP values. The scheme is more advantageous for small QP value, i.e., higher visual quality of image. Similarly, recovering from a case when a PE computing DC coefficient is faulty (i.e., PE_1), is the most beneficial.

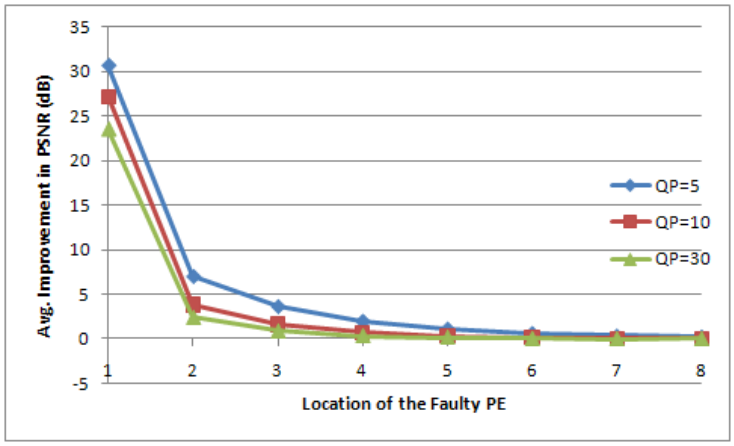
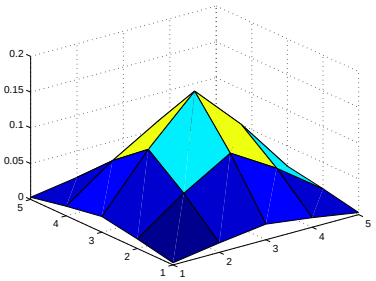


Figure 3.17: Improvement in average PSNR after fault recovery

Case Study-2: Edge Detector

The sustainability of edge detecting applications is desirable in harsh operating environments [97]. A Canny edge detector [98][99] is popular for image-processing due to its enhanced edge detection capability. Therefore, we evaluate the behavior of faults in a Canny edge detection module. For this purpose, as shown in Fig. 3.18(a), a 5×5 Gaussian Kernel is used for smoothing phase of the detector. We employed a distributed architecture where the convolution operation is performed

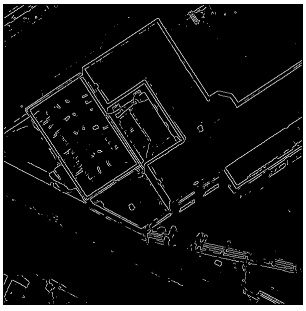
by multiple PEs to accelerate the performance of the edge detection. Fig. 3.18 illustrates the qualitative result of fault-handling for an image in the dataset available online [100].



(a) 5×5 Gaussian Kernel for smoothing

(b) Original image

(c) Edge detection by a faulty detector



(d) Edge detection after fault recovery

Figure 3.18: Gaussian kernel and qualitative results

Distance-Ranked Fault Identification (DRFI)

Fault Detection

A pair of configurations bitstreams is randomly selected to realize a CED pair. Only those configuration pairs can be instantiated which utilize mutually exclusive device resources. Mutual exclusion

can be ensured either by virtually dividing the chip into two halves[18], keeping a record of the utilized tile's locations during runtime, or using two FPGAs for a given circuit. An instantiated pair provides the desired DMR which can be used for error detection as illustrated in Fig. 3.19. In the following discussion, the configurations in a DMR arrangement are referred as the CUT and the RS corresponding to Active Circuit-Under-Test and Reconfigurable Slack, respectively. A discrepancy between the outputs of the two instances in DMR arrangement reveals the faulty nature of at least one of those configurations. Afterwards when a discrepancy in the outputs occurs, *fault detection* is asserted and fault handling methodology is initiated. The problem of identifying healthy configurations out of suspected configurations is then formulated as a system-level diagnosis problem.

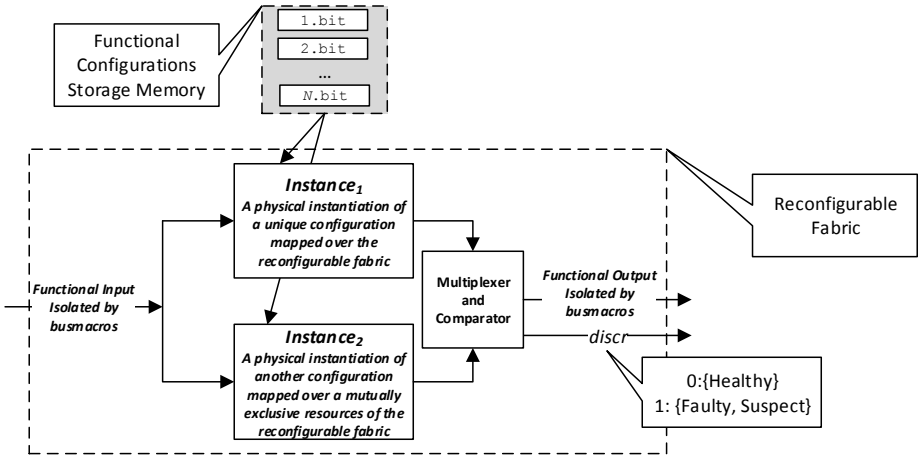


Figure 3.19: A CED arrangement of a functional element

System-Level Diagnosis of Hardware Configurations

The fault(s) occurring in an FPGA chip may affect multiple circuit implementations in the configuration pool. Thus, after fault detection, the health of all of the configurations is *Suspect*. The

objective becomes identifying correct configurations which utilize pristine resources. Formally, given a pool of N configurations out of which N_f configurations are faulty, the objective is to identify N_h fault-free configurations that utilize pristine resources. At least 2 healthy configurations are necessary to realize a DMR arrangement after fault recovery.

Fig. 3.20 outlines the scope, diagnosis approaches, and metrics of this paper. The diagnosability formulation for identifying faulty nodes is developed herein using a syndrome function. The three diagnosis algorithms of *Exhaustive Evaluation approach, the State Transitions approach, and the DRFI approach employing PageRank technique* developed are described in Sections 3, 3, and 3, respectively. Section 3 reports experimental results for MCNC benchmark circuits and H.263 video encoder's DCT hardware core.

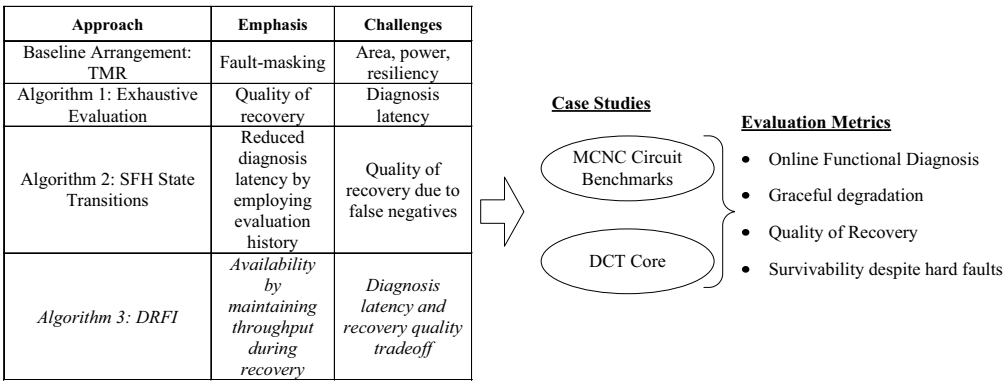


Figure 3.20: Online fault-diagnosis strategies evaluated herein

The same diagnosis formulation applies to each of the three algorithms developed and is described first here. Given an undirected graph $\mathbf{G}(V, E)$ of vertex set V and edges set E, the diagnosis objective is to identify faulty nodes. The nodes of $\mathbf{G}$ correspond to either PEs or processors in a multiprocessor network connected through an interconnection network. The diagnosis process is described in terms of CED comparisons to identify discrepancies. However, the formulation is not

restricted to a pair-wise comparison. Instead, the fault diagnosis process can utilize N-Modular Redundancy (NMR) in accordance with availability of resources. NMR is a generalization of TMR where $N \geq 2$ modules provide $N - 1$ redundant instances, which has found applicability in adaptive fault-handling [56] [101].

An element (u, v) in the edge set E indicates the feasibility that the output from corresponding PEs can be compared. Let the actual fitness states of nodes be represented by vector Φ, and the fitness states estimated based upon the fault-diagnosis process by vector $\hat{\Phi}$. We define the *Connectivity Matrix* $\mathbf{C}$ to show the comparison performed between two nodes in $\mathbf{G}$. Thus, an entry $c_{ij} = 1$ denotes that a comparison between node i and node j is performed. *Syndrome Matrix* $\mathbf{\Psi}$ indicates the outcome of comparisons. An entry ψ_{ij} of this matrix denotes comparison outcome corresponding to the outputs of node i and node j. Both of these matrices are symmetric about the diagonal due to commutativity of pairwise comparison for discrepancy.

$$\mathbf{\Psi} = \begin{bmatrix} 0 & \psi_{12} & \dots & \psi_{1N} \\ \psi_{21} & 0 & \dots & \psi_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{N1} & \psi_{N2} & \dots & 0 \end{bmatrix} \tag{3.10}$$

Where $\psi_{ij} = 1$ indicates that output from node i and j is discrepant for the same input, $\psi_{ij} = 0$ shows their agreement, while $\psi_{ij} = x$ stands for the case when no comparison has been performed between the corresponding nodes. A $\psi_{ii} = 0$ on the diagonal corresponds to the comparison outcome for a node i with itself,

The syndrome matrix $\mathbf{\Psi}$ can be used to estimate the fitness states of nodes in $\mathbf{G}$ under certain condition as we will discuss 3 diagnosis methods in further sections. Thus, faulty nodes can be identified based upon the syndrome matrix values. After fault detection, all the entries of $\mathbf{\Psi}$ except those on the diagonal are initialized with x implying that the health of all the PEs is suspect. The

following identifies the condition for healthiness, with the estimated fitness vector being updated accordingly:

Condition: $\psi(i,j) = 0$ for any $1 \leq i \leq N$ and $1 \leq j \leq N$, where $i \neq j$ and $c_{ij} = 1$
Update: $\hat{\phi}_i = 0$

Exhaustive Evaluation

To address the identification of faults without loss of generality, it is assumed that no correctness information is known a-priori for the functionality of the bitstreams in configuration pool. In particular, there are no configurations that are known to be fault-free nor resilient to faults. Such components have been referred to in the literature as *golden elements* [76]. Golden elements are subcircuits which are assumed be remain fault-free throughout the duration of the mission. As there are no *golden* configurations by which we can examine the fitness of other *Suspect* configurations, one unsophisticated way is to exhaustively evaluate all the configuration pairs in a mutual checking paradigm as given by Algorithm 2. Given a configuration pair instantiated in a CED manner, a complete agreement in their output throughout all the possible inputs affirms their healthy nature.

Assume that an I-input circuit is instantiated in duplex manner. The cardinality of the input set is 2^I and all distinct combinations of input samples need to be applied to evaluate the behavior of the circuit to the entire input space. If the number of defective configurations is not known a-priori, an upper bound on diagnosis time in terms of number of input evaluations can be derived as given in the following: the number of ways in which k objects can be chosen out of a set of N objects is given by Binomial Coefficient [102]. To realize a CED pair, two configurations are selected out of the pool. Thus,

Number of all possible configurations pairings = $\binom{N}{2}$

Algorithm 2 Exhaustive evaluation of configurations for fault diagnosis

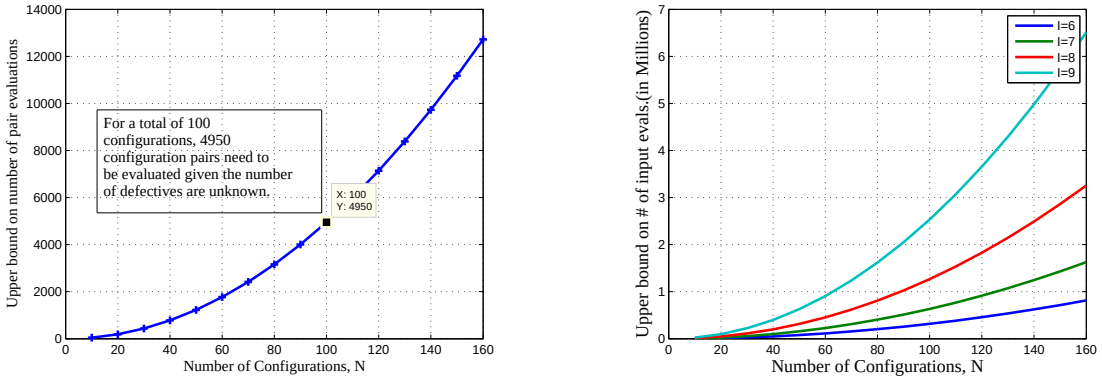
Require: N

Ensure: $\hat{\Phi}$

1: Initialize $CUT = 1$
2: **while** $CUT < N$ **do**
3: Designate the CUT as active configuration, and $RS = CUT + 1$ as slack. Then, the pair under test is given by: $V_{CED} = \{CUT, RS\}$
4: Perform concurrent comparison to the same inputs for various edges of the graph represented by connectivity matrix $\mathbf{C}$
5: Update the Syndrome Matrix $\mathbf{\Psi}$ based upon comparisons outcome of all possible inputs
6: **end while**
7: Given $\mathbf{\Psi}$, isolate the faulty nodes:
$\hat{\phi}_i \leftarrow 0$ and $\hat{\phi}_j \leftarrow 0$, if $c_{ij} = 1$, and $\psi_{ij} = 0$
$\hat{\phi}_i \leftarrow 1$ if $\hat{\phi}_j = 0$, $c_{ij} = 1$, and $\psi_{ij} = 1$

Total number of input evaluations required to test all configuration-pairs = $\binom{N}{2}2^I$

Fig. 3.21 shows the upper bound on number of reconfigurations required to identify faulty configurations by duplex evaluations for various configuration pool sizes.



(a) Upper bound on number of reconfigurations required to isolate faulty bitstreams

(b) Upper bound on number of input evaluations required to isolate faulty bitstreams

Figure 3.21: Fault-diagnosis cost of exhaustive evaluation method

The SFH Fitness States Transitions Diagram Method

As a competing approach to the DRFI technique, we consider a state transitions diagram method based upon the *Suspect, Faulty, and Healthy (SFH)* fitness transitions of the configuration bit-streams. An individual configuration undergoes different transitions in its fitness state throughout the life of a circuit. After fault detection, the fitness state of every configuration is *Suspect*. If two configurations show complete agreement in a given *Evaluation Period, E*, both are declared as *Presumed Healthy*. However, if a *Suspect* configuration exhibits discrepancy with a *Healthy* one, it is marked as *Faulty*. The state transition diagram is illustrated in Fig. 3.22. The objective of the state transitions flow is to identify *Healthy* configurations in a pool of *Suspect* configurations which in turn, helps to identify *faulty* items. The problem is similar to the counterfeit coin identification problem [103] with a restriction that only two coins can be tested at a time.

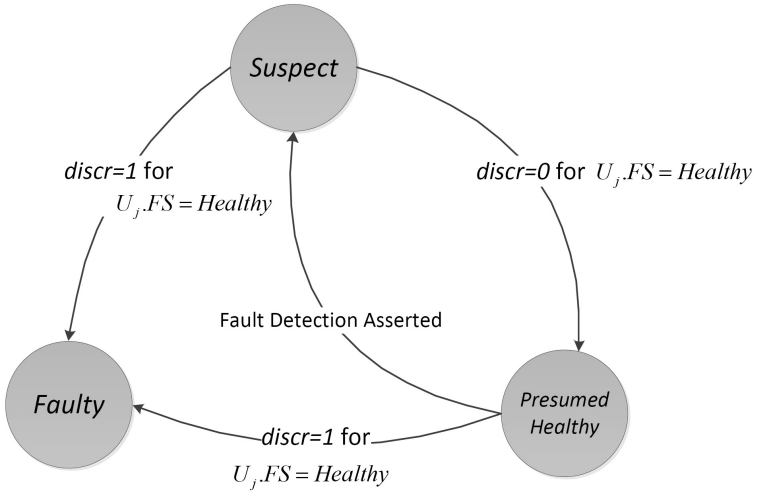


Figure 3.22: Fitness states of a design configuration during a circuit's life time

The SFH method is evaluated using monte-carlo simulation of the configurations' behavior. Let, u_i represent configuration labels $\forall 1 \leq i \leq N$ for a configurations pool V of size N. The number of healthy configurations are $N_h = N - N_f$, and we identify them using discrepancy information. For

Algorithm 3 SFH state transition algorithm for functional diagnosis of configurations

Require: N

Ensure: $\hat{\Phi}$

1: Initialize $CUT = 1$
2: **while** $(V_h == \phi)\&(CUT < N)$ **do**
3: Designate the CUT as active configuration, and RS as slack. Then, the pair under test is given by: $V_{CED} = \{CUT, RS\}$
4: Perform concurrent comparison to the same inputs for various edges of the graph represented by connectivity matrix $\mathbf{C}$
5: Update the Syndrome Matrix $\mathbf{\Psi}$ based upon comparisons outcome during an evaluation interval
6: Increment the reload number $n_r = n_r + 1$
7: Given $\mathbf{\Psi}$, isolate the faulty nodes:
$\hat{\phi}_i \leftarrow 0$ and $\hat{\phi}_j \leftarrow 0$, if $c_{ij} = 1$, and $\psi_{ij} = 0$
$\hat{\phi}_i \leftarrow 1$ if $\hat{\phi}_j = 0$, $c_{ij} = 1$, and $\psi_{ij} = 1$
8: **end while**
9: Use a healthy identified configuration to test all other configurations

this purpose, a configuration pair $\{CUT, RS\}$ is randomly chosen to be instantiated on the chip, where $\{CUT, RS\} \in V$. The CUT and RS correspond to active and slack configurations of a CED pair, respectively. Once a discrepancy is detected, the fitness state of all of the configurations is suspected.i.e.,

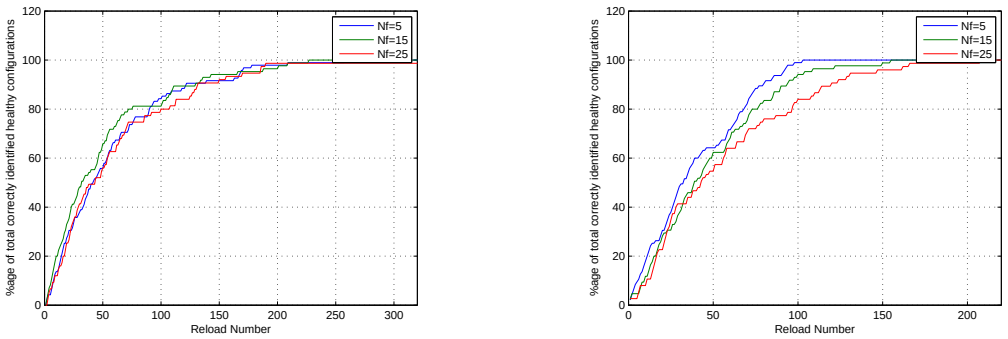
$$u_i.FS = Suspect; \forall 1 \leq i \leq N$$

As the knowledge about fitness of those configurations is not available initially, the estimated number of *healthy* configurations is $\hat{N_h} = 0$. Afterwards, another pair of configurations is randomly selected for instantiation while incrementing a variable *Reload Number,* n_r as listed in Algorithm 3. If two configurations completely agree in terms of their output throughout their instantiation period, their fitness state is updated to fault-free while incrementing number of *Presumed Healthy* configuration, $\hat{N_h}$ by 2.

$$v_{CUT,RS}.FS = Healthy, \hat{N_h} = \hat{N_h} + 2$$

To reduce the number of configurations reloads, as it costs reconfiguration latency which, in turn, would affect the throughput, a *without replacement* policy is also evaluated. In this strategy, an identified healthy pair is never re-instantiated during the diagnosis process under SFH approach.

Fig. 3.23 shows history of knowledge about various configurations while they are instantiated for evaluation. The $y-$axis of the plot shows percentage of total number of presumed healthy configurations correctly identified using the discrepancy information. i.e., $\hat{N}_h/(N - N_f)$. We can see from the plot that required number of configuration reloads increases with increased number of defective configurations, N_f. Moreover, the *without replacement* policy provides better results than the *with replacement* policy given no additional faults occur upon initiation of the fault diagnosis phase. In the following, we provide some probability analysis of the problem. The diagnosis problem of configuration-bitstreams can be formulated as given below:



(a) With Replacement

(b) Without Replacement

Figure 3.23: Identifying healthy configurations in a suspect pool

Given a collection containing a mix of defective and nondefective items, what is the probability that two items selected at random are nondefective [102]. To analyze this problem, first suppose $p(h)$ is the probability of selecting a single nondefective (healthy) item, and $p(f)$ is the probability of selecting a defective (faulty) item. Using the notation introduced in the previous section, $p(h) =$

$\frac{N_h}{N}$ and $p(f) = \frac{N_f}{N}$. Thus, the probability of selecting a nondefective pair is given by $p(hh) = p(h) * p(h|h)$ where $p(h|h)$ is the probability that the second item is nondefective given the first item was nondefective. The experiment of instantiating and evaluating a configuration pair is a Bernoulli trial whose outcome is either *success* when a healthy pair is selected or a *failure* when at least one of the configurations is faulty. The probability of k successes in the outcome of n Bernoulli trials with replacement strategy is given by binomial probability law [102]:

$$p(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

where p is the probability of success of a Bernoulli trial.

As each trial consists of picking a pair of items instead of a single item, the *probability mass function (pmf)* [102] becomes:

$$pdf(k) = \binom{n}{k} p(hh)^k (1-p(hh))^{n-k}$$

The *cummulative distribution function (cdf)* of a random variable X provides the probability that the event will be found with value less than or equal to k [102].i.e.,

$$cdf(k) = P[X \leq k]$$

The probability of finding nondefective items in a batch of 100 items with various number of trials is shown in Fig. 3.24. Out of one hundred items, 5 items are assumed to be defective. The pmf and cdf depend upon p, k, and n. For example, the pmf for $n = 100$ trials shows that the probability

of choosing exact $k = 91$ nondefective pairs is only 0.1331. In addition, the cdf plot shows that probability of success of selecting $k \leq 91$ healthy pairs in $n = 100$ trials is 0.6549.

To relate the probability analysis with the results from the monte-carlo simulation in Fig. 3.23, assume that we are interested in finding the probability of success greater than k given n trials. This measure relates to probability that each healthy configuration is selected at least once paired with a healthy other configuration, in a certain number of loading of configuration bitstreams of pairs. The *cdf* in Fig. 3.24 shows that probability of successes $k \leq 95$ is approximately 0.1 given 110 trials, implying that probability of successes, $k > 95$ is 0.9, i.e., $p(k > 95) = (1.0 - 0.1)$. Thus, we can expect that 90% of the trials would be successful in terms of selecting $k > 95$ nondefective pairs in $n = 110$ trials. It is evident from Fig. 3.23 that roughly 90% of nondefective items are isolated in 110 iterations under SFH transitions method of input evaluations of various pairs.

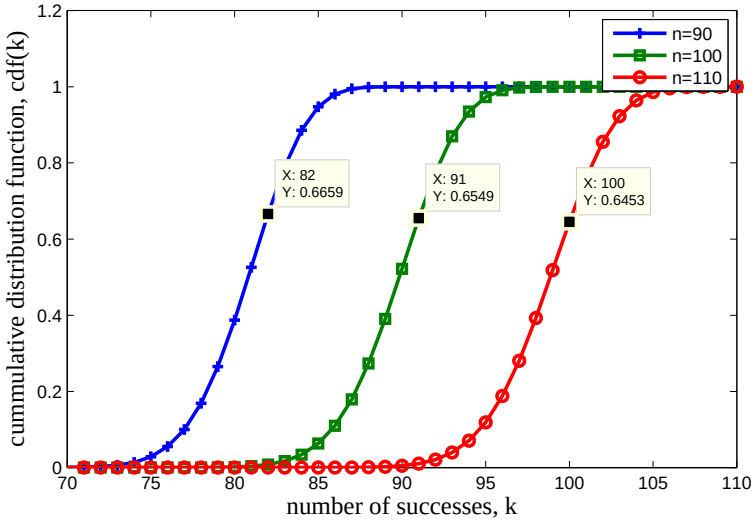


Figure 3.24: Probability of success for various trials with replacement

It is essential to note our assumption here that if two configurations are loaded for a given eval-

uation period, they will exhibit discrepancy at least once if at least one of them is faulty. This assumption may not be true in many cases as we discuss in the next section. This is acceptable since SFH is just providing a baseline for comparison of the proposed DRFI approach.

The DRFI Approach

The DRFI technique of fault-diagnosis using a functional testing paradigm fully exploits the dynamic reconfiguration capability of contemporary FPGAs. This technique utilizes the information about difference in output values of the duplex, in addition to discrepancy information. The diagnosis process begins with building a circuit similarity graph, and then applying the PageRank algorithm to compute the rank score of each node in the graph. The top μ configurations, having a score greater than the average score of a pool, are assumed to be fault-free and hence can be used by the system. However, if no healthy configuration exists, then the pool is sorted in ascending order according to the scores and higher score configurations are preferred. Algorithm 4 lists various steps in the DRFI technique of functional diagnosis to rank hardware configurations in a reconfigurable, fault-resilient hardware platform. Fitness and throughput heuristic can be customized by considering the throughput quality during diagnosis phase and fault detection latency tradeoffs.

Building the Circuit Similarity Graph:

The CSG is a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{W})$, where $\mathbf{V}$ is the vertex set, $\mathbf{E}$ is the set of edges and $\mathbf{W}$ is the weight adjacency matrix associated with the graph. This is similar representation used for image features in a feature similarity graph of [104]. Each entry of $\mathbf{W}$ represents the degree of match between the corresponding circuits in terms of their output.

For constructing the weight adjacency matrix $\mathbf{W}$, for each entry the corresponding pair of the configurations forming a CED arrangement are evaluated during an evaluation period. The Euclidean

Algorithm 4 DRFI algorithm of ranking the functional configurations

Require: N

Ensure: $\hat{\Phi}$

1: **while** (1) **do**
2: Select CUT by using the fitness and throughput heuristic
3: Designate the CUT as active configuration, and RS as slack. Then, the pair under test is given by: $V_{CED} = \{CUT, RS\}$
4: Perform concurrent comparison to the same inputs for various edges of the graph represented by connectivity matrix $\mathbf{C}$
5: Update the Syndrome Matrix $\mathbf{\Psi}$ based upon comparisons outcome during an evaluation interval
6: Build the Circuit Similarity Graph from Syndrome Matrix
7: Use PageRank Algorithm to rank the configuration pool according to the fitness assessment.
8: **end while**

distance between the outputs x_i and x_j represents the dissimilarity of the two circuits for online inputs. In general the Euclidean distance d between two points $\mathbf{x}$ and $\mathbf{y}$ in n-dimensional space is defined as [105]:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Where n refers to the number of inputs in an evaluation interval. Then, the distance is normalized so that the measure becomes the matching score of the two circuits. For this purpose, Gaussian kernel is used to compute the matrix entries w_{ij} that represent the pair-wise similarity of corresponding indices [106].

$$w_{ij}(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{3.11}$$

where σ^2 represents the variance of the Gaussian kernel.

A pair having consistently matched outputs with each other for a whole range of inputs will get higher score as compared to the configurations differing in their outputs. In addition, the configura-

tions completing agreeing during evaluation window are rewarded by subtracting a *Reward Score* from their associated DV. Consequently, a sparse matrix $\mathbf{W}$ is obtained for the configurations pool by randomly selecting different configuration pairs.

The size of the CSG seems a significant concern of the proposed method. It growths rapidly as the number of configurations increase. The size of CSG is directly impacted by the number of configurations created at design time and are determined by the extent to which fault capacity is desired. A large number of configurations at design time implies increased storage memory required, increased fault-handling latency due to evaluation time, yet an improved fault-coverage due to increased diversity of resource usage.

Ranking via PageRank:

Given the CSG, we are interested in assigning score to each node where each node represents a particular circuit configuration. The idea is to give more score to the circuit whose output is consistently matched with the other circuits. Faults injected at random locations affect the different circuit configurations in different ways, and hence the circuits behave inconsistently to the inputs when evaluated in pair with the other circuits. The CSG may be thought of as a graph similar to that of all linked web pages. The webpage which gets many votes or gets vote from high ranked pages, receives higher rank. Therefore, to rank the pages according to their importance we apply the PageRank algorithm, which is demonstrated in Section 3. For web, the rank vector is computed for n web pages by observing the hyperlinks coming to and leaving from the webpages. For n circuit configurations, the rank vector P_r is $1 \times n$ vector where each value of P_r represents the PageRank score of the corresponding configuration.

The PageRank of a page A is computed by [107]:

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{c(T_1)} + ... + \frac{PR(T_n)}{c(T_n)}) \tag{3.12}$$

where $PR(A)$ = PageRank of a page A

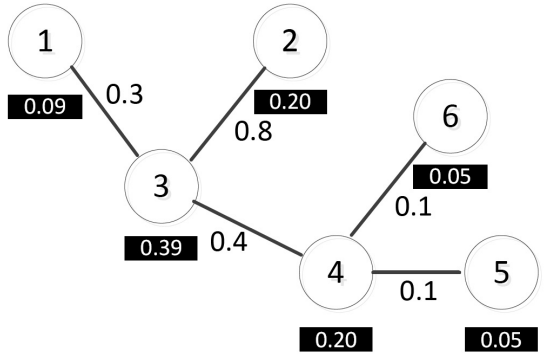
$T_1...T_n$ = The pages which refer to page A

$c(A)$ = Number of links going out of page A

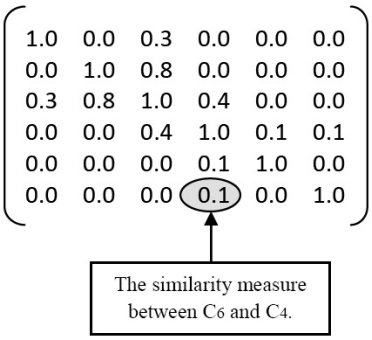
d = Damping Factor, empirically set to 0.85

The PageRank is a probability distribution over all the linked web pages, and a random reference occurs to a webpage with a probability given by its PageRank value $PR(A)$ [107].

An example of configuration ranking process after evaluating multiple number of reconfigurations is shown in Fig. 6.9. A pool of six design-time generated configurations are represented by nodes in a graph. The similarity of these configurations in terms of their output is given by the matrix, which represents the weights of edges between nodes. The PageRank value of the nodes is given below the node labels. As it is evident that configuration `label 3`'s similarity measure is higher among other configurations; therefore, it is ranked higher by the algorithm and thus preferred for fault recovery as described in the following section.



(a) A pool of 6 configuration bitstreams represented by a graph with similarity measure on edges and Pr in black



(b) The matrix representing edges weights

Figure 3.25: An example of configurations ranking

```
wire DONE_OBUF_4985;

  X_SFF #(
    .INIT ( 1'b0 ))
  DONE_3 (
    .CLK(CLK_BUFGP),
    .I(\count[6]_DONE_Select_14_o ),
    .SRST(RST_START_OR_1_o),
    .O(DONE_OBUF_4985),
    .CE(VCC),
    .SET(GND),
    .RST(GND),
    .SSET(GND)
  );

  X_LUT2 #(
    .INIT ( 4'h8 ))
  Mmux_DOUT1111 (
    .ADR0(DONE_OBUF_4985),
    .ADR1(aes_dout[84]),
    .O(DOUT_84_OBUF_5035)
  );
```

(a) Before Fault Injection

```
wire DONE_OBUF_4985;
wire DONE_OBUF_4985_tmp;
assign DONE_OBUF_4985_tmp=1; //Stuck-At '1' fault

  X_SFF #(
    .INIT ( 1'b0 ))
  DONE_3 (
    .CLK(CLK_BUFGP),
    .I(\count[6]_DONE_Select_14_o ),
    .SRST(RST_START_OR_1_o),
    .O(DONE_OBUF_4985),
    .CE(VCC),
    .SET(GND),
    .RST(GND),
    .SSET(GND)
  );

  X_LUT2 #(
    .INIT ( 4'h8 ))
  Mmux_DOUT1111 (
    .ADR0(DONE_OBUF_4985_tmp),
    .ADR1(aes_dout[84]),
    .O(DOUT_84_OBUF_5035)
  );
```

(b) After Fault Injection

Figure 3.26: An example of fault-injection into the simulation model of the circuit

The fault model used in the experiment work to evaluate the proposed fault handling scheme is *Stuck-At (SA)* model in which fault can occur at any of the LUT inputs. The SA model reasonably models the *permanent* affect of aging-degradation and radiation hazards on an FPGA device in a space environment. In addition, DRFI technique deals with the faults at a higher level, that is, by functional evaluation of the overall circuit and therefore, it should be capable of handling a wide variety of fault models. SA faults are injected in the simulation model of circuit generated by the Xilinx Xtool flow. We utilized our previously developed *Fault Injection and Analysis Toolkit (FIAT)* which invokes various commands of the Xilinx flow to study fault behavior. An example

of injecting SA fault to one of the LUT-inputs is shown in Fig 3.26.

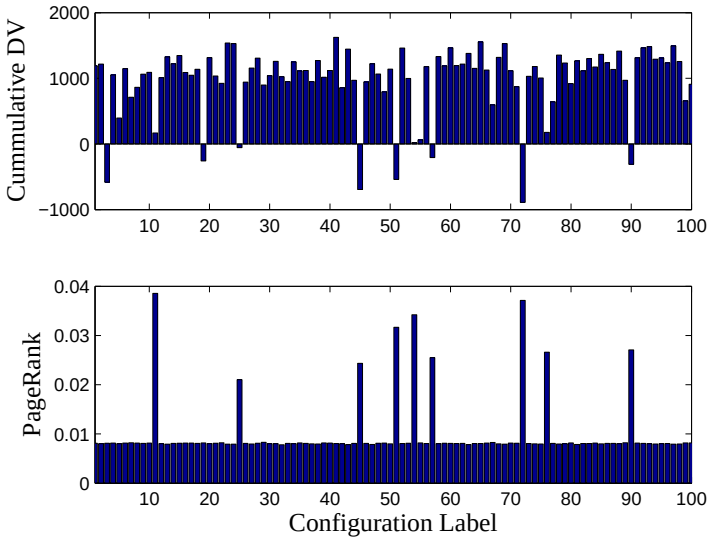
Experiment-1: MCNC benchmark circuits

To evaluate the DRFI technique of fault-handling, MCNC [108] benchmark circuit `z4ml` is analyzed in detail first, then recovery of other circuits in the MCNC benchmark are assessed. The benchmark circuits are implemented targeting a Virtex-4 device. We used the MATLAB implementation of the PageRank algorithm by Gleich [109]. The ISim simulator output is an interface to the MATLAB program which issues commands to the ISE.

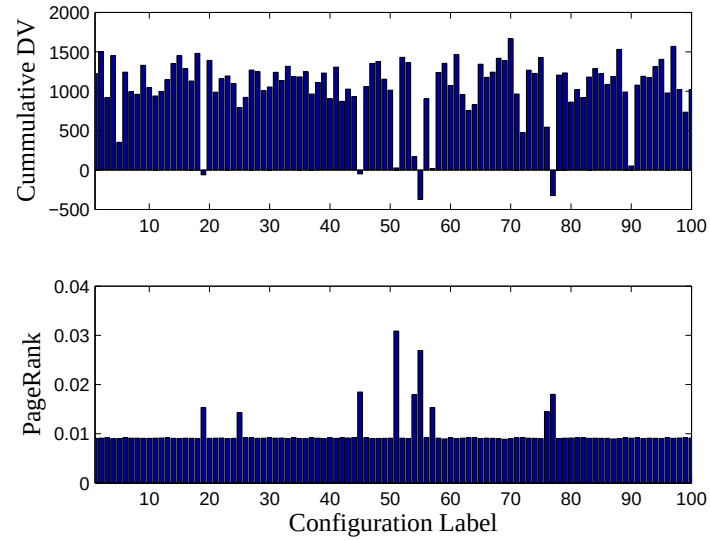
The selection of number of functional configurations is lower-bounded by the amount of diversity required to mitigate faults while upper-bounded by the tractability to handle the CSG. In this experiment, a total of 100 diverse configurations for the `z4ml` are generated at design time. Then faults were randomly injected into the post place-and-route simulation model affecting 86 circuit configurations thereby leaving only 14 designs fully functional. The healthy configuration labels are listed here: (`3,5,11,19,25,45,51,54,55,57,72,76,77,90`). The CSG is built by evaluating a pair of circuits to a subset of random inputs, specifically $E = 30$. A sliding window of size 20 with overlapping 10 is selected to evaluate the circuits in sub-pools. Instead of evaluating all exhaustive pairs, to all exhaustive set of inputs, the similarity matrix is built using a smaller set thereby resulting in a sparse CSG. The observed reduction in input evaluations is over 75% when using this approach for an MCNC benchmark circuit. By computing the PageRank for the resulting graph, the results are shown in Fig. 3.27 in which the PageRank value of each circuit implementation identified by its configuration label is plotted. The Cummulative Discrepancy Value (CDV) defined as:

$$CDV = \sum_{i=1}^{\tau} DV_i$$

Where τ denotes the evaluation interval as the number of input pairs applied. DV's are represented as signed value prior to converting into the evaluation distance. CDV is used to build the CSG, and then PageRank is computed given the CSG of the reconfigurable design. As it can be seen from the plots, the CDV of various configurations cannot assist much in differentiating the healthy configurations from faulty configurations. However, the corresponding PageRank values clearly distinguish the healthy and faulty groups.



(a) first run



(b) second run

Figure 3.27: CDV, and PR of various configurations for different simulation runs (a), and (b)

Analyzing the circuits with relatively high score in Fig. 3.27, we observe that they utilize fault-free resources. It is, however, worth noting here that as few as two configurations are needed at any given time for the circuit to produce validated output, although we have identified much more successfully. The cumulative *Consensus Similarity Value* history of first 10 configurations is plotted in Fig. 3.28. It is evident that the CSV of C_3 and C_5 increases with time as both utilize fault-free resources.

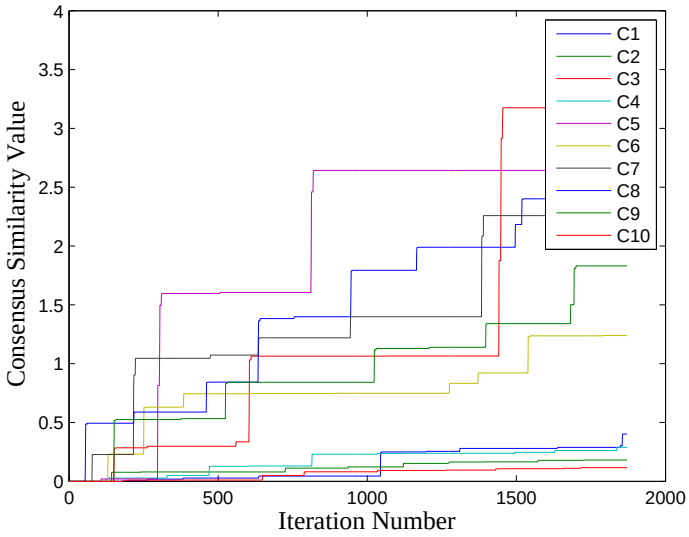


Figure 3.28: The discrepancy history of various configurations of the circuit

As compared to the exhaustive testing method which involves evaluating all possible pairs with all possible set of inputs, DRFI achieved considerable improvement, as evident by Fig. 3.29 which shows the results for other benchmarks. For example, number of configurations evaluations are reduced by 75% when using DRFI approach over the exhaustive testing approach in `misex1` benchmark circuit. Thus, considerable fewer configuration pairs are needed for evaluation using the DRFI approach.

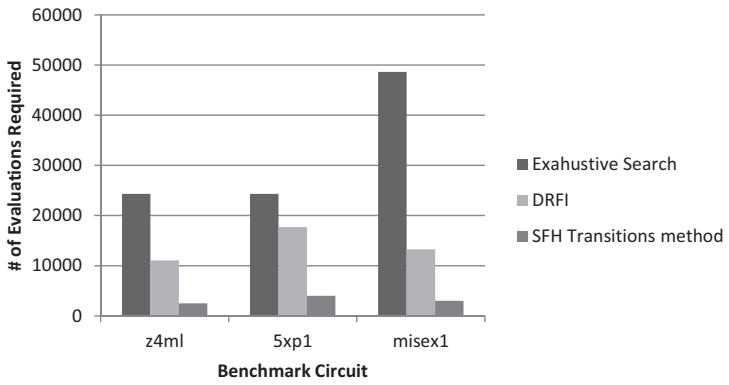


Figure 3.29: A comparison of fault-diagnosis methods for various MCNC benchmarks

The operation of the circuit in duplex manner is simulated in Fig. 3.30. The DV between outputs of the duplex circuit in each evaluation window is shown. During the *normal operation* period of the circuit, when no fault is present, the discrepancy value is zero. After one or more faults occur, the difference in output increases. During *repair process*, different pairs of configurations are loaded and evaluated using random inputs. After a sparse similarity matrix is build, the PageRank algorithm is executed. Once the configurations are ranked and identified correctly, the normal operation of the circuit is recovered. It should be noted that Fig. 3.30 is only for the illustration of the system's operation and not scaled by actual time, which depends upon the time complexity of the controller and the configuration under test (CUT).

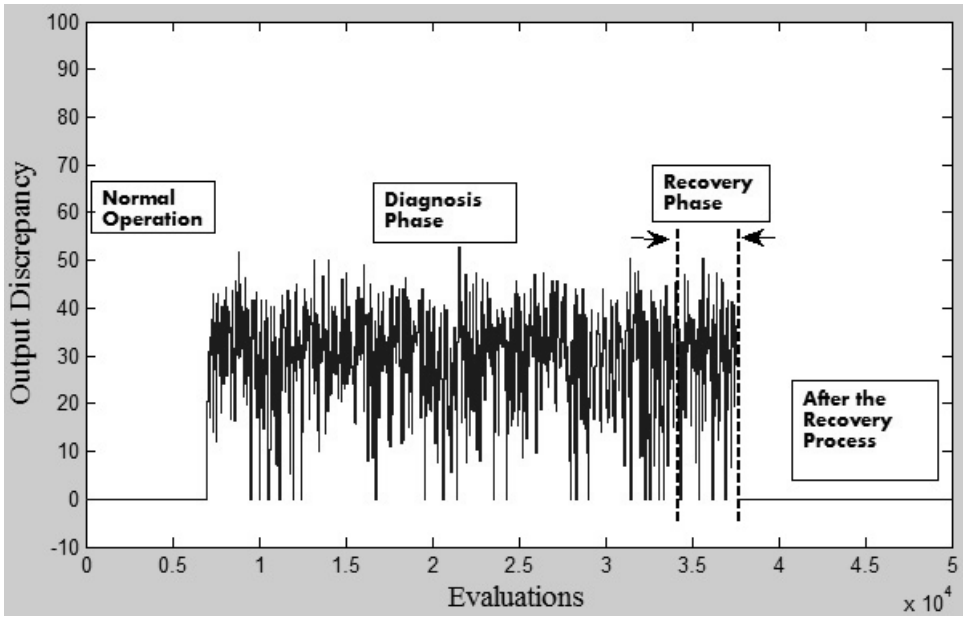


Figure 3.30: An operational example of a circuit on a x10^4 evaluations scale

Experiment-2: DCT core

The DRFI scheme is validated using H.263 video encoder's 1-dimensional DCT block implemented in FPGA using Xilinx ISE and PlanAhead for partial reconfiguration flow. There are 8 *Processing Elements (PEs)* computing the DCT coefficients [94], [110] of a row of pixels in 8x8 macroblock. Each PE's function is to compute one coefficient of the DCT function. For example, PE_0 computes the DC-coefficient, PE_1 computes the AC_0 coefficient and so on. The 2-D DCT is computed by using the 1-D DCT twice. Table 3.2 lists the resource utilization generated by the Xilinx tools for the DCT core interfaced with the on-chip PowerPC processor [96] which illustrates a significant reduction in reconfigurable resources when embedded multipliers are used.

Table 3.2: Resource utilization summary of the DCT core

Logic Resource	Used by the static region	Used by a PRR	Used by a PRR employing multipliers
Number of Slices	4516	169	62
Number of Slice Flip Flops	5961	79	44
Number of 4 input LUTs	6155	312	100
Number of FIFO16/ RAMB16s	45	0	0
Number of DCM_ADVs	1	0	0
Number of DSP48 Blocks	0	0	1

Fig 3.31 shows qualitative results of the fault identification scheme. A frame in the video encoder's frame memory is shown in Fig. 3.31(a). A total of 10 alternate configurations are generated at design time which utilize various PRRs in the chip. The DCT core is instantiated in the CED mode to provide error detection capability. Fig. 3.31(b) shows an intra-frame after a fault is injected in the PE_0 of the DCT core. The system is recovered by instantiating a pre-generated configuration which uses fault-free resources, as given in Fig. 3.31(c).

(a) fault-free system (b) faulty system (c) recovered system

Figure 3.31: An image in the frame memory of video encoder

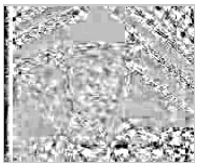
Experiment-3: Partial Recovery

In many signal processing applications, the underlying algorithms are inherently robust and a complete recovery is not necessary. While the latency of fault-diagnosis may be excessive to get diminishing returns, sometimes it may barely improve the overall health metric of the system.

On the other hand, a partial recovery can be quick and sufficient for the application. In a broad sense, provision of resilience in reconfigurable architectures for signal processing can take advantage of a shift from a conventional accurate computing model towards an approximate computing model [111],[112],[113],[20]. This significance-driven model provides support for operational performance which is compatible with the concepts of signal quality and noise.

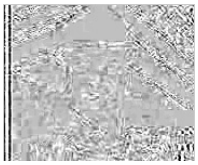
If all the configurations become faulty, then in the current scheme a complete recovery is not possible. However, we are interested in at least those configurations whose behavior exhibits more correct outputs than the others for the relevant online input subspace. In a case, in which no individual configuration in the design pool is operational due to the faults affecting all the configurations, it is beneficial to assign higher scores to the circuits which are relatively better. Fig. 3.32 shows results of a simulation in which all the pre-generated configurations are affected by fault. The discrepancy check is made on the DC and AC_0-coefficients output values which contain most of the information about the image content. Fig. 3.32(b) shows a case in which faults are injected in PE_1. As it can be seen that the image in Fig. 3.32(d) is visually better than that in Fig. 3.32(b) as the former is an output of the configuration which utilizes a fault-free PE_1. Although the recovered system utilizes a faulty PE_3, a graceful degradation is made by the proposed recovery solution. Thus, the image quality in the frame buffer reflects the benefit of such partial recovery.

(a) faulty system (PSNR=28.38dB)



(b) error of faulty system

(c) recovered system (PSNR=30.75dB)



(d) error of recovered system

Figure 3.32: Partial recovery results of the scheme

CHAPTER 4: SOFT-RESILIENCE USING AN ONLINE MULTI-OBJECTIVE GA

A self-aware signal processing architecture is proposed based on adaptive resource escalation which is guided by a multi-objective GA. The GA prioritizes tasks within a reconfigurable hardware fabric to maintain the quality-of-service and power consumption objectives. Attainment of these objectives is subject to the intrinsic reliability and performance of the computational elements in the resource pool. A health metric at the application layer, such as PSNR measurement in a DCT or Measure of Confidence in a Support Vector Machine (SVM) classifier, is used to assess throughput performance. When performance decreases beyond acceptable tolerances, the primary objective is to maximally recover output quality. The secondary objective is to minimize power consumption which also depends upon the input signal characteristics, in addition to the utilized computational resources. An adaptive guidance function for GA-driven recovery is proposed and validated for these objectives. It retains healthy processing elements in the throughput datapath to gracefully-degrade throughput by optimizing resource selection.

The developed scheme is evaluated using signal-processing case studies on a Xilinx Virtex-4 FPGA device. A SVM classifier with original accuracy of 75.7% is recovered in a low power configuration to 73.2% accuracy from a failure impact of 59.1% accuracy. Similarly, the PSNR of output from an evolvable hardware DCT module indicates maintenance of quality objectives by recovering the PSNR to 34.1dB from a fault impact of 25.2dB. When the GA was invoked during the resource escalation phase in these case studies, an individual configuration arrangement from the pareto set is found to realize advantageous quality and energy tradeoffs. Overall, the proposed technique of health metric based multi-objective online evolution provides a promising method to facilitate soft computing by evaluating modules subject to their actual real-time inputs, rather than exhaustive test vectors.

Autonomicity is a desirable property for signal processing architectures in dynamic real-time environments. Ideally, image processing systems should autonomously maintain the desired levels of accuracy with rapid convergence and optimized power consumption throughout a range of operating and reliability conditions. Survivability under these constraints can be enhanced by the provision of self-awareness properties at the system-level [114][115][116]. These attributes are most critical in real-time environments where the reliability of CMOS devices in nanoscale regimes is becoming increasingly sensitive to variations in temperature, process manufacturing tolerances, aging effects, and supply voltage stability [117][56][8][31]. For example, to achieve energy-efficiency in nanoscale CMOS circuits, voltage scaling [55] continues to be realized as one of the most effective methods. However, near threshold voltage operation of these circuits can manifest process defects and variations as run-time computational errors which appear in the context of signal processing applications as accuracy degradation [118][19]. This chapter develops a cross-layer signal processing architecture which uses self-adaptation to address these concerns.

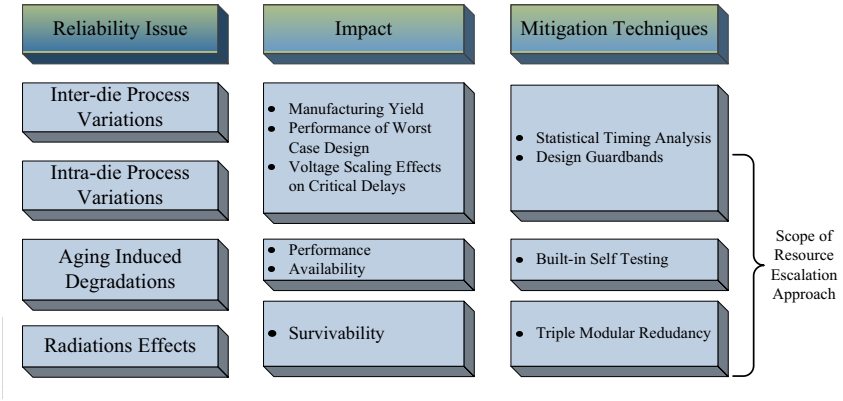


Figure 4.1: Reliability issues of digital systems built with deep submicron devices

Fig. 4.1 provides an overview of stability and reliability issues in sub-90 nm CMOS systems and

some popular corresponding mitigation techniques. This layered model can be adapted and leveraged in Digital Signal Processing (DSP) applications due to their inherent *soft-resilience* to errors. The soft-resilience property arises from redundancies in input data as well as the statistical nature of employed algorithms [112] and at the application-level from inexact perception of output quality by the user [113]. For instance, an example of soft-resiliency at the algorithm-level is Kalman filtering in which errors in prediction at a given instance are corrected in subsequent iterations. Soft-resiliency is compatible with a recent trend of attempting to sustain Moore's law by designing computing systems using various error-permissible computing models [111][113][119]. The inherent resiliency of signal processing algorithms allows some relaxation of exact computation to embrace this type of soft-computing paradigm. In particular, the provision of error de-sensitizing mechanisms and hardware graceful degradation is desirable to maintain output quality objectives.

We present a cross-layer soft-computing approach which leverages the different priorities that DSP tasks inherently have on the overall output accuracy. These are evaluated at runtime by monitoring a specific health metric or dynamic operating condition which is observed at the cognitive layer along with decisions to trigger adaptation. This avoids the complexity of search space overgeneration from rigid exhaustive fault coverage by handling only those subset of errors which affect the output quality beyond acceptable tolerances. Thus, the system adapts concisely to manifested errors while nullifying any false-positive demands. In addition, the need to synthesize test vectors with high resource coverage becomes unnecessary.

The proposed system is demonstrated using FPGAs which are widely chosen to realize signal processing applications in hardware due to their processing speed and potential for accelerated execution. As a computational platform, an additional major advantage of FPGAs is their runtime reconfigurability. Various reconfigurable regions can be defined at design-time for a circuit and later at runtime these regions can be re-assigned to alternative tasks dynamically. To reconfigure a Processing Element (PE) with an alternative task, the other regions of the device which are not

being reconfigured need not be removed from service. This online partial reconfiguration ability provides great flexibility for novel soft-computing approaches at the architectural level [120].

Herein, the problem of reconfiguration to maintain accuracy and performance is formulated as a multi-objective optimization problem. The term *configuration* will be used to denote a distinct mapping of tasks assigned to PEs. The event of *reconfiguration* will be used as a synonym for the task re-allocation process within a reprogrammable hardware fabric. As various tasks have different priority levels, the tasks are best mapped for execution when their priorities are directly correlated to the healthiness of underlying resources in the computational fabric. Thus, those configurations are preferred in which prioritized tasks are mapped to healthier elements in the resource pool of reconfigurable regions. The number of potential mappings can be quite large so we have employed the optimization capability of multi-objective Genetic Algorithms (GAs) to search the mapping space for throughput quality and power consumption alternatives. The proposed approach is evaluated for signal processing applications including a SVM and a DCT implemented in FPGA hardware. Performance metrics such as power consumption, measure of confidence, and PSNR demonstrate that a *health metric based multi-objective online evolution approach* achieves those objectives while incurring acceptable runtime overhead costs.

The following are the main contributions of this work:

1. The tradeoffs of reliability and power savings are formalized as a generalizable runtime mapping problem based on the underlying resource performance and operating workload.

2. A multi-objective GA approach is proposed for this mapping optimization problem in which a population of solutions is guided by a novel adaptive guidance function.

3. Instead of reserving redundant units for fault-detection, a throughput health metric is identified. Thus, fault-detection is feasible using a uniplex instance of the datapath without re-

quiring redundancy for error checking. This also allows a consolidation phase to distinguish transient conditions in the detection method.

4. Soft resilience is introduced as an iterative task remapping process to maintain the output quality metric within acceptable limits. Namely, an integrated diagnosis and recovery scheme is presented which neither requires a voting mechanism nor bringing the system entirely offline as recovery progresses.

Previous Techniques of Soft Resilience

Although a greedy algorithm like [45] is successful at small-scale optimization with single objectives (i.e., throughput), large-scale multi-objective problems necessitate meta-heuristic algorithms to explore the associated large search space. The proposed scheme is based on the technique of performing iterative reconfigurations until the system's output meets quality objectives. To avoid the requirement of redundancy which can incur significant area overhead in the case of cold-spares and power consumption in the case of replicated paths for comparison-based detection, the proposed approach leverages a health metric and the inherent computational priority in its system design. Such an approach is especially promising for signal processing applications which can accommodate a graceful degradation of functionality.

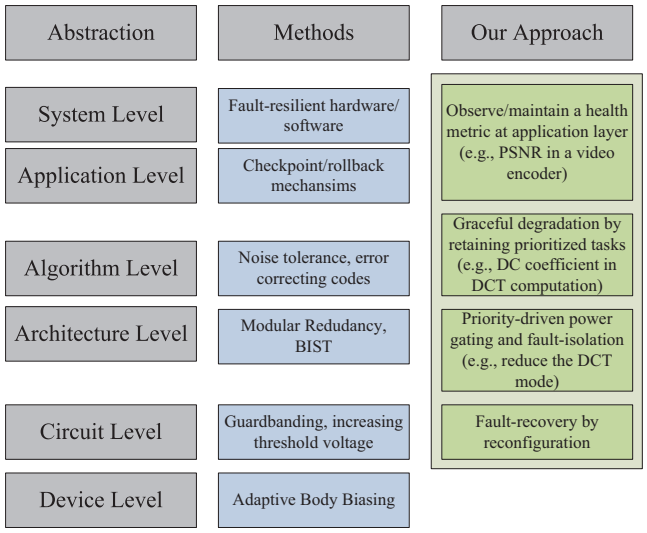


Figure 4.2: Hierarchy of fault-mitigation techniques at various abstraction levels

As illustrated in Fig. 4.2, there is a spectrum of techniques dealing with error-tolerance of signal processing systems ranging from the device-level up to the system-level. Fault-handling at the *architectural-level* is often oblivious to the error-mechanisms in the underlying hardware. For example, a Triple Modular Redundancy (TMR) arrangement is a technique in which a datapath is replicated to create three identical instances and then each output is passed into a majority voter for selection [63]. Although, a TMR scheme maintains all three instances in the datapath thereby achieving fault-masking capability, the resource overhead is considerable in both area and power consumption, even for the vast majority of the device lifetime which may be error-free. Namely, a TMR arrangement incurs a power consumption overhead that is approximately three-fold higher than a uniplex arrangement even if the voter overhead is negligible and throughput operation is fault-free. A Concurrent Error Detection (CED) arrangement detects faults by comparing the output of two replicas subjected to the same inputs [31]. A discrepancy reveals faulty nature of either instance without identifying which of the modules is faulty. Again, the area and power overhead are significant concerns in CED as they are doubled over the baseline design. As an alternative, Built

In Self Test (BIST) mechanisms diagnose faulty components by evaluating them with some test inputs generated by an Automated Test Pattern Generator (ATPG) to provide one-time or periodic fault-assessment. In practice, a BIST scheme rarely achieves 100% coverage, yet may generate false alarms [121]. Moreover, an evaluation of some test vectors may not necessarily correspond to the actual runtime scenario of a module under test. On the other hand, the proposed technique of *health metric based multi-objective online evolution* relies on the actual behavior of the signal processing module under runtime conditions. We show that an evolutionary inspired scheme of reconfiguration which correlates the output history information with task mapping can meet these goals.

Algorithmic-level fault-handling approaches exploit signal processing algorithm properties to make the system robust and error-resilient. Hegde and Shanbhag *et al.* [122] proposed an Algorithmic Noise-Tolerance (ANT) technique to compensate the errors introduced into DSP architectures due to voltage scaling. Voltage scaling has been an effective method of reducing power consumption, yet the correctness of throughput becomes an issue when the supply voltage is scaled beyond a critical voltage. To mitigate these concerns, the authors developed a prediction-based error control scheme which requires knowledge of the system transfer function which was a digital filter in their prototype case. Applying algorithmic-level fault-handling to video processing, Varatkar *et al.* [123] proposed a sub-replica of the motion estimation block to concurrently check the error-prone main block. Meanwhile for image processing, Kim *et al.* [56] proposed a soft voter employing a Bayesian detection technique. The soft voter is demonstrated to provide correct output in a Discrete Cosine Transform (DCT) based image coder. Lisboa *et al.* [124] proposed a fault-tolerance technique to mitigate faults in matrix multiplication algorithms, which comprise the heart of many signal/image processing applications.

Research have also studied area and accuracy tradeoffs for various computationally-intensive applications such as data encryption [125] and less-than-exact computation [20] for Signal processing

applications. To further reduce the power consumption or enhance the throughput of DSP applications, numerous architectural and algorithmic approaches have been taken over the years. To combat both challenges in a unified manner, this article specifies a multi-objective design paradigm using GAs. While area and delay multi-objective optimization using genetic algorithms [126] has been a well established technique in electronic design [127][128][129][7] prior to manufacture, the methodology developed herein adopts the concept of runtime adaptation to meet the performance and energy objectives according to the signal's characteristics.

Finally, power consumption remains one of the key issues in rapidly-scaling CMOS technology. This is especially true in both high-density deep submicron designs due to cooling considerations, as well as portable electronic systems where battery life, size, and weight are concerns. Although voltage scaling has been used to drastically reduce power consumption, this increases the circuits' susceptibility to faults, and hence the desirability for soft-resilient operation under these conditions. While there is a body of research work dealing with power versus fault-tolerance tradeoff at design time [130], there remains a need to develop runtime tradeoff techniques. Runtime techniques are also promising to handle faults in unforeseen mission-critical scenarios as well as commonly encountered manufacture-induced process variations that impact the yield, stability, and aging-behavior of commercial products. The presented *health metric based multi-objective online evolution* scheme addresses the issue of power consumption and quality tradeoffs through a novel runtime architectural adaptation technique formalized below.

Problem Formulation and Methodology

Consider a computing Array-Under-Test (AUT) realized by a set of N-PEs namely PE_1, PE_2, ..., PE_N each executing a task T_1, T_2, ..., T_N, respectively. The priority of the tasks assigned to the PEs is given by a vector $\mathbf{P} = \{p_1 p_2 \dots p_N\}$ having its i^{th} component denote the priority of

the i^{th} task. As a fault-recovery provision, a PE can be reconfigured to an alternative function or equivalently, a task can be re-assigned to an alternate PE at runtime. Given a homogeneous computing array of PEs, a reconfiguration controller can re-assign an alternative task to any PE in the array. Let the healthiness of resources which comprise the PEs be denoted by a vector $\mathbf{H} = \{h_1 h_2 \dots h_N\}$ as illustrated below. In the proposed fault-handling scheme, the defectiveness degree of PEs is assumed to be unknown. The formulation here allows utilization of a-priori information about the priority of tasks mapped at runtime as discussed in Section 4 and Section 6.

Power consumption in such an AUT can be reduced by power gating of some of the PEs [131],[132] which acts to exclude them from operation. The choice of PEs selected for use depends upon input signal characteristics, assigned tasks priorities, and desired quality levels. To maintain the generality of the notation without becoming restricted to a specific signal processing algorithm, consider a *zero-task* denoted by T_0 which corresponds to the power-gating OFF condition of the underlying computational resources for which the task has been mapped. In practice, a T_0 task can be realized by a power-gating technique in an ASIC implementation or by configuring a blank bitstream into a FPGA reconfigurable region. We have selected the latter approach for our case studies as we are utilizing a FPGA device.

A set of active PEs is defined as $V_a = \{PE_i\}; \forall_i T_i \neq T_0$. Thus, V_a contains those PEs which are assigned by non-zero task and operate to provide the throughput of the system. There is a one-to-one mapping of tasks to active PEs such that the cardinality of the set of active PEs is given by $|V_a| = N_a$ where $N_a \leq N$. In the following discussion, the terms processor *node* and *PE* are used interchangeably. It is worth highlighting the assumptions of the above formulation:

1. A PE node can be configured with any task, namely, a homogeneous array of PE resources is considered here.

2. Input data can be multiplexed to any or all of the PE nodes. This can be realized using bus

macros in the target FPGA platform, e.g., as per the Xilinx-specified partial reconfiguration based design flow [133].

Fig. 4.3 shows an architectural view of the proposed fault-handling approach employing pareto set solutions, health metrics, task priorities, computational resources, and on-demand power-gating. The computationally demanding portion of a signal processing application has been mapped to an array of PEs to accelerate throughput. The reconfigurable PEs array is managed by the reconfiguration controller to map tasks into the computational regions. A health metric is communicated from the software application to the reconfiguration controller. The overall software application can be executed on a PowerPC processor as such on-chip processors are already provided in most commercially-available FPGA chips. The value of the health metric will vary due to either input signal characteristics or hardware defects. To identify the latter case, a health metric outside nominal operating range triggers the fault-identification process. To keep the area overhead minimal, fault-identification is performed by a comparison-based discrepancy detector on a PE-scale resolution rather than at a system-wide resolution. In particular, an RS [45] region is utilized to consolidate a non-transient fault-detection of decreased health metric value. In particular, a RS is a single task-grained tile reserved as a cold-spare for the entire design; only one RS is needed regardless of N. Operationally, an RS is loaded to test suspect PEs successively, and in order of their priority of impact on the output quality. Namely, a discrepancy between the output of an active PE and RS indicates a transient or permanent hardware error. Thus, fault-identification is asserted without rendering any decision about exact location of fault being either in the active PE or the RS. Afterwards, the diagnosis and recovery process is carried out by the GA engine embedded in reconfiguration controller to locate which of these two is actually faulty. However, if no discrepancy is observed between the active PE and RS, then the health metric is assumed to have exceeded tolerance simply due to the input characteristics or due to a transient fault in the computational resources which has subsequently resolved. Thus, we focus below on the case where the active PE

and the RS outputs are discrepant.

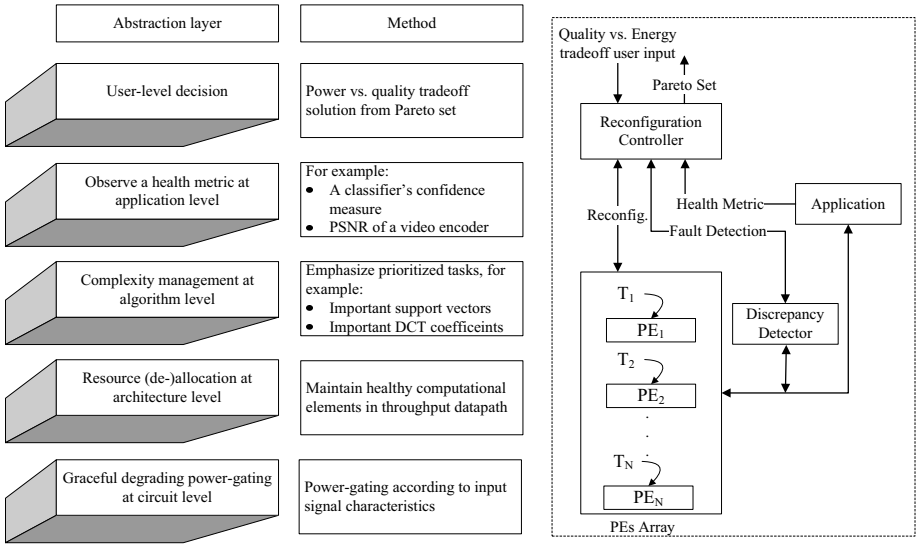


Figure 4.3: Cross-layer fault-handling architecture with hierarchical support

The fault-handling processes employs a data structure representing the task mapping to PE resources. The PE array and corresponding task mappings use a fixed-length chromosome in this formulation which is suitable for GA processing [1]. The genetic representation is illustrated in Fig. 4.4 which shows array of 7 PEs concurrently executing a set of tasks communicating with data memory for the tasks inputs and outputs. A PE can be configured with any task T_i where $0 \leq i \leq N$. For example, PE_1 is configured to execute the task T_1, PE_2 is configured to execute the task T_3, and so on. An example of the task-mapping chromosome is shown in Fig. 4.4. The number of fields in a chromosome is equal to the number of PEs in the processing array. The value of a particular field identifies the task number allocated to the corresponding PE. For example, the third field in the chromosome contains the value 4 implying that PE_3 is assigned to execute task T_4. It is worth noting the exemplified task mapping of PE_4 being allocated T_0 which corresponds to configuring a blank bitstream on this particular PE; zero or more PEs may be configured with T_0

based on the instantaneous or near-term throughput quality requirements. Such a dynamic assignment of blank tasks acts to reduce and dynamically optimize power consumption at the expense of some quality degradation whereby some functional task, for instance the corresponding DCT coefficient, is omitted from computation. The formulation of the tradeoff of these objectives is described below.

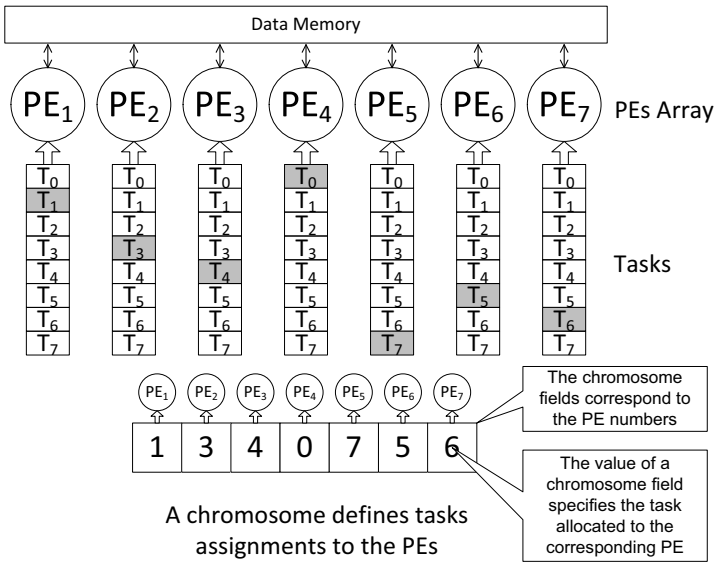


Figure 4.4: An array of 7 configurable PEs and its genetic representation

Multi-Objective function

The power versus quality tradeoff in DSP systems is formulized as a optimization problem using the composite function to be minimized given by:

$$f = w_1 f_1 + w_2 f_2 \tag{4.1}$$

where w_1 and w_2 are corresponding weights of the opposing functions f_1 and f_2. The functions f_1 and f_2 represent the throughput degradation and power consumption, respectively, of the current task mapping using the selected computational resources. An effort to improve f_1, i.e. minimize quality loss, results in degradation in f_2, i.e. consumption of more power, and vice-versa. The pareto solution set to this problem corresponds to a set of configurations aimed at exploring the design space of the quality versus energy efficiency tradeoff. The goal of soft-resilience is achieved by mapping prioritized tasks to the healthy resources, while energy efficiency is achieved by loading blank bitstreams into both failed and healthy PEs. Of course, disabling healthy PEs while saving power, degrades throughput as discussed below.

Throughput Degradation

The evaluation interval size τ is defined as the period of calculations over which the fitness assessment of an AUT is performed, expressed in units of the number of input instances. For higher throughput quality and accuracy over an evaluation interval, the following metric which is essentially a measure of Mean Squared Error (MSE), should be small:

$$f_1 = \frac{1}{\tau} \sum_{i=1}^{\tau} ||\Gamma_i - \ddot{\Gamma}||^2 \tag{4.2}$$

where $\ddot{\Gamma}$ is desired value of health metric. The health metric selected can include the PSNR, bitrate, measure of confidence, or other application-level quality of throughput indicator.

Power Consumption

Power consumption of an array of PEs is directly proportion to its size N. Therefore, a normalized power consumption measure is defined in terms of N and is given by:

$$f_2 = \frac{\sum_{\forall k \in V_a} \pi_k}{\sum_{i=1}^{N} \pi_i} \tag{4.3}$$

where V_a is the set of PEs assigned with non-zero tasks and π_k is the power consumption of the k^{th} task. Thus, the AUT's power consumption is maximized when all N PEs are assigned to have active tasks resulting in $N_0 = 0$. On the other hand, power consumption is minimized when all the PEs are assigned with zero-task assignment yielding $N_0 = N$, yet throughput quality in that case is also non-existent and hence not an option selected in practice.

The objective functions given in Eq. 4.2 and Eq. 4.3 are oppositional. A higher number of active PEs results in increased throughput quality at the expense of increased power consumption. On the other hand, power gating of the PEs results in reduced power consumption while incurring output quality degradation. A runtime multi-objective GA approach is used in finding a pareto optimal set as described below, thus spanning throughput versus power optimization, and also error soft-resilience, by a single strategy.

Guidance Function

Although a solution to the minimization of Eq. 4.1 is the objective realizes the desired soft-resilience operating point, the search space of the mapping problem is considerably large. For example, an exhaustive search will require $(N + 1)!$ reconfigurations in a cluster of size N to explore the search space in the worst case. Thus, exhaustive or randomized approaches can be

intractable for absolute minimization of large-size problems which render the practicality of non-guided search to be very limited. To this end, we propose to incorporate evaluation history information of the influence of mapping on throughput quality which further guides the population towards the pareto front. The history information of the individuals helps developing a health estimate of computational resources which in turn prunes the search space of the problem. Thus, adaptive guidance of the population using runtime healthiness estimate acts to benefit the convergence of the online multi-objective GA.

An a-priori knowledge of tasks' default priorities is generally useful in terms of carrying out a graceful degradation strategy, and is available in many cases such as the coefficient computing functions in the DCT core whereby the DC coefficient should be computed on the healthiest resource. However, such a knowledge of healthiness of computational resources is often dynamic and may be subject to soft-faults due to agressive voltage scaling, aging, and supply variations, or even permanent faults. Thus, it is beneficial to estimate the healthiness of computing resources at runtime to evaluate Eq. 4.5. This uses the nodes's output discrepancy history to develop its healthiness estimate. Thus the overall error observed in the output is automatically weighted by the priority level of its assigned task. Thus, $h_i(t) = 1/d_i(t)$ where $d_i(t)$ denotes the defectiveness estimate at evaluation instance t as follows:

$$d_i(t) = d_i(t-1) + p_j * |\Gamma - \ddot{\Gamma}| \tag{4.4}$$

where p_j = priority value of task j assigned to PE$_i$.

Thus, to calculate the defectiveness estimate d_i of a node i, the throughput degradation is weighted by the task-priority value and accumulated into the previous estimate of d_i. By employing the fault articulation history as well as the task priorities, the defectiveness estimate becomes an effective measure to guide the adaptation towards an optimal mapping.

The guidance function can then be defined as:

$$g = \frac{|\sum_{i=1}^{N} p_i h_i - \sum_{\forall_k k \in V_a} p_k h_k|}{\sum_{i=1}^{N} p_i h_i} \tag{4.5}$$

This measure g to be minimized guides the GA to find the pareto front while maintaining partial throughput during fault-resolution phase. Here, the tasks' priorities are weighted by the healthiness of the underlying resources on which the tasks are mapped to. As eq 4.5 reveals, a minimum value of g corresponds to the mapping when vectors $\mathbf{H}$ and $\mathbf{P}$ are highly correlated. That is, high priority tasks are mapped to healthier resources. Guidance function assists in guiding the evolution according to the fitness function when system is faulty. Otherwise, the fitness function continues to use f_1 and f_2 functions for throughput assessment and power optimization, respectively.

The proposed fault-handling methodology is summarized below:

1. detect when the application-level health metric exceeds tolerance,
2. consolidate non-transient fault-identification via a CED-based discrepancy-based detection using a RS,
3. invoke the GA during the healthy resource escalation phase of task remapping, and
4. select an individual from the obtained pareto set to finally map tasks on to the fabric based upon their quality and power consumption tradeoff.

Execution Results

Synthetic Nodes Simulation

To illustrate the process and the impact of the function given in Eq. 4.1, the approach is evaluated using an array of simulated nodes. For this purpose, a PE-array of size $N = 7$ is chosen and the fault scenarios are simulated by assigning healthiness values to the PEs as listed in Table 4.1. The priority values are assigned to various tasks such that T_1 receives the highest priority (i.e., the maximal value of 7 for an arrangement comprising 7 possible tasks, while T_7 receives the least priority, i.e.,the value minimal non-zero task value of 1. Thus, $\mathbf{P}$ vector's component values reflects the reverse ordering of Task Numbers. For example, in a DCT, Task T_1 would correspond to the computation of the DC coefficient. However in this illustrative example, for generality assume the effect of priority values on the overall output is unknown at this point. Therefore, it is not feasible to initially evaluate the first term of the objective function given in Eq. 4.1. Instead, the healthiness values are assumed to be already available in this scenario in the form of a monotonically decreasing linear function while the guidance function of Eq. 4.5 is considered to be the first term of the objective function of Eq. 4.1. The duration of the evaluation interval is considered as one sample here such that $\tau = 1$, i.e., the objective function is evaluated for every input in this synthetic nodes case study. It is worth mentioning here that although the healthiness and priority values are generated by a linear function in the synthetic nodes simulation case, these values can be substituted with results from any fault model and the impact on output quality in the practical case studies as discussed further below. For example, the fault impact is simulated by a stuck-at fault model and the corresponding PEs are evaluated for functional output in the practical case studies with favorable results.

Table 4.1: Example of priority values, $\mathbf{P}$, and healthiness of resources, $\mathbf{H}$

PE Number i	1	2	3	4	5	6	7
H_i	0.25	0.2143	0.1786	0.1429	0.1071	0.0714	0.0357
Task Number j	1	2	3	4	5	6	7
P_j	7	6	5	4	3	2	1

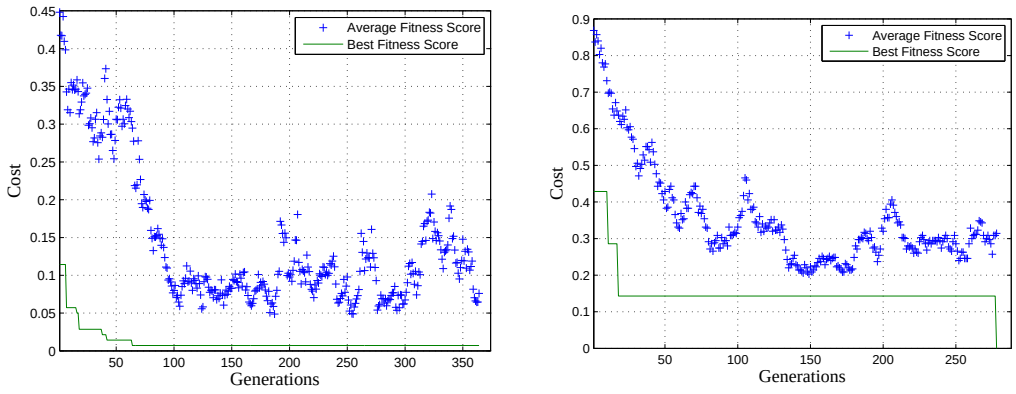
The GA parameters used are given in Table 4.2. The Population Size corresponds to various tasks configurations to the AUT. Migration parameters specify the individuals of population's movement among multiple sub-populations. The individuals are created by a uniform function while being selected using rank criteria [134]. The standard two-point crossover operator is used with the mutation option compatible with a standard GA [1]. The elite count parameter ensures that some of the best individuals are guaranteed to be propagated to the next generation [134].

Table 4.2: GA paramters

Parameter	Value
Population Size	25
Migration Direction	forward
Migration Interval	20
Migration Fraction	0.200
Population Creation Function	Uniform
Fitness Scaling Function	Rank
Selection Function	Uniform
Crossover Function	Two-point
Elite Count	2
Crossover Fraction	0.7
Mutation Rate	0.01

Figure 4.5 shows the throughput degradation and power consumption objective costs on the vertical axis for various iterations as two curves over time in units of generation number on the horizontal axis. The two curves depict the average behavior of the population as the upper scatter plot and the best-performing individual's behavior as the lower curve on each plot. The throughput degra-

dation is described in terms of the guidance function. The best solution reached by the GA was $\{1, 2, 4, 3, 5, 6, 7\}$ after 500 generations. As Figure 4.5 shows, the average behavior significantly improves within 100 generations, and then fluctuates due to the mutation operation. A sufficient population size together with a mutation function is necessary in order to diversify the population to reach a good solution in terms of meeting multiple criteria. The cost scores are defined in terms of the number of active PEs as well as the synthetic priority and health values. Then, after normalization, the unit-less ratios are the cost scores to be minimized. A converging trend of the cost plots after 100 generations implies that the proposed evolutionary methodology can achieve power and quality goals by employing the runtime-behavior information of the processing array. The global optimum solution for this problem is $\{1, 2, 3, 4, 5, 6, 7\}$ as it corresponds the resource escalation of the weighted prioritized tasks over the reconfigurable fabric. Thus, the GA is successful in finding a near optimal configuration in this problem, within a very reasonable number of generations suitable for runtime operation.



(a) Throughput degradation

(b) Power consumption

Figure 4.5: Cost functions

Figure 4.6 shows the pareto set of solutions of the Multi-Objective Online Evolution (MOOE) problem. Here, both costs, namely throughput degradation and power consumption are employed

to engage quality and energy efficiency tradeoffs, respectively. For example, a 40% tolerable behavior in terms of throughput degradation allows power consumption reduction to 30% of the maximum budget. A further reduction in power consumption is feasible as low as to only 10% yet only if approximately 80% throughput degradation can be tolerated. As the result shows, the proposed evolvable hardware MOOE recovery formulation allows finding a set of optimal solutions which facilitates design space exploration in terms of quality and energy efficiency tradeoffs.

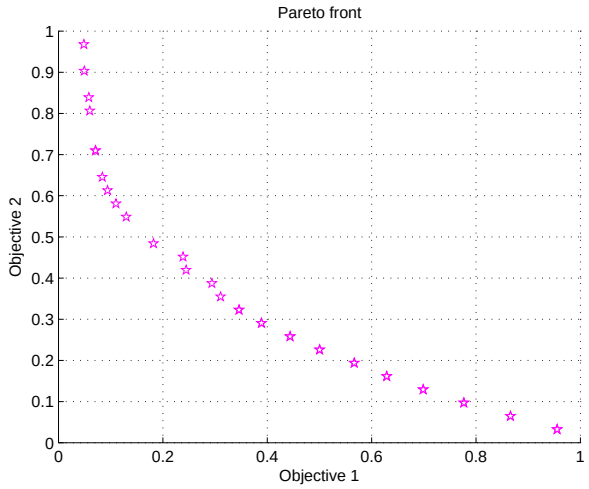


Figure 4.6: Pareto set of solutions for the synthetic graph MOOE problem

A Computer Vision Case-Study: Support Vector Machine (SVM)

In this Section, a SVM is used as a case study to evaluate the *health metric based multi-objective online evolution* scheme. A hardware core of a SVM is monitored for its health status by observing the Measure of Confidence. The intuition is that an unusually low confidence measure from a SVM may indicate hardware failures. Thus, the proposed online evolution mechanism architecturally adapts the SVM core to recover from failures by utilizing the health metric based feedback in the recovery loop.

SVMs are popular as supervised machine learning methods in classification problems. While the learning phase can either be carried out offline or online, the testing phase is usually desired online due to real-time requirements of many applications. Thus, hardware implementation is favorable, and also to accelerate intensive computations involved. We employed LIBSVM [135] for training purpose, and thereafter the learned kernels are implemented in hardware by MAC-based PEs. Because SVMs are favorable in image detection tasks in space missions [136], we consider them as a case study to evaluate the proposed self-healing mechanism.

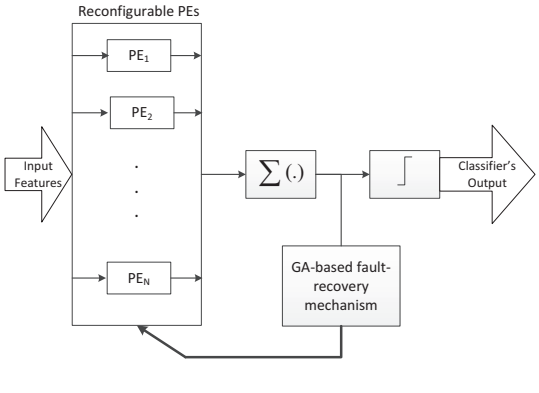


Figure 4.7: Functional block arrangement in a Self-Healing SVM case study

An architectural view of the proposed self-healing SVM is provided in Fig. 4.7. In this pattern recognition task, the SVM's measure of confidence is employed as a feedback health metric to guide the architectural adaptation through fault scenarios and power efficiency tradeoffs. As the objectives such as power consumption are secondary to minimally-acceptable throughput quality, first the proposed approach is evaluated in terms of correctness under fault-handling conditions. Fault injection and fault recovery results are listed in Table 4.3 and Table 4.4, respectively. As the results demonstrate, the measure of confidence based online evolution scheme recovers a faulty SVM classifier with only 50.24% classification accuracy to 69.12% accuracy whereas the original fault-free classifier had a 75.68% classification accuracy. Such a graceful degradation can be ac-

ceptable, or even desirable in many image pattern recognition tasks, especially when low-power and survivability objectives are to be sustained simultaneously.

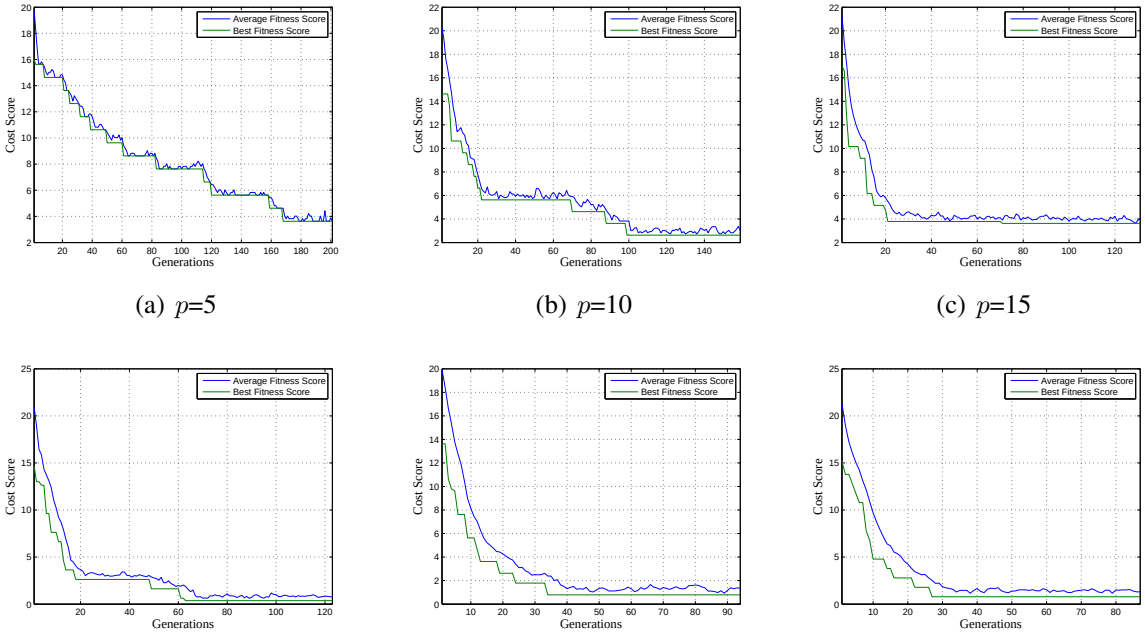
Table 4.3: Fault impact on the classifier output

	Fault-free Classifier		Faulty Classifier		
Sample Number	Estimation Probability	Detector Output	Estimation Probability	Detector Output	Actual Class
1	-1.0675	False	-0.8485	False	False
2	-1.0645	False	-0.8472	False	False
3	-1.0019	False	-0.7211	False	False
4	-0.8932	False	-0.6180	False	True
5	-1.0126	False	-0.7939	False	False

Table 4.4: Fault recovery for *Covertype*[4] dataset

Number of Faulty PEs	*Faulty Classifier's Accuracy*	*Recovered Classifier's Accuracy*
1	69.12%	75.19%
2	59.09%	73.24%
3	58.09%	73.02%
4	52.26%	72.83%
5	50.24%	69.12%

Fig. 4.8 illustrates the effect of population sizes on convergence of a single objective GA. A large population size is advantageous in terms of exploring the problem's search space as it is evident for population size of 30 as compared to a population size of 5 which needs far more number of generations of the GA to converge. Convergence required approximately 160 generations for population size of 5, approximately 100 generations for population size of 10, and about 30 generations for larger population sizes. However, it's worth mentioning that a large population size requires a longer duration to evaluate the individuals for the purpose of estimating their fitness behavior. Thus, a large population size may not necessarily correspond to faster convergence. Regardless of population size selected, it is important to note that only single instance of hardware resources is used; the population size represents only the number of entries in the data structure used to represent the dynamic set of mapping perumations being explored by the GA.



(a) p=5 (b) p=10 (c) p=15

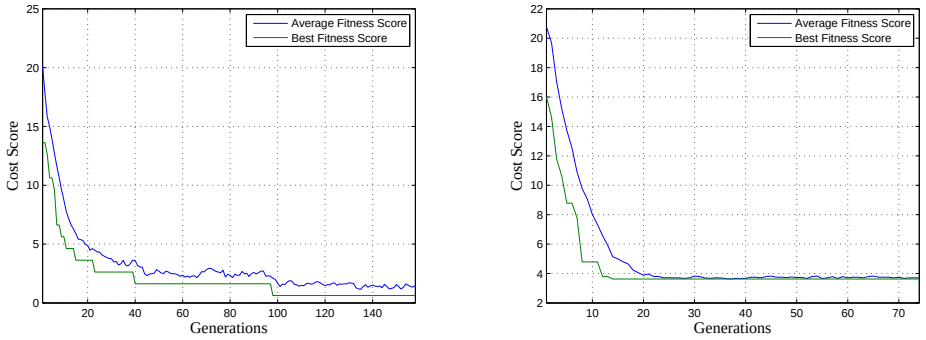
(d) p=20 (e) p=25 (f) p=30

Figure 4.8: Effect of population size on recovery results

A critical operation in GAs is crossover which combines attributes of two existing individuals in the population to create a novel individual. Fig. 4.9 illustrates the impact of the cross-over operation. In this experiment, a fixed population size of 25 is selected based on the sufficiency of that population size indicated by the previous experiment. In this experiment where 20% of the population undergoes crossover operation (i.e., $f_c = 0.2$), the cost score improves after 100 generations with elitism of the 2 best-performing individuals. Thus, the guidance function is effective at escalating the computational resources as per application needs. On the contrary, the cost score levels only after 25 generations for an exceesive crossover fraction parameter (i.e., $f_c = 0.9$). However, it is to be noted in the later case that the algorithm cannot further improve the best fitness value after generation 14, because all the individuals in the population become essentially identical. Such an overly-early convergence does not help to find the best individual in a fewer number of genera-

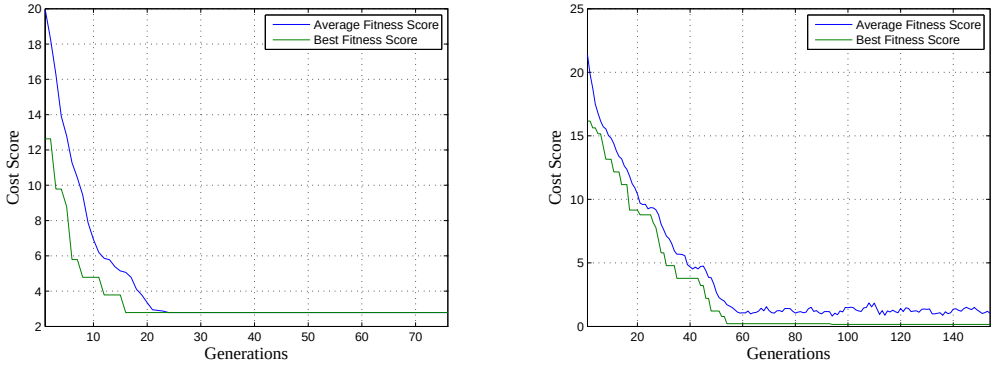
tions. Thus, this case study illustrates that the latency to converge the reconfiguration solution and the quality of the desired solution should be taken into account to determine the crossover fraction parameter in practice.

Fig. 4.10 depicts the effect of the choice of the mutation operation on the soft-resilience search progression. As Fig. 4.10(a) reveals, a mere use of crossover without any mutation improves the fitness behavior of the population to some initial level. However, at that point, a local minimum solution is reached and no further improvement is observed beyond 20 generations. Further GA operations without mutation are seen to not improve the average nor best-performing objective score. On the other hand, using a mutation operation only as in Fig. 4.10(b), the random changes applied by the algorithm exploit the diversity in solutions and hence a better solution is eventually realized, although after a larger number of generations than use of crossover and mutation together with suitable occurence probabilities.



(a) f_c=0.2 (b) f_c=0.9

Figure 4.9: Effect of crossover fraction on convergence property of the GA, p=25



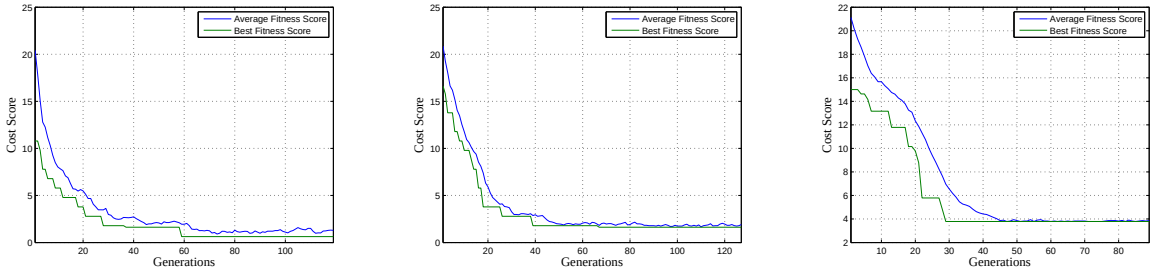
(a) With No Mutation

(b) With Only Mutation

Figure 4.10: Effect of mutation on convergence property of the GA

To analyze the effect of elitism, a fixed population size of 25 is used with a crossover fraction of 0.5 in Fig. 4.11. A lower number of elite count, such as 2, maintains the opportunity of realizing diverse individuals through the rest of the population. On the other hand, a very high number of elite count can result in slower progression towards convergence when poor average behavior occurs as those elite members become the dominating individuals and prevent more diverse exploration of the search space.

Fig. 4.12 shows the pareto set of solutions for the multi-objective evolution problem. The health metric degradation is specified in terms of degradation in measure of confidence on a normalized-to-maximum value scale. Similarly, the other objective cost to minimize, i.e., power consumption, is described on a normalized scale. For an example, if a throughput degradation of 40% is acceptable, it reduces power consumption to 30%. A further throughput degradation to an extent of 60% allows degraded operation at only 15% power consumption of the maximum power budget.



(a) n_E=2 (b) n_E=10 (c) n_E=20

Figure 4.11: Effect of elite count on convergence property of the GA, p=25, f_c=0.5

Thus, the measure of confidence results from the SVM core demonstrate the applicability of *health metric based multi-objective online evolution* approach to realize self-recovery. We investigated the effect of GA parameters on the convergence properties of evolving hardware at runtime. By carefully choosing a set of parameters, the designer can tradeoff various objective metrics such as power consumption, quality in terms of measure of confidence, throughput degradation of the SVM core during the recovery phase, latency of fault-recovery, and the reconfiguration controller overhead. Furthermore, these diverse objectives are achieved using a single cohesive strategy.

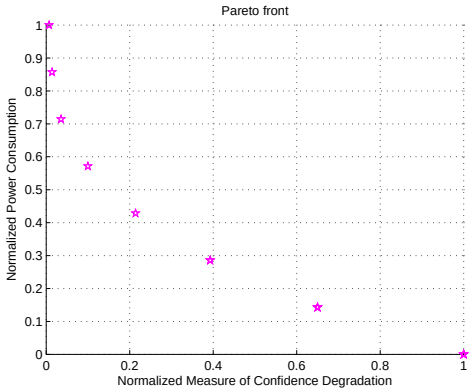


Figure 4.12: Pareto set of solutions for the SVM MOOE problem

An Image/Video Processing Case-Study: Discrete Cosine Transform

Another case-study, DCT, is used to evaluate the *health metric based multi-objective online evolution* scheme to recover from hard-faults within the DCT core. In the hardware arrangement, PSNR is employed as a health metric to guide the architectural adaptations needed for fault-mitigation. We demonstrate that PSNR based fault-detection and fault-recovery together with the proposed online multi-objective hardware evolution framework is a low-overhead technique to realize a fault-tolerant, self-healing, and low-power version of the DCT core.

To analyze the quality degradation of a faulty DCT core during the fault-handling process, the H.263 video encoder application is executed on the on-chip PowerPC processor of a Virtex-4 FPGA provided on a Xilinx ML-410 development board. The DCT module is implemented in hardware. A 256MB memory module is used to hold the executable code (`.elf` file) of the video encoder as well as providing the data memory required to hold the images. Namely, the data from the first stage of the DCT is not overwritten, rather it is kept in its own span of the frame buffer. Xilinx PlanAhead is used for Partial Reconfiguration (PR) flow while the software and hardware system is built using Xilinx Platform Studio. Various Partial Reconfiguration Regions (PRRs) are defined where each PRR corresponds to a PE of the DCT core. The Xilinx Internal Configuration Access Port (ICAP) is used for downloading the partial bitstreams from external compact flash. The Xilinx System ACE is a controller to manage configuration data. It provides an interface between CompactFlash and the FPGA. This controller is connected in slave mode over the PLB bus and the embedded processor can read the bitstreams stored on the Compact Flash. The combined ACE file consisting of full system reconfiguration file (`.bit`) and the executable file (`.elf`) can be stored on Compact Flash. The FPGA chip is configured with the stored ACE file upon a power-ON event.

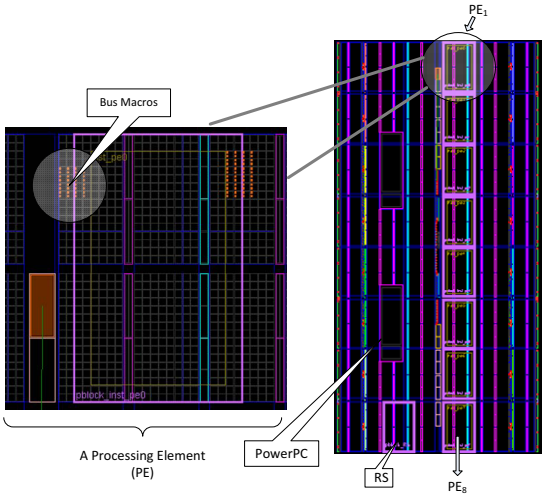
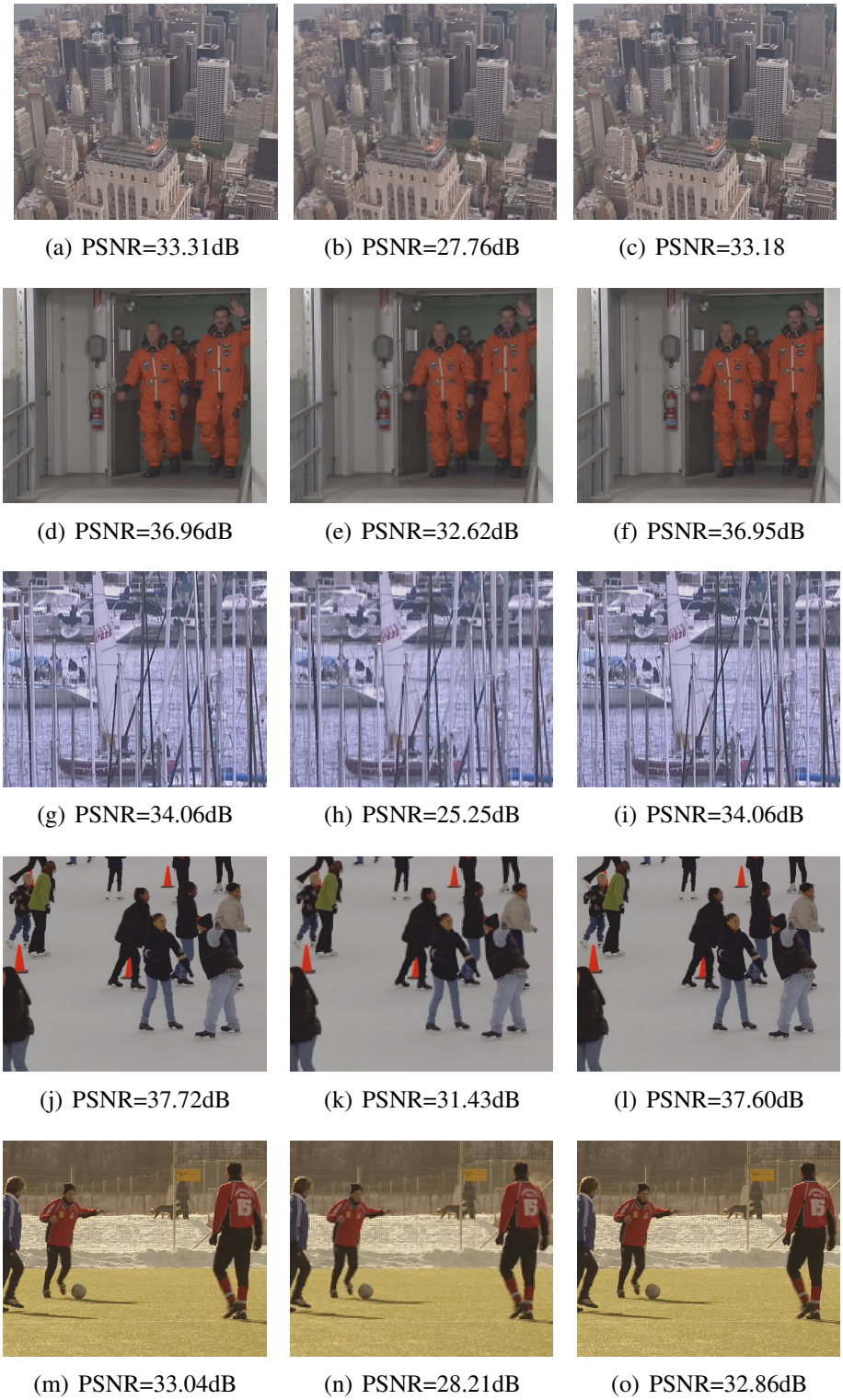


Figure 4.13: Floorplan of DCT module for Virtex-4 device

The floorplan of the DCT hardware is shown in Fig. 4.13. There are 9 reconfigurable PEs shown, each PE communicates to the static logic through the Bus Macros. The static modules of the design include PowerPC, DCT controller, Frame Buffer, Digital Clock Manager, DDR SDRAM controller, CompactFlash controller, and GPIO cores. The RS is reserved at design time to provide redundancy needed for fault-handling. Initially, the RS is configured with a blank bitstream. After fault-detection, iterative reconfiguration of the slack is performed to identify faulty PEs in the throughput datapath. If a faulty PE is identified in the datapath, the RS is configured with its functionality and introduced into the datapath thereby completing the recovery process.

In order to demonstrate fault recovery capability of the proposed MOOE resource escalating approach, throughput degradation is described in terms of PSNR-degradation. To demonstrate the energy saving capability of the proposed adaptive methodology, power consumption is reported as an evaluation metric. Fig. 4.14 shows the qualitative and quantitative results of fault-tolerant DCT module. In this evaluation scenario, no availability of a slack PE is considered, i.e, $N_s = 0$. Thus, fault recovery is realized by the sucessive re-mappings of DCT functions on the reconfigurable fab-

ric. As the PSNR results show, the proposed soft-computing framework can realize near-healthy quality objective by architectural adaptations. For example, a PSNR of 32.86dB is achieved with the power consumption of 119mW after fault-recovery when the faulty-DCT provided a PSNR of 28.21dB at 142mW power consumption. This quality recovery is reasonably comparable to the fault-free DCT's output which was 33.04dB. The reduction in power consumption becomes feasible due to the feasibility of power-gating of a least priority PE whose output was not much of a contributing factor in terms of the PSNR. In this way, the PSNR-based multi-objective online evolution explores the search space by the architectural re-mappings and their corresponding effect on output quality. In Fig. 4.14, left column shows images in frame buffer of 142mW healthy DCT, center column for 142mW faulty DCT, while right column for post fault-recovery 119mW DCT.



(a) PSNR=33.31dB (b) PSNR=27.76dB (c) PSNR=33.18

(d) PSNR=36.96dB (e) PSNR=32.62dB (f) PSNR=36.95dB

(g) PSNR=34.06dB (h) PSNR=25.25dB (i) PSNR=34.06dB

(j) PSNR=37.72dB (k) PSNR=31.43dB (l) PSNR=37.60dB

(m) PSNR=33.04dB (n) PSNR=28.21dB (o) PSNR=32.86dB

Figure 4.14: Fault recovery results for various 4cif test video sequences [2]

Comparison of Proposed Approach with Conventional Fault-Handling Techniques

Modular Redundancy

Comparing our technique to the conventional approaches used in the fault-tolerance domain, there are several criteria of improvement. For example, TMR will require 24 modules for 8x8 DCT computations and the fault capacity would be limited to errors in only one voting path. However, the proposed approach allows additional modules during normal operations, and can handle even the case when 6 out of 8 modules are faulty. Thus, compared to the TMR scheme, the area and power requirements are about one third, yet fault tolerance is improved. Moreover, fault-handling can be adjusted by the DSP circuit designer based upon the tradeoff desired between detection latency and the area overhead incurred. In addition to fault-capacity, TMR power consumption is significantly higher. On the other hand, the proposed health metric based multi-objective online evolution strategy achieves power and quality objective at uniplex area cost and significantly reduced power consumption especially for the majority proportion of the mission lifetime which is fault-free.

BIST-based Evaluation

An exhaustive test vector strategy would require 2^{96} vectors (8 values of 12 bit precision) to exercise all the logic inside a module computing a DCT function, which is computationally intractable. However, the proposed scheme evaluates the modules subjected to their actual inputs. Given the contained faulty resources do not interfere with the desired functionality, a PE can be continued to be deployed in the circuit. In the DCT core, each PE spans one Partial Reconfiguration Region (PRR) and each PRR consists of 1152 LUTs. In addition, there are other resources like FF, BRAM and DSP48 blocks. In a BIST-based resource testing scheme [89], these resources need to be tested

exhaustively, at all times even before a fault occurrence. This affects throughput as well as power consumption. However, in the proposed approach, the fault isolation phase is initiated only after a fault is detected as significant. Here, the PRR is treated as a black box in terms of the contained resources to check its health. Thus, a health metric based multi-objective online evolution offers a promising soft-resilience technique which tackles *operationally significant* faults rather than *innocuous faults*. Meanwhile, it covers both quality and power optimization using the same cohesive strategy.

CHAPTER 5: POWER AND QUALITY-ORIENTED SOFT-RESILIENCE

An architecture proof-of-concept is developed which adapts the throughput datapath based on the anticipation of computational demand in dynamic environments is demonstrated and evaluated for a ME engine. The input signal characteristics are exploited to anticipate the time varying computational complexity as well as to instantiate Dynamic Replicas (DRs) to realize fault-resilience. The scheme employs Amorphous Processing Elements (APEs) which either perform as Active Elements (AEs) to maintain quality/throughput, serve as DRs to increase reliability levels, or hibernate passively as RS available to other tasks.

Experimental results from a hardware platform for FPGAs-based video encoding demonstrate power efficiency and fault-tolerance of the ME engine. A significant reduction in power consumption is achieved ranging from 83% for low-motion-activity scenes to 12.5% for high motion activity video scenes. The scenes motion activity is utilized to improve redundancy for the purpose of priority based diagnosis of the computing modules. In addition, a graceful degradation strategy is developed to recover from hard errors by adapting the search range of candidate motion vectors. This adaptive hardware scheme is shown to automatically demote the faulty resources in FPGA devices based on streaming performance.

Motion Estimation

The demand for low power video encoders is growing while it is also desirable that their operation is maintained on acceptable quality levels. The Motion Estimation (ME) kernel can be considered to be one of the most computationally intensive units in a video encoder system [137] such as MPEG, H.263, H.264, and HEVC. Thus, a significant amount of system's overall en-

ergy consumption can be reduced by designing a low power version of the ME core [58]. To this end, various approaches have been taken to optimize ME at algorithmic level [138], architectural level [132][139][140][141], or circuit level [142], as discussed below.

There are extensive research works on reduction in power consumption at *algorithmic levels*. Usually these methods rely on reducing the computational complexity of ME for objectives including reduction in power consumption or throughput enhancement. The computational complexity issue has been tackled by techniques focusing motion search patterns, motion starting points, or adaptive search etc. [143]. Some of the examples are dynamic search window adjustment (DSWA) [144], Three-Step Search (TSS) [145][146], Diamond Search[147], Enhanced Predictive Zonal Search (EPZS) [148], and Hierarchical search [149] algorithms. The scene's activity characteristics have also been employed to reduce the computational complexity of ME [150].

The above mentioned approaches were aimed for software implementations. However, as computational demand of video coding systems is very high, hardware implementations can be preferred especially for real-time systems [151]. Conventionally, hardware-oriented schemes usually involve fixed-regular structures to conveniently map the algorithms onto the hardware. In other words, the static nature of hardware implementations lacks the capability to incorporate time-varying computational demand which varies significantly due to input signal characteristics. The presented work combines software flexibility and hardware performance by proposing a complexity prediction algorithm which interfaces with a *dynamically reconfigurable architecture*. This co-design approach allows to adapt the hardware according to runtime conditions and input signal characteristics.

In this work, a novel *Fault-Handling Motion Estimation* (FHME) core is developed which leverages the priority of functions to guide power reduction and fault-mitigation. A unified approach to mitigate aging-induced degradations, PV effects, and radiation-induced temporary or permanent faults is developed for various objectives such as survivability, power efficiency, and availabil-

ity. A resource allocation scheme is developed to reconfigure the parallelized architecture of the ME at runtime. The concept of directed management of computational resources to achieve *fault-resiliency* and *energy efficiency* is illustrated in Figure 6.1. Figure 6.1 conceptually depicts the demand and response flow of the component bitstreams. An Amorphous Processing Element (APE) can be configured as either Active Element (AE) to serve as a processing unit of the ME core, Dynamic Replica (DR) to concurrently check an AE, or RS to reduce energy consumption when processing low-activity video scenes.

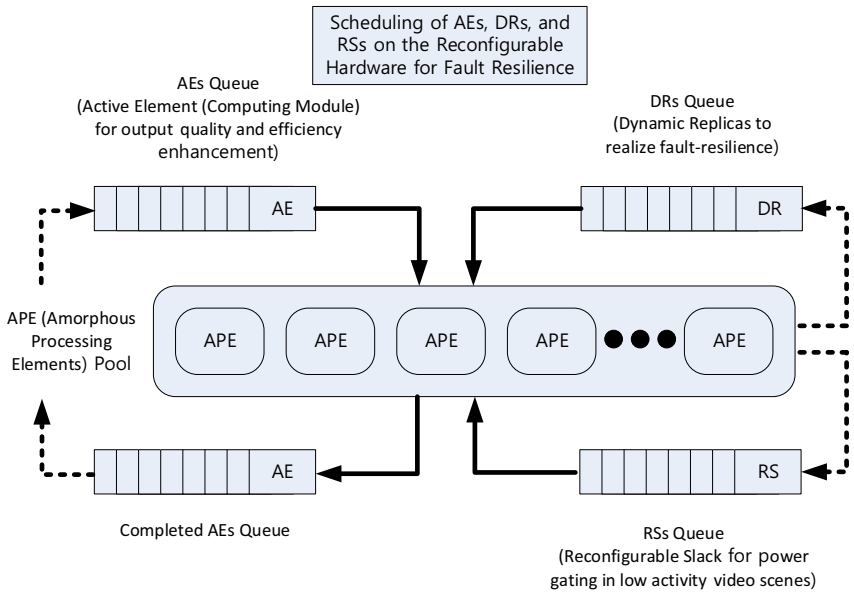


Figure 5.1: Flexible configuration of Amorphous Processing Elements (APEs)

The following are the main contributions of the presented work:

- A computational resource prediction algorithm combined with a reconfigurable architecture is proposed. The scene's motion activity runtime knowledge is incorporated to utilize/vacate the computational fabric from computations. This framework allows a significant reduction in average power consumption especially for low-activity input video.

- A fault-handling flow is proposed to tackle PV and permanent faults due to radiations effects in a unified manner. In terms of the throughput, the system remains partially online during the diagnosis process.

- During fault-diagnosis, input signal characteristics are exploited to create some dynamic redundancy. Thus, the provision of an online repair mechanism is realized with small area-overhead. In fact, during fault-free normal operation, the area requirement is uniplex. Although not necessary, the proposed scheme also allows utilization of some back-up units depending upon their availability to further enhance reliability levels.

- FHME employs a graceful degradation strategy when full recovery becomes an unfeasible option. For a given power-budget, the FHME design is shown to outperform a baseline design in terms of the compression efficiency of the video encoder. Meanwhile, an increase in average bitrate of encoder's output is very small compared to the baseline design.

Previous Techniques of Low Power ME

Power efficiency of the ME engine of a video encoder has been achieved by numerous techniques ranging from algorithm level optimizations to architecture level modifications. As discussed in the previous section, algorithm level techniques typically reduce computational complexity thereby decreasing the required number of computations per seconds and possibly the number of memory references which would otherwise incur power overhead. While such optimizations with some quality tradeoffs are beneficial in terms of architectural independence, they lack exploiting some architecture level knowledge which is necessary for efficient management of resources. That is because these algorithms are targeted for software implementations intended for those hardware platforms that are general purpose machines. On the other hand, architecture level enhancements usually focus a specific ME algorithm and try to eliminate or reduce some redundant compu-

tational units. For example, by computing an approximation initially, the search space can be pruned. For this purpose, as long as the approximator circuit has less area overhead, a reduction in average power can be achieved [139]. Recently, reconfigurable hardware based hardware-software co-design approaches [152][153][154] have been proposed to realize scalable video coding. Run-time reconfigurable architectures are of current interest such as Reconfigurable Bit-plane Matching [155], [156], Reconfigurable Systolic PE Array [153], and SoC recongurable architecture for multi-standard video compression [157]. Thus, it's desirable to combine algorithm information with underlying architectural behavior to dynamically adapt the system according to runtime conditions. Although, as a case study, we present a full-search based ME core, our intention is to develop a framework for activity-based resource allocation without constraining to any specific estimation algorithm.

The inherent robustness of ME to a certain extent has been identified previously. However, the modern trend of integrating a large number of computational modules into a System-on-Chip (SoC) design justifies the need to incorporate design for testability in video systems [158]. BIST-based approaches have been adopted to improve the manufacturing yield of video coding systems [159]. The impact of manufacturing PVs can be masked via variation tolerant design of ME exploiting some algorithmic properties [160]. A variety of approaches for fault-resilience of ME architectures have been proposed in literature [161].

Activity Based Resource Allocation Framework

Block-based motion estimation algorithms involve searching for a block in a reference frame that most closely matches with the MB in a current frame. Sum of Absolute Difference (SAD) is widely used as a matching metric to compare the MB of current frame with that of reference frame. Typically, instead of evaluating all block positions in a reference frame, a search range S is defined and

the search is performed within that search window $[-S, S]$. To pipeline the computation, the data corresponding to various sub-regions within a search window can be assigned to multiple APEs. As shown in Figure 5.2, AE_1 computes the matching metric between current frame's MB and an MB located at the same location in the reference frame. AE_2 computes the matching metric for reference frame's data located at displacements -1 and +1 pixels with respect to current MB's location, and so on. Overall, an N_a number of AEs operate for the search window data to compute a Motion Vector (MV). Thus, each APE is comprised of n PEs where $n \in \{1, 2, 4, 8, ...\}$. Since the number of APEs in the active datapath directly corresponds to the search window defined, N_a can be reduced for low-motion-activity video input in which a large search range is not required. Figure 5.2 illustrates computation of MV spatially along j-axis in a reference frame's search window with $n = 2$

FHME exploits the time varying nature of input video frames to the ME to adapt the underlying hardware resources. We consider a data parallel architecture of ME in which various APEs concurrently operate on sub-regions of input video frames. We will discuss the implementation details of such an architecture in next sections. However, it is worth mentioning here that the resource allocation framework developed here employs reconfigurable APEs. Specifically, each APE can be reconfigured to operate on the dataset of any other APE. In addition, an APE can be power gated for reducing the power consumption of the ME core.

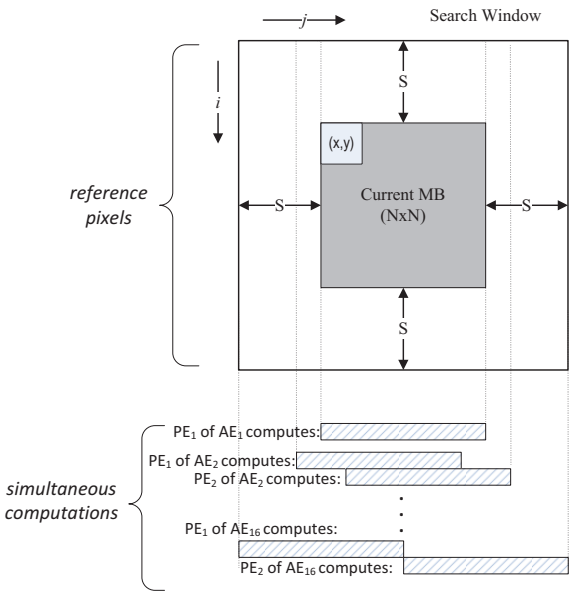
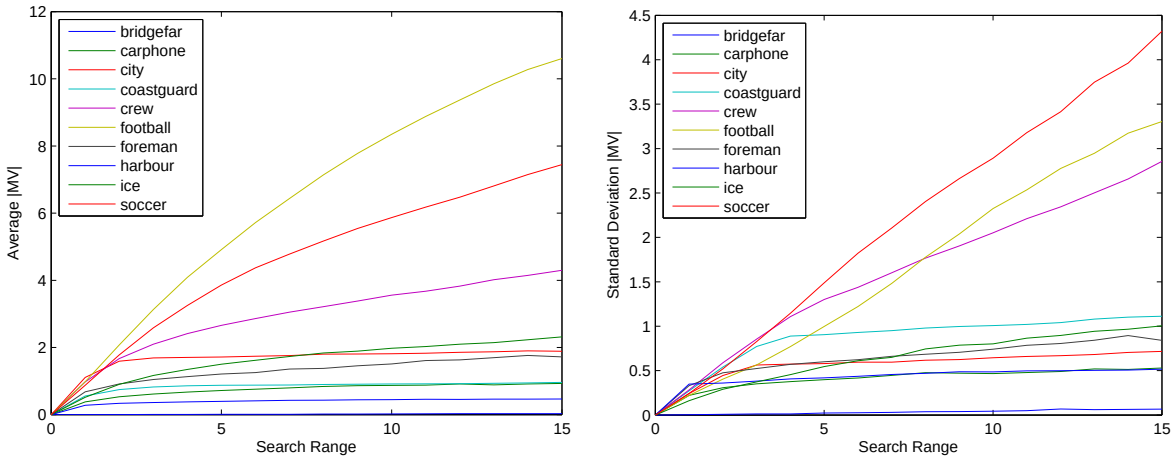


Figure 5.2: Computation of a motion vector

Runtime analysis of time varying characteristics of input data is beneficial in predicting the computational complexity and hence the required hardware resources. Hardware parallelism combined with software flexibility provides architectural support to deal with these time-varying computing workloads. Section 5 provides an algorithm to dynamically anticipate computational demand based upon scene's activity. In addition, the hardware resources released by the computational demand prediction scheme are utilized to provide the capability needed for fault-diagnosis once fault-handling scheme is triggered. We described the proposed fault-handling methodology in Section 5. The evaluation results show that fault isolation can be improved by taking into account the input signal characteristics.

To analyze the impact of motion activity in a video scene on the magnitude of MVs computed by ME and their fault effects, various video sequences [3] are in assessed in formats ranging from `.qcif` in Figures 3, 4, and 5 through `.4cif` in Figure 14. Figure 5.3 shows the average and standard deviation of magnitude of MVs for video sequences with varying activity levels. Although, high-motion-activity sequences (e.g., football and soccer) effectively utilize the maximum search range as evident by the magnitude of the MVs, yet the utilization is very poor in the case of low-motion-activity video sequences. For example, the average MV's magnitude in `foreman` is only about 2 pixels even though the search range was set to $S = 15$. When the search range is decreased from $S = 15$ to $S = 10$, its impact on the MV's magnitude is not significant even for a medium activity sequence `crew`. On the other hand, its impact on the magnitude of MVs is considerably increased in the case of high-motion-activity sequences.



(a) Average of MVs' magnitude

(b) Standard deviation of MVs' magnitude

Figure 5.3: Effect of search range on motion vector's values for various video sequences

The above discussion implies that the search range can be safely reduced for some video sequences.

Since the search range parameter is directly related to the number of active computational units (i.e., N_{AE}), energy savings can be achieved by power gating some AEs. However, an aggressive reduction of search range also impacts the compression efficiency of a video encoder. The average bitrate of a compressed video bitstream from the encoder's output increases considerably for high-motion-activity sequences when the search range is reduced from $S = 15$ down to $S = 0$. This is because a too small search range becomes insufficient to identify best matching MBs and SAD error increases consequently. Motion prediction errors increase the SAD values and in turn this impacts the energy compaction capability of the DCT block in a video encoder loop. Thus, the entropy coding compression efficiency is reduced, and hence the output bitrate of the encoder increases. The impact of SAD magnitude on bitrate is illustrated via Figure 5.4.

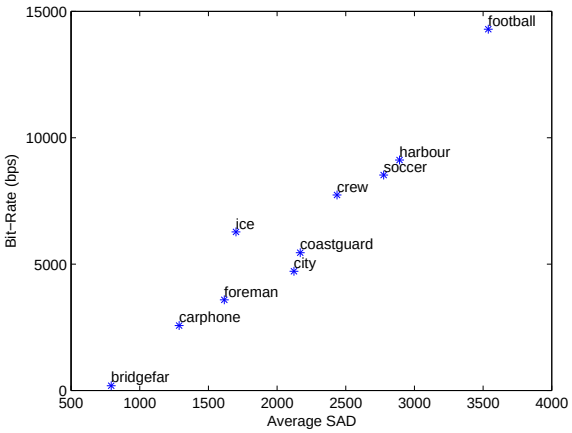


Figure 5.4: Effect of ME's SAD error on encoder's bitrate, $QP = 10$

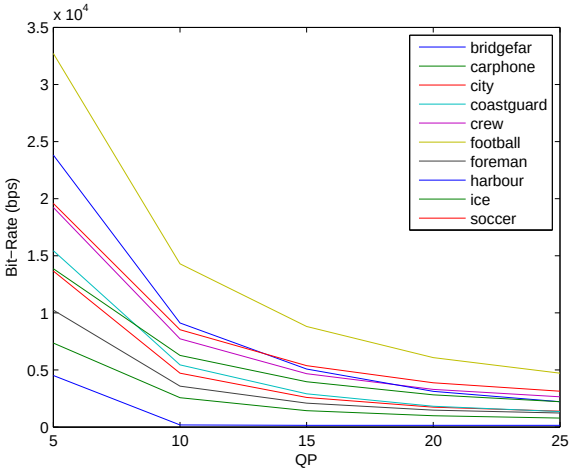


Figure 5.5: The effect of QP on bitrate, $S = 15$

The compression ratio of a video encoder also depends upon Quantization Parameter (QP) value which is user selectable. A higher QP value implies higher intended compression ratio; thus a reduced bitrate compressed video stream can be achieved although with some PSNR degradation. As shown in Figure 5.5, an increase in QP is more effective in high-motion-activity scenes than that in low-motion-activity sequences.

Next, we describe the algorithm for search range and hence computational demand prediction at run-time. Algorithm `resource_predict` is used to predict the computational resources based upon MVs and SAD values and should always be activated. For initialization purposes, all of the available APEs are utilized in the datapath. i.e., $S \leftarrow \frac{n(N-1)}{2}$. The search range is reduced for low-motion-activity scenes when we observe smaller MVs. On the other hand, this reduction results in increase SAD, and hence reduced compression efficiency for high motion activity videos. The search range is adapted to accommodate larger MVs in that case. Consequently, a threshold τ_{SAD} is defined and the change in minimum SAD error (Δ_{SAD}) is compared to this threshold to check if S should be either increased or decreased. The computational resource prediction is performed at a temporal resolution of 5 frames. We take into account standard deviation of MVs, bitrates for

various search range values, QP and reconfiguration time T_r to empirically choose the threshold to $\tau_{SAD} = 100$ and temporal window duration. After applying resource prediction algorithm, some APEs released from the ME engine are available for fault-handling purposes.

Algorithm resource_predict: Dynamic computational resource prediction based on motion activity

Require: n, μ, S, SAD
Ensure: N_{AE}
1: $\hat{S} \leftarrow round(1.3 * \mu)$. Predict search range according to motion vectors. A value 1.3 is selected based upon the average and standard deviation of MVs' magnitudes in benchmarks used.
2: **if** $(\hat{S} < S)$. search range can be reduced for power saving **then**
3: $S \leftarrow \hat{S}$. Adapt the search range according to the prediction
4: **else**
5: $\hat{S} \leftarrow S$. Keep the same search range
6: **end if**
7: $\Delta_{SAD} = SAD - SAD_{prev}$
8: **if** $(\Delta_{SAD} > \tau_{SAD})$ **then**
9: $\hat{S} \leftarrow S + n$. Increase the search range
10: **end if**
11: **if** $(\hat{S} > \frac{n(N-1)}{2})$. If the predicted search range cannot be attained using available APEs **then**
12: $\hat{S} \leftarrow \frac{n(N-1)}{2}$. Utilize all the available APEs in the datapath
13: **else**
14: **if** $(\hat{S} = 0)$ **then**
15: $\hat{S} \leftarrow 1$. lower threshold on search range
16: **end if**
17: **end if**
18: $SAD_{prev} \leftarrow SAD$
19: $N_{AE} \leftarrow \frac{2\hat{S}}{n} + 1$. Compute the computational demand of predicted search range

Faults Mitigation Strategy

As discussed in the previous section, multiple APEs are allocated on the computational fabric, where each APE is used for n SAD computations per row corresponding to a sub-region in reference frame's search area. In addition, some empty APEs are reserved at design time to provide reserve for fault recovery. In absence of such spares due to area or power constraints, some low priority AEs can be vacated to perform more prioritized computations as discussed here.

In video encoders, PSNR is possibly maintained by fixing the QP while allowing the bitrate of the encoded bitstream output to vary. Our choice of the ME architecture is based upon the intuition that defectiveness of some of the APEs can be compensated by a reduction in the search range. Such

an approach is promising in terms of graceful degradation when confronting faults and mitigating their effects. A decrease in the search range results in reduced coding efficiency as demonstrated by bitrate increase in Figure 5.6. On the other hand, its impact on the PSNR measure and hence visual quality of the images is imperceptible as shown by PSNR curve in Figure 5.6. Hence, the bitrate can be used as a health metric to detect hardware faults in the ME engine. After fault-detection, the scheme proceeds to identify/isolate faulty APEs, and then the fault recovery phase involves avoidance of those isolated APEs. A re-mapping of tasks from faulty AEs to healthy APEs completes the recovery process. We discuss each phase of fault-handling below.

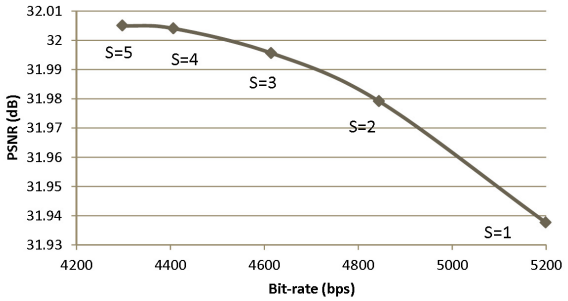
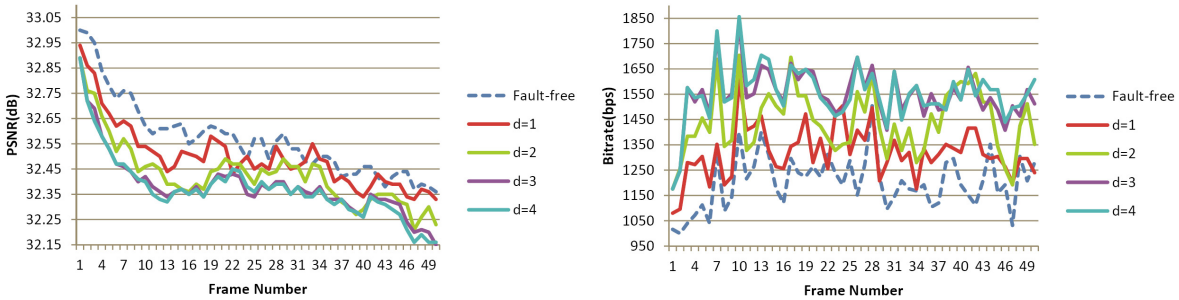


Figure 5.6: RD curve showing the effect of increasing search range



(a) PSNR for various number of defective APEs

(b) Bitrate for various number of defective APEs

Figure 5.7: Fault injection results for container video sequence

In the experiments to evaluate the proposed detection, isolation, and recovery, variable bitrate mode is selected for video encoder. In addition to scene's high-motion-activity, a failure in ME process-

ing due to hardware faults can also be causal in increasing the bitrate of encoded video stream, thereby degrading overall compression efficiency. The PSNR and bitrate of encoded bitstream for the `container` input video sequence for various fault-scenarios are shown in Figure 5.7(a) and Figure 5.7(b), respectively, in which 'd' corresponds to number of faulty APEs.

Detection of Hardware Faults

As we discussed, the bitrate should correspond to the MV values for fault-free operation. For example, a small MV with a larger enough search range, yet producing large bitrate compressed video stream may imply potential hardware errors. Faults in ME core result in incorrect computation of MVs and hence the SAD increases. To adaptively allocate the resources for power efficiency and fault-mitigation, we monitor the MVs as well as the bitrate for a given search size and QP. When the bitrate of the compressed bitstream from the encoder increases, we evaluate the following two possible scenarios:

1. For the variable bitrate mode (i.e., fixed QP), prediction for P frame is not working quite well due to the high motion activity in the scene, or

2. Hardware faults on ME occurred.

Since we do not have an a-prior knowledge about the cause, we first assume that scenario-1 occurred. Therefore, more APEs are assigned as per Algorithm `resource_predict` to increase the search window, and the bitrate change is observed by monitoring the output buffer occupancy of the encoder as well. If the change in minimum SAD error is significant as a result of search range increase, then the fault-handling mechanism is not triggered. However, if a widening of search

range still does not correlate with the changes in SAD error, then the proposed fault-handling scheme is triggered to detect and isolate any faulty modules. It is worth mentioning here that the proposed fault-handling scheme runs at a higher layer than the computational layer and does not impact the throughput datapath. The proposed reconfiguration scheme does not contribute to the critical path which would otherwise degrade the performance of the ME core. An additional benefit of keeping it outside the datapath is that the FHME scheme does not impact the functionality of the ME for those video sequences which are outliers to the threshold selected.

Fault Diagnosis using Dynamic Redundancy

After fault-detection, the next step is to perform CED at the APE level so that faulty APEs are identified by a pairwise comparison diagnosis procedure. Such a functional testing approach does not require the CUT to be brought offline for input test vector evaluation. Thus, the CUT sustains partially useful throughput even during the diagnosis phase.

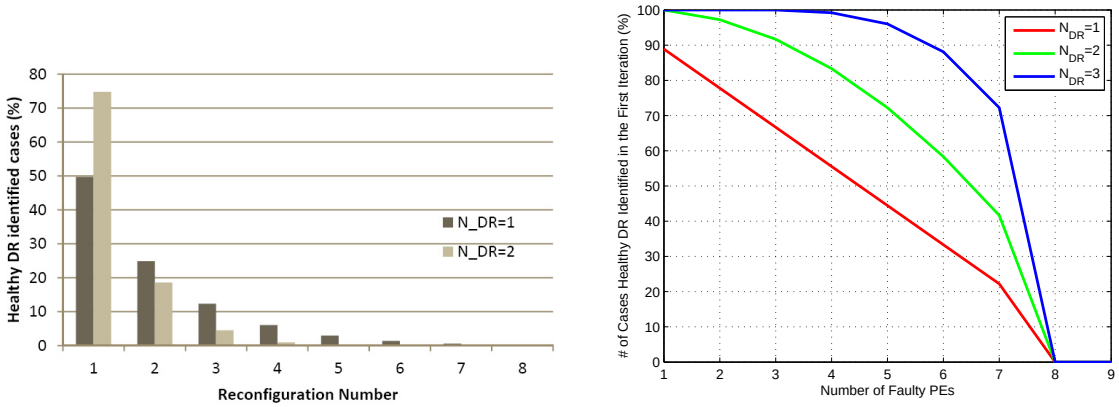
Phase-1-Identifying a healthy APE: We assign different priorities to each APE according to the relative importance of input data which they process. For example, the APE used to search the locations near the predicted MV point in inter-frame mode has higher priority, while the APE to search the unlikely locations is assigned a lower priority. Initially, some low priority APEs are configured as DRs to serve as checkers for those APEs which are in the throughput datapath. Such a temporary replacement operation of least priority APEs minimally impacts the output's bitrate as shown in Figure 5.3. Alternatively, the availability of some healthy RSs obviates the need even to temporarily vacate the least priority APEs for diagnostic purposes. Nonetheless, irrespective of availability or unavailability of healthy RSs, the diagnosis Algorithm `FHME` proceeds to identify a healthy APE in the resource pool. It illustrates how dynamic redundancy is employed in fault-diagnosis process to check the APEs in the throughput datapath.

Algorithm FHME: Comparison-based diagnosis to identify at least a single healthy APE

Require: Current Search Range (S), Number of slacks available for ME (N_{RS}), Number of dynamic replicas to employ in the diagnosis procedure (N_{DR})
Ensure: Identified set of healthy APEs, V_h

Initialization: $V_h = \varnothing, TCAE = 1$. Current testing candidate APE
if $N_{DR} < N_{RS}$ **then**
    $N_{RS} \leftarrow N_{RS} - N_{DR}$. Remove from slack
else
    $S \leftarrow S - \frac{n(N_{DR}-N_{RS})}{2}$
    $N_{AE} \leftarrow \frac{2S}{n} + 1$. Reduce the search range
end if
while $((V_h = \varnothing) \wedge (V_T \neq \varnothing))$ **do**
    $(APE_j.function \leftarrow APE_{TCAE}.function)\ \forall_j\ where\ N_{AE} < j \leq (N_{AE} + N_{DR})$. Configure N_{DR} APEs as DRs to check the APE_{TCAE}, .e.g., check the zero vector APE first
    $V_T \leftarrow \{APE_{TCAE}, APE_j\} \forall_j N_{AE} < j \leq N_{AE} + N_{DR}$. Form a pool under test
    $v = majority(V_T.output)$. Perform majority voting of the pool under test
    if $APE_i.output = v$ **then**
        $\phi_i \leftarrow 0, V_h \leftarrow APE_i$
    else
        $\phi_i \leftarrow x; \forall_{i \in V_T}$. Estimate the health status of the pool under test
    end if
    if $TCAE < N_{AE}$ **then**
        $TCAE \leftarrow TCAE + 1$. Update the test candidate APE
    else
        if $(N_{AE} - N_{DR}) \geq 1$ **then**
            $N_{AE} \leftarrow N_{AE} - N_{DR}$. Configure the AEs as DRs
        else
            if $N_{AE} > 1$ **then**
                $N_{DR} \leftarrow N_{AE} - 1$
                $Na \leftarrow 1$
            else
                $N_{AE} \leftarrow 0, V_T \leftarrow \varnothing$
            end if
        end if
        $TCAE = 1$. Re-check the first APE with a different DR
    end if
end while

Figure 5.8 shows the number of group reconfigurations required to isolate the faulty modules. For example, if the current $N_{AE} = 8$ and one RS is available to serve as a DR for diagnosis, approximately 50% of the fault scenarios are successfully isolated in the first iteration. As there are 9 modules in total, there are 512 possible combinations of the faulty/healthy modules. Out of these 512 cases, about half of them require only one iteration to find a healthy DR. It is clear from Figure 5.8(a) that the fault isolation process is accelerated by increasing the number of DRs. Figure 5.8(b) shows the probability of isolating the faulty modules in the first iteration for different number of DRs. As compared to the case when using a single DR, more faulty scenarios are resolved using a pair of DRs.



(a) Number of reconfigurations required

(b) Probability of finding a Healthy DR in the first iteration

Figure 5.8: The effect of N_f on iterations required for the fault-diagnosis algorithm

Phase-2-Isolation of faulty APEs: After at least a single healthy APE is identified, it is used to check the health state of all other APEs in the overall resource pool. In case of multiple healthy identified APEs, the fault-isolation process can be accelerated by concurrently checking the output discrepancy of multiple AEs. For an identified healthy APE$_j$, the fitness state ϕ_i of a suspect APE$_i$ is updated as follows:

$$\phi_i \leftarrow 0;$$
$$while_{i \in V_T} \{\, if\ (APE_i.output \neq APE_j.output)$$
$$then\ \ \phi_i \leftarrow 1; \quad \}$$

Next, faulty APEs are reconfigured with blank configuration bitstreams thereby eliminating any switching/processing to save power. For non-reconfigurable devices, this step can be accomplished by power gating the identified faulty APEs. Lastly, the faulty APEs are removed from the resource pool, i.e., $N \leftarrow N - d$.

Fault Recovery

In video encoders, the predicted MV which is calculated by the surrounding MBs' motion vectors becomes the search center since differential coding is used to encode the current MB's MV information to save bits. If the Coded MB Indication (COD) bit in the output bitstream is set to 1, no further information is transmitted for the given MB, meaning that it is an inter-MB with MV for the whole block equal to zero and with no coefficient data. This saves considerable bits in static scenes. In the proposed hardware architecture, SAD(0,0) computation, which corresponds to predicted MV location, is assigned to AE_1. Thus, AE_1 performs the most prioritized computational function, i.e., computing the SAD corresponding to zero MV data. AE_{Na} computes the SAD values corresponding to candidate block positions which are farthest from the current block's predicted position.

In the recovery phase, the datapath is reconfigured by selecting healthy APEs using a reconfiguration controller. In addition, AEs priority is also engaged when recovering from a fault situation. For example, to mitigate the defectiveness of faulty AE_1, its faulty resource is demoted by releasing its pre-assigned task. A healthy resource from the least priority AE is promoted to perform this important task. Finally, the APEs are re-labeled after the reconfigurations so that AEs sorted in ascending order perform computations in descending order of priorities.

Case-Study : FPGA-based Implementation of Full Search FHME

A full-search (FS) approach of ME [162] guarantees optimality by exhaustively searching for the minimum SAD metric to match the Macro Block (MB) in current frame over all possible positions of candidate MBs within a designated search area in the reference frame. In Figure 5.2, all the candidate positions within a search window are evaluated to find the best matching block position

for an MB in current frame. Here, an MB of size (NxN) in current frame located at (x, y) is evaluated for various block positions in the range $(-S, S)$ along the row and column indices, i and j, respectively in the previous frame. For instance, a search area ranging from -15 to +15 pixels requires 961 SAD computations as it can be deduced from Figure 5.2 where SAD(x,y) represents the SAD value computed for MB location (x,y) in a current frame.

This regular architecture of conventional FS style of ME facilitates reconfiguration for fault-handling and runtime resource management to reduce power consumption while output signal quality degrades gracefully. As shown in Figure 5.9, the APEs are defined with a data-parallel organization and each APE operates on a sub-region in an image. Except APE_1, each APE computes 2 SADs for a given row of pixels in the previous frame. This type of architecture maps well to the inherently regular fabric of FPGAs. FPGAs provide a reconfigurable fabric of fixed size. Utilizing some of the fabric as standby APEs does not increase static power consumption compared to the case when APEs are not defined. Any techniques that reduce standby power of the fabric remain applicable to our architecture. In addition, it may be noted that our proposed scheme is not limited to FPGAs. For ASIC implementations, it is beneficial to consider the term reconfiguration in a context where routing component is controlled so that data is multiplexed to the desired target computational modules by the Array Control Unit shown in Figure 5.9. The signal lines `din_current`, `din_reference`, and `ref_pels` denote input pixels of a current frame, input pixels of a reference frame, and pixels from reference frames buffer, respectively. Both current frame data as well as reference frame data is cached in an FPGA's BlockRAM. All elements depicted in Figure 5.9 reside in the FPGA's fabric.

Memory access is key constraint in the motion estimation architectures of video processing which have been investigated by many previous works [163] so that the total number of memory accesses is reduced by maximizing the data reuse. Once the pixel data are read from the frame memory, they are kept in the pipelined datapaths of ME to be reused row and column wise in the given

search range during the calculation of the motion vector matching criterion such as SADs. In this work, while optimized address generation scheme to reduce the accesses to the frame memory and buffering of pixel data are handled in the presence of a conventional memory interface as shown in Figure 4, we mainly focus on optimizing the use of processing elements to support fault-handling through architectural adaptation at the algorithmic level.

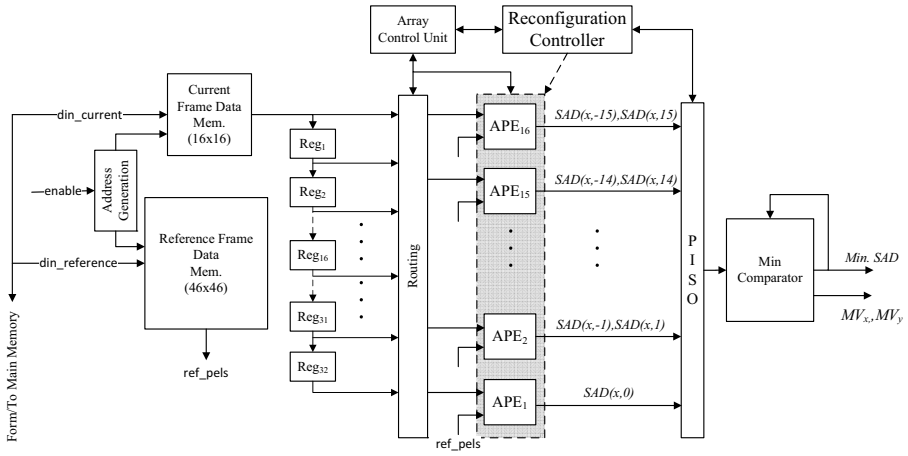


Figure 5.9: Hardware architecture of FHME

The FHME core has been described using Verilog Hardware Description Language (HDL), and then synthesized and implemented in Xilinx Virtex-4 FPGA using Xilinx Integrated Software Environment (ISE) version 14.3 design tool. PlanAhead 14.3 [164] is used for defining reconfigurable partitions (RPs) [165], mapping the reconfigurable modules (RMs) to those regions [166], and partial bitstream (`.bit`) files. Xilinx Embedded Development Kit (EDK) is used to build the overall processor-based system while the software is built in the Xilinx Software Development Kit (SDK) environment. We used Xilinx ML410 development kit which is an evaluation board for Virtex-4 devices with two on-chip PowerPC processors. The on-chip processor shown in Figure 5.10 implements software-based Reconfiguration Controller shown in Figure 5.9. It reads the output of APEs via Parallel-In-Serial-Out (PISO) buffer to diagnose the APEs by detecting the output dis-

crepancy, and subsequently performs reconfiguration of the datapath. Partial `.bit` files are stored on a compact flash and are used by the processor to activate/power-gate certain APEs by fetching them to configuration memory. Depending upon the size of the `.bit` files and the overhead of library functions, a reconfiguration of an RP takes 170 milliseconds on average. Thus, the reconfiguration time is comparable to 5 frames of video at 30 frames per second input frame rate. The on-chip processor also executes a modified software implementation of Telenor TMN H.263 video encoder's blocks except for the ME block which is performed by a hardware-implemented core.

The hardware utilizations for Virtex-4 `4vfx60ff1152-12` device in the baseline ME design and the FHME design which includes a router component to facilitate reconfiguration are listed in Table 5.1. It indicates that although FHME incurs some area overhead, the maximum operational clock frequency still remains the same as the multiplexors contained in the router component do not impact the worst path delay of the circuit. Other components not listed in Table 5.1 include compact flash for storing the bitstreams, Double Data Rate-Synchronous DRAM (DDRAM) for holding the software implemented video encoder blocks, and hardware Internal Configuration Access Port (ICAP) core for reconfiguration through the internal programming port of the FPGA. Table 5.1 also indicates that the overhead of FHME in terms of increased resources is only a few percent.

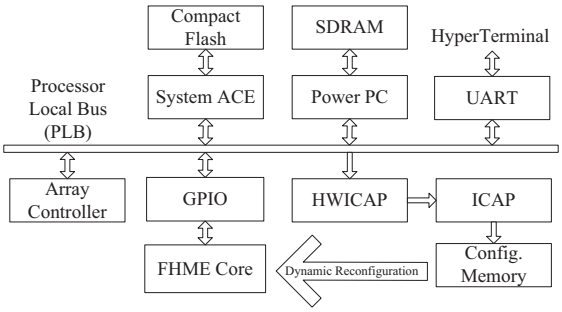


Figure 5.10: Evaluation Setup: FPGA based FHME's interface with on-chip processor

Table 5.1: Hardware utilization summary for Virtex-4 FPGA

Resource	Utilization		
	Baseline	**Using FHME**	**% increase**
Number of Slices	1484	1575	6.1
Number of Slice FFs	1374	1374	no change
Number of 4 input LUTs	2383	2639	10.7
Number of FIFO16/RAMB16s	3	3	no change
Clock frequency	150 MHz	150 MHz	no change

Evaluation Results of FHME

Energy Saving in Reconfigurable Design

Given some tolerance of bitrate variation, the number of APEs that can be vacated depends upon a scene's motion activity as illustrated by results from videos in Table 5.2. Thus, a significant number of RSs can be created dynamically for low motion activity video scenes. On the other hand, disabling the AEs in high motion activity video scenes causes an increase in bitrate. One way to examine these interacting effects is to calculate power savings of the ME architecture as the PEs are power gated. The number of inactive PEs influences the bitrate overhead which can be measured for a given video sequence. For example, Figure 5.11 shows the trend of saving between 20 mW and 120 mW as N_{RS} is varied.

Table 5.2: Number of vacated APEs while bitrate within 3% tolerance

Video sequence	**Motion activity**	Baseline ME		FHME		$\mathbf{N_{RS}}$
		PSNR (dB)	**Bitrate**	**PSNR (dB)**	**Bitrate**	
Soccer	High	32.32	8.43	32.31	8.62	1
Football	High	31.44	14.22	31.45	14.61	2
Ice	Medium	33.56	6.29	33.51	6.38	4
Suzie	Low	34.29	2.05	34.22	2.07	5

Figure 5.12(a) illustrates the effectiveness of Algorithm `resource_predict` in adapting the number of active APEs in the datapath according to the runtime conditions. As it is evident from

various frames of a video sequence, the number of predicted computational resources (i.e., N_{AE}) dynamically adapts according to the average MV value. A higher than the minimum required search range leads to power overhead while a lower search range increases the SAD values and bitrate increases consequently. Furthermore, as demonstrated via Figure 5.12(b), the bitrate increase is small implying that proposed computational resource prediction algorithm works well. Figure 5.13 shows power savings and consequently the bitrate overhead of FHME for various video sequences. The average power consumption is reduced from 320 mW to 280 mW thus saving 12.5% for a high-motion-activity video sequence (`football`) while the power savings in case of low-motion-activity sequences is significantly as high as 83% in case of `bridgefar`. Also, the bitrate overhead due to prediction errors is negligibly small as it can be seen from Figure 5.13(b) where columns for FHME and the baseline circuit are comparable.

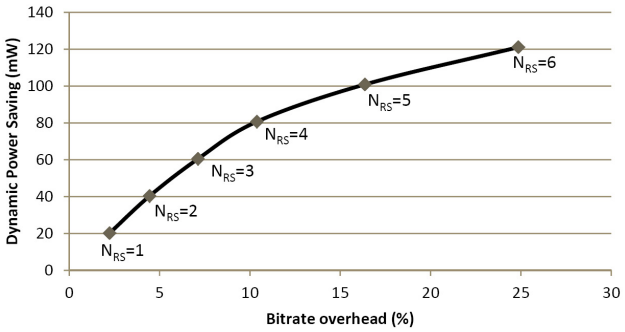
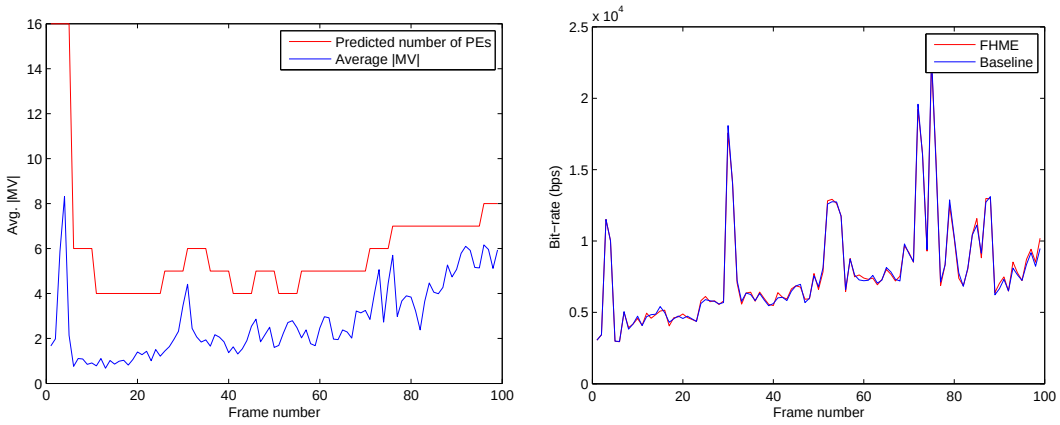


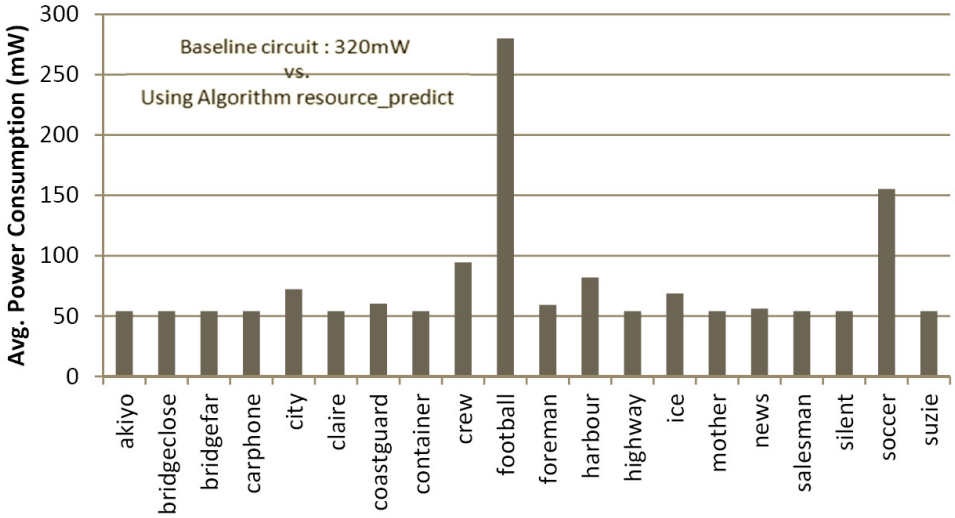
Figure 5.11: Power saving at the cost of increased bitrate for Soccer video sequence



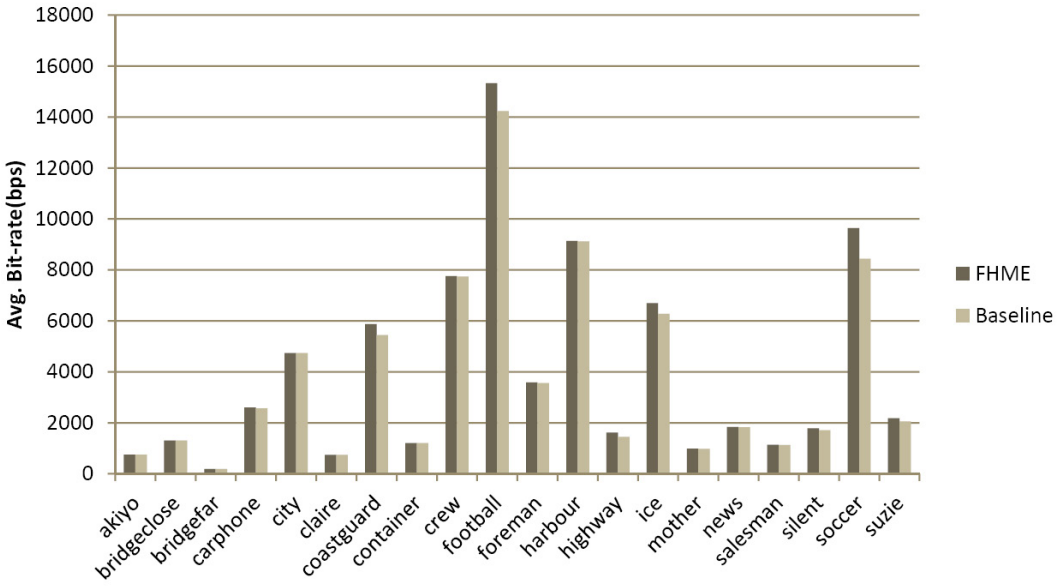
(a) Number of active APEs predicted by Algorithm `resource_predict`

(b) Bitrate variation

Figure 5.12: Dynamic computational resource prediction for `crew` video sequence



(a) Average power consumption



(b) Bitrate

Figure 5.13: Energy saving results of FHME with low overhead of bitrate

Figure 5.14 illustrates the qualitative results of a 704×576, 30fps, `city.4cif` video sequence [2]. The FHME architecture provides considerable reduction in power consumption with only a slight degradation in image quality as evident from the PSNR measure as well.

(a) Image in the baseline's frame buffer, PSNR= 33.02dB, Power=320mW

(b) Image in the FHME's frame buffer, PSNR=32.09dB, Power=120mW

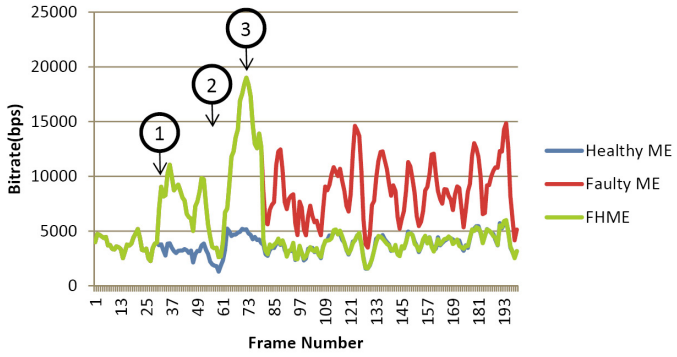
Figure 5.14: Power and quality tradeoff results for `city.4cif` video sequence

Online Recovery Results of FHME core

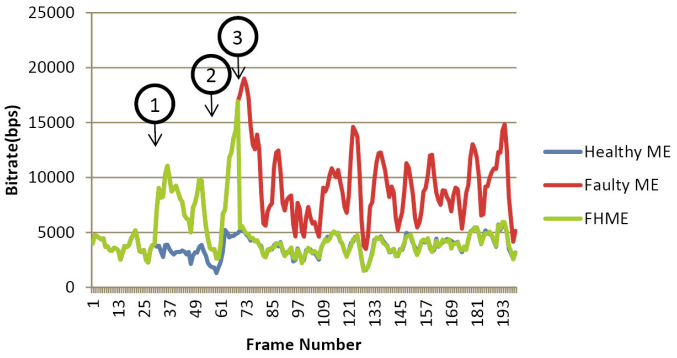
We evaluate the effectiveness of the proposed approach by simulating fault injections in the post place-and-route simulation model of the ME circuit generated by Xilinx software flow. For this purpose, Verilog HDL (`.v`) and Xilinx User Constraints File (`.ucf`) source files are modified to simulate stuck-at faults at Look-Up Tables (LUTs) inputs, FF-inputs, and routing resource. A stuck-at 'one' or 'zero' fault manifests itself in the form of some output deviation from the actual. This fault model reasonably simulates Local Permanent Damage, Single Event Latchup, Electromigration, and Localized Aging due to high switching activity.

Figure 5.15(a) illustrates an operational example from a video encoder containing a FHME core for `foreman` video sequence. For this simulation, random faults are injected in AE_1 at frame

30 which is highlited in Figure 5.15(a) as Event ①. Without any fault information available, Algorithm `resource_predict` adjusts the search range; however, bitrate does not get improved because of the faulty AE. Fault detection is triggered at the instant shown by Event ②. Then, Algorithm `FHME` is trigged for diagnosis purposes in which evaluation period of 10 frames is selected for discrepancy check. The reconfiguration of a healthy module from the RS set to AE set completes the recovery process at instant labeled by Event ③. Figure 5.15(b) shows another scenario in which two DRs are employed for fault-diagnosis purposes, and hence, fault-isolation latency is improved from a latency of 51 frames down to 40 frames. Table 5.3 illustrates the average reduction in compression efficiency in terms of bitrate increase to compare fault-free, baseline, and FHME modules. This shows a significant reduction in bitrate overhead from 117% to 24% depending upon the number of DRs available.



(a) using a single DR



(b) using a pair of DRs

Figure 5.15: An example of online fault-handling

Table 5.3: Bitrate of encoded bitstream for foreman video sequence

Condition	Average bitrate (bps)	Average increase (%) in bitrate (%)
Fault-free ME	3755	0.0% (Ref.)
Faulty Baseline ME	8166	117.4%
FHME with a single DR	5246	39.7%
FHME with a pair of DRs	4678	24.6%

CHAPTER 6: HEALTH METRIC BASED DYNAMIC RESOURCE ALLOCATION

Reconfigurable hardware fabrics have been widely used as platforms for signal processing applications. The customizable datapath in FPGAs can be very beneficial for accelerating time demanding tasks, such as image/video coding applications, cryptographic algorithms, and speech processing [82]. While erroneous data over a noisy communication channel is usually detected on the receiver side by employing Error Correcting Code (ECC)-based methods, powerful *consistency properties* of data transmission have been exploited by using signal processing techniques by various researchers. For instance, a relevant work in which an SNR measure is employed for soft-error resiliency has been applied to an FIR filter with objective of achieving reduced energy operation [55]. For video encoders, PSNR and sum-of-absolute-difference measures are used to characterize error-resilient architecture of motion estimation kernel [58]. Another technique to detect errors is by replicating the system to realize a CED pair. The scheme in [167] proposes an additional decoder to realize an inverse-comparison CED for a data-compressor. The data from the output of an encoder is reconstructed to match with original source input to detect errors in the encoder. In contrast to a CED scheme, FaDReS [45] and PURE [168] avoid the duplication of functional blocks while errors are detected in simplex mode of operation, thereby reducing logic resources and power.

Fault-Handling Method

A system-level block diagram is shown in Fig. 6.1 which identifies the roles of the Reconfigurable Logic Fabric and On-Chip Processor Core of a typical FPGA device. Within the Reconfigurable Logic Fabric, the desired processing function such as a DCT or Advanced Encryption

Standard (AES) core, is realized by the PEs which comprise a processing array. These PEs are reconfigurable at runtime in two ways. First, they can be assigned alternative functions. Functional assignment is performed to leverage priority inherent in the computation to mitigate performance-impacting phenomena such as Extrinsic Fault Sources, Aging-induced Degradations, or manufacturing Process Variations. Second, the input data can be re-routed among PEs as necessary by the PURE Reconfiguration Controller. These reconfigurations are only initiated periodically, for example when adverse events such as aging-induced failures occur based on perturbations to the health metric. The functional re-mapping is performed by fetching alternative partial Configuration Bitstreams for the PEs which are stored in a Compact Flash external memory device. A Configuration Port, such as the ICAP on Xilinx FPGAs, provides an interface for the Reconfiguration Controller to instantiate the PEs with the bitstreams used to perform computational functions in the processing array. The input data used by the PEs, such as input video frames, resides in a DRAM Data Memory that is also accessible to the On-chip Processor Core. Together these components support the data and reconfiguration flows needed to realize a run-time adaptive approach to resilient architectures.

Relaxing the requirement of test vectors for fault-detection can realize a significant reduction in the testing overhead of previous approaches. To realize fault-diagnosis and recovery, PURE utilizes runtime reconfigurability by considering priorities in the underlying computation. To re-assign the function executed by an identified faulty PE, either a design-time spare is engaged into the active path, or some least-priority PE is utilized by multiplexing the input-output data. Functional reassignment is realized by fetching its function-to-PRR mapping configuration bit file from external memory into the FPGA configuration logic memory. In addition, the faulty PE is configured with a blank bitstream to cease switching activity which otherwise would incur additional power consumption.

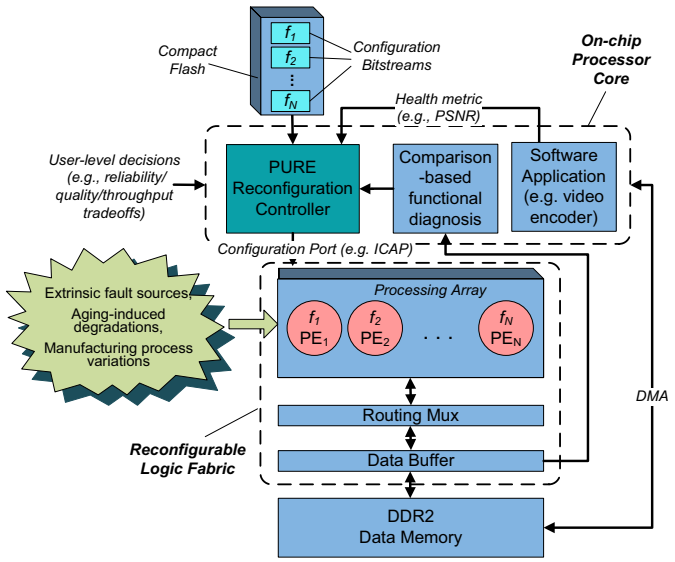


Figure 6.1: Self-adapting resource escalation of the FPGA device

In a broad sense, provision of resilience in reconfigurable architectures for signal processing can take advantage of a shift from a conventional precisely-valued computing model towards a significance-driven approximate computing model [111],[112],[113]. This significance-driven model provides inherent support for a continuum of operational performance which is compatible with the concepts of signal quality and noise. In this way, PURE recasts the reliability issues of contemporary nanoscale logic devices in terms of the significance associated with these computations.

Similar to previous approaches, the techniques developed herein progress through explicit fault-handling stages of *fault detection*, *fault-diagnosis*, and *fault recovery*. Fault-detection can be either performed by continuously observing a system health metric like Signal-to-Noise Ratio (SNR), or checking the processing nodes in an iterative fashion, as will be discussed in Section 7.2. For example, in the case of a video encoder, the PSNR of a video sequence provides a health metric for a uniplex arrangement without redundancy. On the other hand, in absence of a uniplex health metric

such as PSNR, the designer can tradeoff the use of periodic temporal CED [169] [7] or spatial CED [30] redundant computations based on throughput, cost, and reliability constraints. To illustrate the details of operation under each phase of fault-handling, two case studies are developed: a DCT core in a video encoder and a 128-bit AES core, using PSNR-based and discrepancy-based CED health metrics, respectively.

In general, the process of identifying faulty nodes in a system $\mathbf{G}$ is called *Fault Diagnosis*. The maximum number of faulty nodes which a scheme guarantees to identify is known as *diagnosability* of $\mathbf{G}$. Consider a fully connected topology so that the diagnosis can be performed between any pair of nodes. Then, after identifying a faulty node, it can be replaced by any of the available healthy nodes. Hence in this work, the term node applies to both PE and RS regions. The overall objectives are to maintain the throughput during the diagnosis phase and rapidly identifying the faulty PEs.

Fig. 6.2 illustrates the scope, approaches, and metrics of this dissertation. While the fault detection phase is discussed later, a diagnosability formulation for identifying faulty nodes is developed in Section 4 using a syndrome function. The three diagnosis algorithms of a divide-and-conquer approach, a latency-sparing approach, and a throughput-sustaining approach developed are described in Sections 5, 6, and 7, respectively. Section 8 reports experimental results for a H.263 video encoder's DCT hardware core and an AES encryption engine. Throughput, fault resilience, and energy duty cycle results are compared to the baseline TMR approach which are summarized in the Conclusion in Section 9.

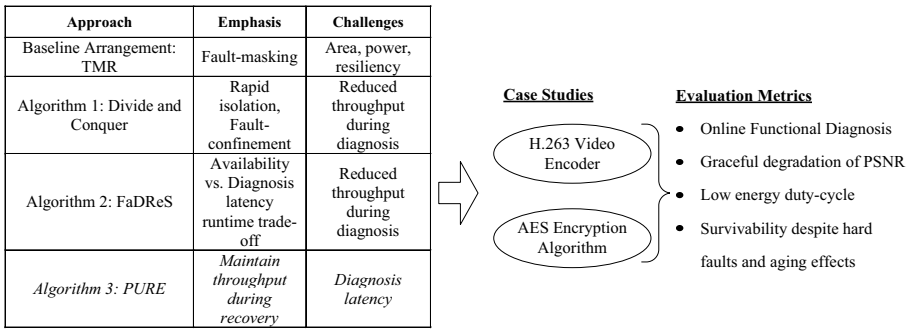


Figure 6.2: Overview of recovery algorithms evaluated herein and the evaluation approach

Functional diagnosis to record discrepancy history

The same diagnosis formulation applies to each of the three algorithms developed and is described first here. Given an undirected graph $\mathbf{G}(V, E)$ of vertex set V and edges set E, the diagnosis objective is to identify faulty nodes. The nodes of $\mathbf{G}$ correspond to either PEs or processors in a multiprocessor network connected through an interconnection network. The diagnosis process is described in terms of CED comparisons to identify discrepancies, however, the analysis is not restricted to a pair-wise comparison. Instead, the fault diagnosis process can utilize N-Modular Redundancy (NMR) in accordance with availability of resources. NMR is a generalization of TMR where $N \geq 2$ modules provide $N - 1$ redundant instances, which has found applicability in adaptive fault-handling [56] [101].

An element (u, v) in the edge set E indicates the feasibility that the output from corresponding PEs can be compared. Let the actual fitness states of nodes be represented by vector Φ, and the fitness states estimated based upon the fault-diagnosis process by vector $\hat{\Phi}$.

The following assumptions are made in the proposed fault diagnosis scheme:

1. Faults are of permanent nature.
2. A fault is observable if a faulty node manifests a discrepant output at least once in a given *Evaluation Window* period.
3. The outcome of a comparison is positive if at least one of the nodes in a CED pair has an observable fault.
4. The comparator/voter is a *golden* element which can be relied upon for fault-free operation.

Let the functions computed by N nodes of a FE be represented by a vector F where f_i is the function performed by node i. In the recovery solution, we seek F^* which gives optimal assignments of functions in a fault-scenario. We define the *Connectivity Matrix* $\mathbf{C}$ to show the comparison performed between two nodes in $\mathbf{G}$. Thus, an entry $c_{ij} = 1$ denotes that a comparison between node i and node j is performed. *Syndrome Matrix* $\mathbf{\Psi}$ indicates the outcome of comparisons. An entry ψ_{ij} of this matrix denotes comparison outcome corresponding to the outputs of node i and node j. Both of these matrices are symmetric about the diagonal due to commutativity of pairwise comparison for discrepancy.

$$\mathbf{\Psi} = \begin{bmatrix} 0 & \psi_{12} & \dots & \psi_{1N} \\ \psi_{21} & 0 & \dots & \psi_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{N1} & \psi_{N2} & \dots & 0 \end{bmatrix} \tag{6.1}$$

Where $\psi_{ij} = 1$ indicates that output from node i and j is discrepant for the same input, $\psi_{ij} = 0$ shows their agreement, while $\psi_{ij} = x$ stands for the case when no comparison has been performed between the corresponding nodes. A $\psi_{ii} = 0$ on the diagonal corresponds to the comparison outcome for a node i with itself,

The syndrome matrix $\mathbf{\Psi}$ is used to estimate the fitness states of nodes in $\mathbf{G}$. Thus, faulty nodes are

identified based upon the syndrome matrix values. After fault detection, all the entries of $\mathbf{\Psi}$ except those on the diagonal are initialized with x implying that the health of all the PEs is suspect. The following identifies the condition for healthiness, with the estimated fitness vector being updated accordingly:

Condition: $\psi(i,j) = 0$ for any $1 \leq i \leq N$ and $1 \leq j \leq N$, where $i \neq j$ and $c_{ij} = 1$
Update: $\hat{\phi}_i = 0$

Thus, the syndrome matrix is used to update the fitness of various PEs based upon diagnosis history information. In case of failure to identify a healthy PE after multiple reconfigurations, the slack is updated to a different PE as described by the specific reconfiguration sequencing algorithms in Section 5, 6, and 7. When a healthy RS is found in a given slack update iteration s, it indicates that the previously selected slacks were faulty.

In the proposed recovery schemes, the priority of functions is taken into account while recovering from fault scenarios. For the DCT case, the PE computing the DC-coefficient is the most important, AC_0-coefficient second most important and so on. Generally, we represent the computational importance of nodes by an $N \times 1$ size priority vector $\mathbf{P}$, where $p_i = 1$ for the most important node i and $p_i = N$ for the least important node. For an application with equally important cores, the priority vector is initialized with all ones. In this work, we assigned the priorities at design-time considering the application properties, e.g., DCT-coefficient computing functions and their impact on PSNR for various video sequences. An interesting future work can be to compute the priority values at runtime. The applications which cannot be characterized by priorities at design-time, or to better utilize the input signal characteristics at runtime, such an approach can be very promising to realize runtime adaptable architectures. An example is to estimate the priority of DCT PEs based upon their runtime impact on PSNR according to the input scene's characteristics.

Given a network, the objective is to identify faulty nodes as soon as possible while maintaining

throughput during fault diagnosis phase. For this purpose, the proposed diagnosis schedule demotes the predicted fitness of a Node Under Test (NUT) based upon their discrepancy history. In the following, we describe some variations of the fault-handling phase starting with a *divide-and-conquer* approach. The choice of algorithm in an application depends upon the designer's preferences about diagnosis latency, throughput availability requirement, and area/power trade-offs.

Reconfiguration Algorithm 1: Divide-and-Conquer Method

Group testing schemes [170][171][103] have been successfully employed to solve many fault isolation problems in which the number of defective items is much smaller than the size of the overall suspect pool. The problem at hand has an analogy to the group testing paradigm, yet with some important distinctions. Although, the task here is to identify defective elements in a pool of computational resources, we do not pose an assumption about presence of a known-to-be-healthy functional output element for testing individual nodes. This assertion makes it infeasible to apply a hierarchical testing approach in which testing up to the last single item is performed by a known healthy item. Therefore, the PURE also relies upon the comparison diagnosis model or NMR voting model to isolate faulty elements.

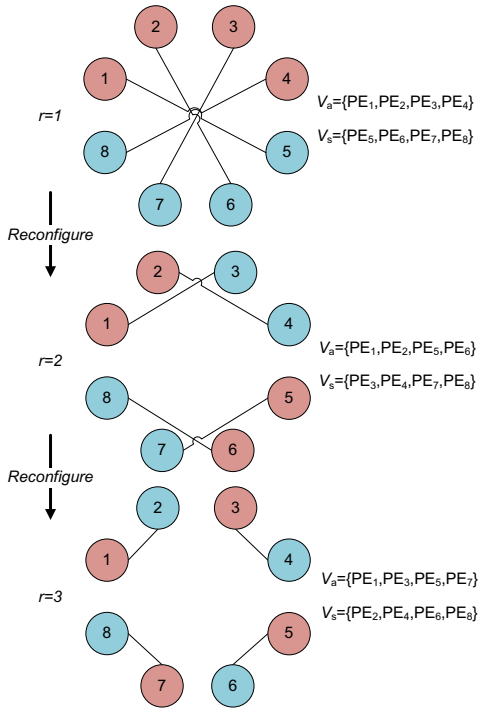
We identify two scenarios in which this hierarchical divide-and-conquer strategy may be more appealing to be employed than the two algorithms discussed in further sections:

- If there are no restrictions on throughput or availability during the fault-handling phase, then halving of the suspect pool [171] offers logarithmic time diagnosis latency, or
- If *fault confinement* is desirable, that is, limiting the influence of the fault as soon as possible, then it becomes advantageous to cut off the suspect nodes from the active throughput path

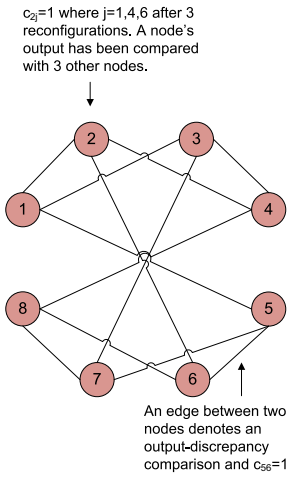
as soon as possible. Then, those nodes can be used for health checking of the active nodes. This scenario is pessimistic, and applies to the case when fault rate is high and a large number of nodes become defective before the fault-handling scheme is initiated. A more optimistic approach is to keep the active nodes in processing datapath while performing diagnosis process as we discuss in the next sections.

Fig. 6.3 illustrates the topologies in the diagnostic flow at various reconfiguration iterations. The number of edges in the graph of Fig. 6.3 corresponds to the total number of reconfigurations performed for diagnosis purposes. Various steps of the diagnosis phase using a divide-and-conquer approach are illustrated in Fig. 6.4 in which dotted lined boxes correspond to the checking slacks and solid lined boxes correspond to active PEs. Algorithm 7 defines the diagnosis process.

To measure the diagnosability of $\mathbf{G}$ obtained by the divide-and-conquer reconfiguration method, we observe from Fig. 6.3(b) that every node has three adjacent nodes. In the worst case, if all the adjacent nodes of a node i become faulty, then it is impossible to check the fitness of node i using a comparison diagnosis model. In that case, the system is no longer diagnosable. However, if only two adjacent nodes of a presumed healthy node j are faulty, then the remaining one node can be used for checking purposes. Thus, the diagnosability t of a divide-and-conquer topology is $(d(G) - 1)$ where $d(G)$ is the average degree of a node in $\mathbf{G}$.



(a) Time varying topologies at various reconfiguration instants



(b) Graph represented by $\mathbf{C}$ after 3 reconfigurations

Figure 6.3: Divide-and-conquer method for fault diagnosis

For example, when PE_4 and PE_6 are faulty in a system with 8 PEs, then after $r = 3$ iterations of diagnosis, the syndrome matrix deduced from Figure is given by:

$$\mathbf{\Psi} = \left[\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 0 & 0 & 0 & x & 0 & x & x & x \\ 2 & 0 & 0 & x & 1 & x & 1 & x & x \\ 3 & 0 & x & 0 & 1 & x & x & 0 & x \\ 4 & x & 1 & 1 & 0 & x & x & x & 1 \\ 5 & 0 & x & x & x & 0 & 1 & 0 & x \\ 6 & x & 1 & x & x & 1 & 0 & x & 1 \\ 7 & x & x & 0 & x & 0 & x & 0 & 0 \\ 8 & x & x & x & 1 & x & 1 & 0 & 0 \end{array}\right] \tag{6.2}$$

where the entry $\psi_{12} = 0$ denotes the healthy nature of PE_1 and PE_2 while $\psi_{42} = 1$ shows the faulty nature of at least one of the PEs in the pair under test.

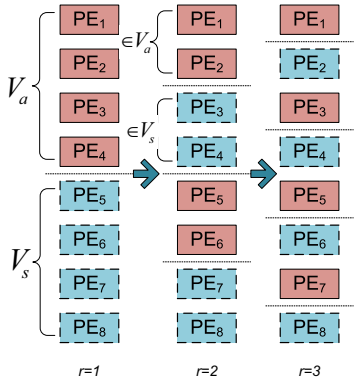


Figure 6.4: Various reconfiguration instants in the divide-and-conquer approach

Algorithm 7 Divide-and-conquer Fault Diagnosis Algorithm (without recovery)

Require: N

Ensure: $\hat{\Phi}$

1: Partition V into two equal-sized disjoint sets V_a and V_s
2: Designate the set V_a as FE and V_s as RS
3: Perform concurrent comparison to the same inputs for various edges of the bipartite graph represented by connectivity matrix $\mathbf{C}$
4: Update the Syndrome Matrix $\mathbf{\Psi}$ based upon comparisons outcome
5: Iterate step-1 to step-4 $log(N)$ times
6: Given $\mathbf{\Psi}$, isolate the faulty nodes:
$\hat{\phi}_i \leftarrow 0$ and $\hat{\phi}_j \leftarrow 0$, if $c_{ij} = 1$, and $\psi_{ij} = 0$
$\hat{\phi}_i \leftarrow 1$ if $\hat{\phi}_j = 0$, $c_{ij} = 1$, and $\psi_{ij} = 1$

Reconfiguration Algorithm 2: FaDReS

Fault Demotion using Reconfigurable Slacks (FaDRes) achieves dynamic prioritization of available resources by demoting faulty slacks to the least priority functions [45]. Compared to divide-and-conquer, it attempts to avoid excessive reconfiguration of the processing datapath. Namely, whenever a redundant PE is not available then a lower priority functional module can be utilized. The output from the vacated RS is compared against functional modules in the datapath providing normal throughput. The discrepancy in output of identical functional modules isolates the permanent or transient fault. Thus, the FaDRes algorithm iteratively evaluates the functional modules while keeping them in the datapath, as well as slack resources used for checking. In general, the identification of healthy slack can be formulated as follows: Given a pool of resources in which the faults are equiprobable in any resource, then what is the probability that at least a single RS is identified within r iterations. The probability of favorable event corresponding to a RS being identified is given by:

$$P(X) = \frac{Number\ of\ favorable\ scenarios}{Total\ number\ of\ diagnosable\ fault\ scenarios} \tag{6.3}$$

where X = Number of healthy RS identified. The *Cumulative Proportion of Diagnosable Conditions (CPDC)* is defined as:

$$CPDC(X \geq 1) = \sum_{r=1}^{N} P(X = r) \tag{6.4}$$

For the case of $N = 9$ total PEs with a single RS, a total of $N_a = 8$ number of *Active* PEs form the throughput datapath of the circuit while number of slacks is $N_s = 1$. Since each PE can either be faulty or healthy, there are 511 unique fault-scenarios in addition to one case where all are healthy. However, two special cases in which none or only one PE is healthy, are non-diagnosable. This yields 10 non-diagnosable configurations corresponding to 9 when one PE is healthy plus one when none are healthy. The RS itself is healthy for a total of 254 of all possible faulty-yet-diagnosable $511 - 10 = 501$ cases. Thus, the proportion of diagnosable conditions is $\frac{254}{501} = 0.5070$.

If a healthy RS is not identified in the first testing iteration, it is marked and not included in the second testing iteration. Then, given a total number of $N = 9 - 1 = 8$ PEs yields 127 diagnosable fault-scenarios involving a healthy RS. Thus, CPDC is given by $\frac{254+127}{501} = 0.7605$ at the $r = 2$ iteration. Similarly, a failure to identify a healthy RS in the second testing iteration leads to testing another set of configurations in which $N = 7$. Here, 63 diagnosable faulty scenarios involve a healthy RS. Thus, $CPDC(r = 3) = \frac{254+127+63}{501} = 0.8862$, in agreement with Eq. 6.4.

Fig. 6.5 demonstrates benefit of employing multiple slacks during diagnosis procedure. As it can be seen, the probability of diagnosis completion after the first instance of testing arrangement is higher in case of $N_s = 2$ compared to the case $N_s = 1$.

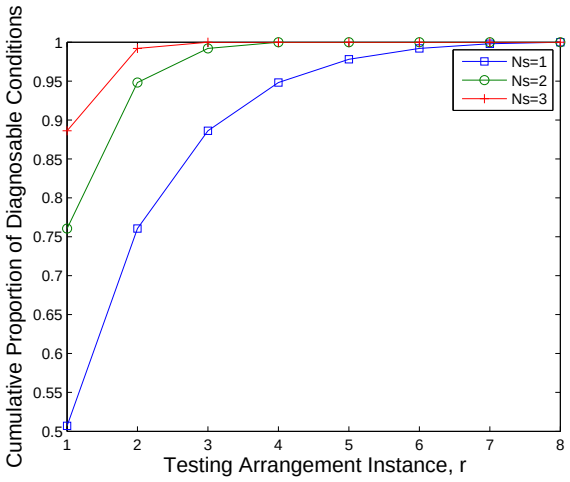


Figure 6.5: CPDC demonstrating diagnosis benefit of additional slacks

The proposed FaDReS architecture is overviewed in Fig. 6.6 where redundancy is employed to isolate and recover from faults, thereby avoiding exhaustive test vectors to test the functions configured on hardware fabric. Once the proposed system detects fault, the fault isolation phase is initiated by reconfiguring multiple PES with the same functionality for discrepancy checking. The dynamic reconfiguration property of FPGA is employed to create the redundancy needed for fault isolation during runtime. If there is no redundant PE available, the lower priority functional module can be vacated. The output from the vacated RS is concurrently checked for discrepancy with that of functional modules in the datapath providing normal throughput. The discrepancy in output of the same functional modules reveals the permanent or transient fault. The proposed algorithm iteratively evaluates the functional modules while keeping them in the datapath, as well as slack resources used for checking. Dynamically configurable slacks can isolate up to *N-2* faulty PEs in a system having N total number of PEs. DMR and TMR techniques have been successfully adapted to utilize dynamic reconfiguration capability of FPGAs in previous research [7]. We propose an isolation algorithm employing flexible dynamic redundancy. It will also be shown how the isolation process can be accelerated by increasing the number of RS.

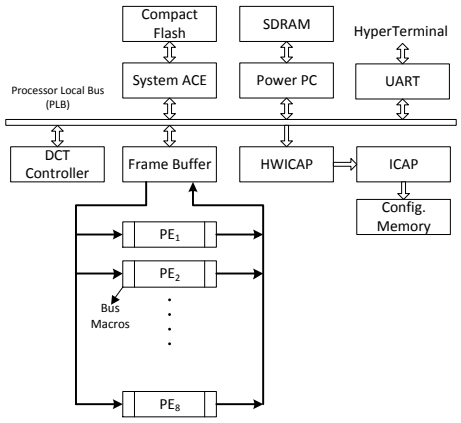


Figure 6.6: The FaDReS approach applied to an H.263 architecture

Hardware Organization in FaDReS Technique

A baseline reallocation strategy of fault demotion for mission-critical applications is proposed that permits recovery emphasizing small throughput degradations via an RS with stepwise PR demotion. Nonetheless, more efficient reallocation strategies such as Divide and Conquer, and others, can be employed when their impact on throughput is acceptable [31]. The hardware architecture to evaluate the proposed scheme is shown in Fig. 6.6 and implemented on a Xilinx ML410 development board which contains a Virtex-4 FX60 FPGA. As a case study, the H.263 video encoder application is run on the on-chip processor. All the sub-blocks except the DCT block [94] are implemented in software, the later being implemented in hardware. Each PE in the DCT block is responsible for computing one coefficient of 8-point 1-D DCT.

The development environment for the processor-based system is Xilinx Platform Studio (XPS9.2). This system is interfaced with the DCT core inside an ISE9.2i project. PlanAhead10.1 manages the partial reconfiguration regions and generates the bitstreams. The GenACE utility is used to map the full system bit file and the executable file into an ACE file which is stored on a Compact

Flash. The Software Development Kit (SDK) is used to build the software for the encoder. Some of the hardware modules are described in detail:

Hardware Components

PowerPC: The Xilinx PowerPC 405 processor core is a 32-bit embedded implementation derived from the PowerPC architecture [172]. There are such two cores embedded in the reconfigurable fabric, and the design presented here utilizes one of them. The Standalone *Board Support Package (BSP)* is used which is a single-threaded simple operating system [173]. It provides minimal libraries for interfacing with hardware. The video encoder application is operational on the processor except its DCT core which is implemented in hardware. Moreover, the processor is responsible for sequencing our fault handling methodology, but without loss of generality it can be realized directly in the FPGA with appropriate fault tolerance.

DCT Core: The processor communicates with the DCT core through GPIOs. The DCT core is composed of a DCT controller, transposition memory, and PEs. To meet the Xilinx specific requirement for a partial reconfiguration design, bus macros are inserted between the transposition memory and the PEs which define each *reconfigurable region*. Each PE has a stored DCT kernel and is responsible for computing one DCT coefficient for a row of input pixels. For instance, PE_1 contains the DC kernel of the DCT block and computes the DC value of a row of input pixels.

The DCT controller manages to read pixels for the input of the PEs from the frame buffer row-wise during the stage-1 of the DCT. The 1-D DCT is computed in the first stage, then in the second stage the controller reads the frame buffer column-wise. The processor writes the pixel values in the frame buffer and DCT core computes the coefficients upon the availability of a macroblock. The precision of the input pixels is set to 9 bits per pixel, and the kernels are stored in a 12-bit fixed-point format. The output from the second stage is rounded to 12 bits and is communicated

to the PowerPC through the GPIO core. We have implemented a pipeline design of the DCT core taking advantage of the parallel datapath capabilities of FPGAs. The effective throughput of the DCT core is one pixel per clock. Internally, the PEs use DSP48 blocks available in the Virtex-4 FPGA. Overall nine DSP48 blocks are used for 9 PEs. The core can execute at rates up to 108 MHz. The static modules of the design include PowerPC, DCT controller, Frame Buffer, Digital Clock Manager (DCM), DDR SDRAM, CompactFlash controller and GPIO cores.

ICAP: The *Internal Configuration Access Port (ICAP)* in the FPGA chip allows access to configuration bitstream data [174]. This access can be for the readback purpose or partial reconfiguration. The Xilinx HWICAP core serves as an interface between the PowerPC and the ICAP. The processor communicates with the core via Xilinx software library-routines. Using these routines, the partial bitstreams of PEs are downloaded from the CompactFlash to the configuration memory. Depending upon the size of the bitstream files and the overhead of library functions, each PE reconfiguration takes about 200 milliseconds on average. This time period constitutes 6 frames of video at 30 frames per second input frame rate.

Fault Detection, Isolation and Recovery

The fault detection methodology used here is observing PSNR of the recovered frames. The PSNR is computed based upon the difference between input frame and the image in frame buffer as described in [58]. Once this measure is below a threshold δ, fault detection is asserted. In this way, continuous exhaustive testing of the resources is avoided. In addition, avoiding redundancy during the normal operation is beneficial in terms of power and area requirements. Once a fault is detected, the proposed fault isolation and recovery process are initiated. The FaDReS algorithm for *Fault Isolation (FI)* and *Fault Recovery (FR)* scheme is given below and illustrated by the flow chart in Fig. 6.7.

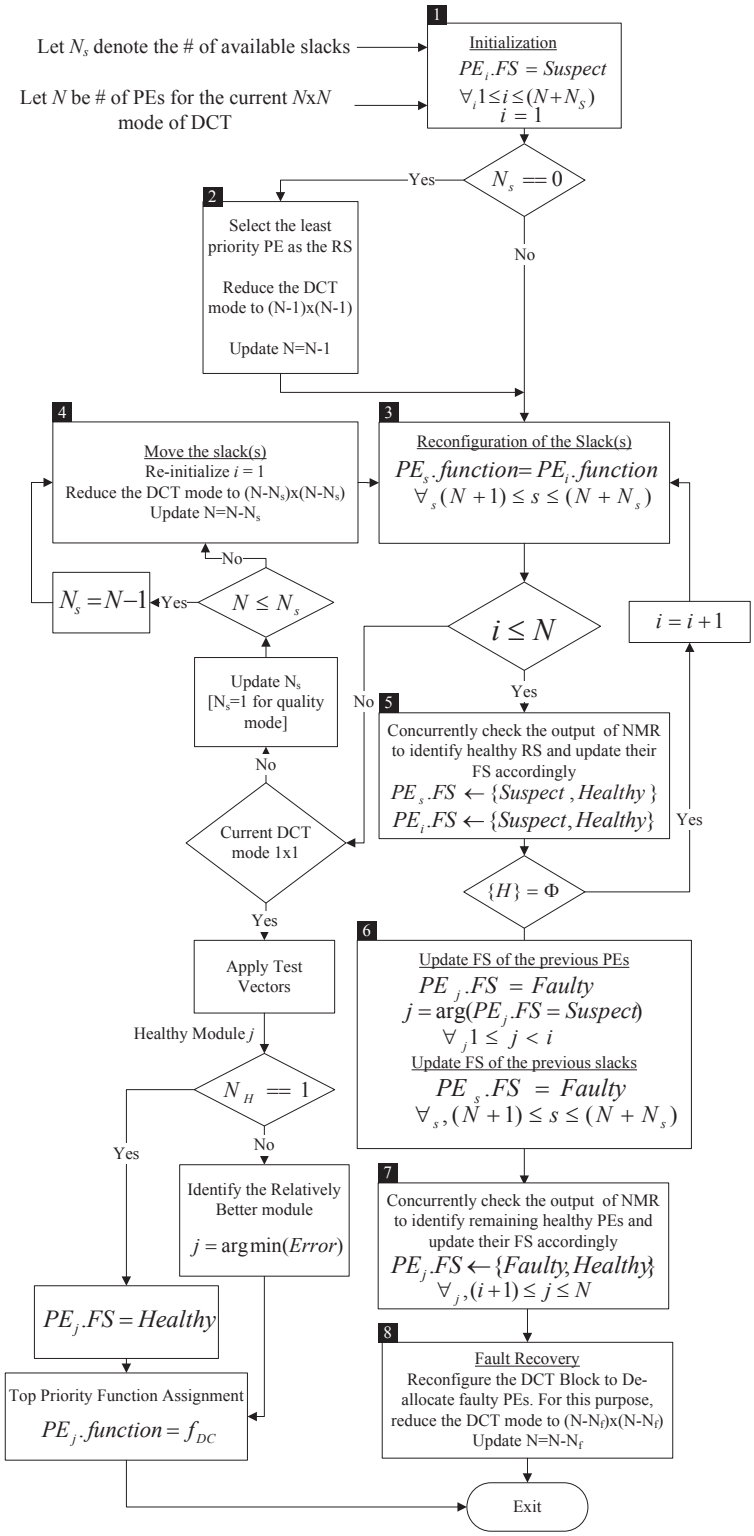


Figure 6.7: The fault isolation and recovery process flow chart

1: Obtain current system parameters (e.g., Current DCT mode, Number of Blank PEs)

2: Determine the number of RS(s).

3: Apply the proposed fault isolation algorithm (Fig. 6.7)

4: Isolate faulty PEs (Bounded Number of Reconfigurations as in Fig. 6.8)

5: Reconfigure the functionality for full recovery or a gracefully degraded mode (e.g., DCT mode)

An NMR system is the one in which N instances compute for the same input and the effective output is the majority output. In this work, the fault model is that if at least one of two PEs is faulty in a pair under evaluation, they exhibit discrepancy at least once in a given *Evaluation Window* or their output remains the same. Similarly, no discrepancy between the module outputs reveals their fault-free *Healthy* status. The terminology used in this dissertation is listed below. The terms *module* and *PE* are used interchangeably.

$PE_i.FS$: Fitness state of i^{th} Processing Element

FS Enumeration : $\{Healthy, Suspect, Faulty\}$

$PE_i.function$: The functionality assigned to a particular PE

Functional Modules : The modules providing normal throughput

$\{H\}$: The set containing all the healthy modules from RS

Once fault occurs and every module needs to be examined, we proceed to identify the faulty PE(s) by employing the RS. In the first step, all the PE's fitness is suspected as labeled block `1` in Fig. 6.7. The slacks may be the blank PEs available in the system. The RS (or multiple slack) is reconfigured with the same functionality as that of the most important functional PE, for example, the module for computing DC coefficient (label `2` and `3`).

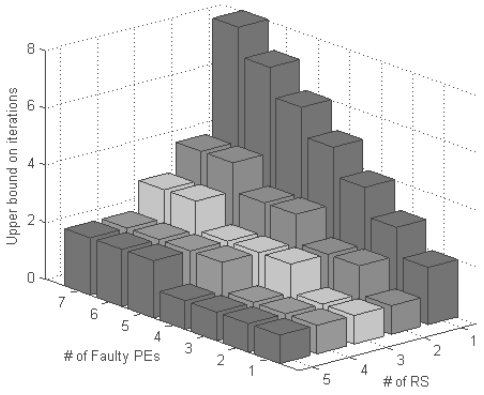


Figure 6.8: Upper Bound on number of iterations for fault isolation

The location of faulty PE is detected by performing the discrepancy check in an NMR arrangement (label 5). For DMR, faulty status of one of the modules whereas for NMR faulty status of more than $N - 2$ modules marks each of the instances as *Suspect*. Therefore, we proceed to reconfigure the RS with the second priority function and so on(label 3). Once an agreement between two modules over a complete evaluation window is observed, the two modules are declared as *Healthy* and their fitness state is updated (label 6). The identification of a healthy RS implies we do not need to reconfigure the PEs as slacks any more. A healthy RS can be used to check the fitness of all the modules (label 7). The discrepancy of a suspected module in pair with a healthy module reveals its *Faulty* nature. On the other hand, an observed discrepancy between suspected modules does not provide any information and keeps them marked *Suspect*. If a *healthy* RS is not identified in the first iteration even after reconfiguring with all of the functions in the datapath, it is moved to the next PE, and so on (label 4). Upon the completion of fault isolation, the priority functions are moved to the *healthy* PEs (label 8), which accomplishes the recovery process.

The fault isolation scheme is illustrated by an example in Fig. 6.9. Here, the normal operation is 8×8 DCT and therefore the PE_1 to PE_8 are providing the normal throughput for 1-D 8-point DCT. Two 1-D DCT operations are performed to compute 2-D DCT. For this arrangement, PE_9 is

required by the system and reconfigured as blank. For this specific situation, assume that 2 out of total 9 PEs are faulty. The faulty PEs labelled `1` and `9` are shown by tagged boxes. We choose the blank PE as the RS for discrepancy check with PE_1. Unfortunately, the RS itself is faulty, yet we have no a-priori knowledge of its fitness state. Therefore, in the first iteration in which the RS is reconfigured 8 times for the CED purpose, no information is obtained. Every module's fitness is unknown and marked as *Suspect* (S). We proceed to the second iteration. The RS is moved from PE_9 to PE_8 and CED is performed with the functional modules, sequentially. No discrepancy between PE_8 and PE_2 implies their *Healthy* (H) nature and therefore, PE_1 and PE_9 are marked as *Faulty* (F) as they did exhibit a discrepancy with the *Healthy* module.

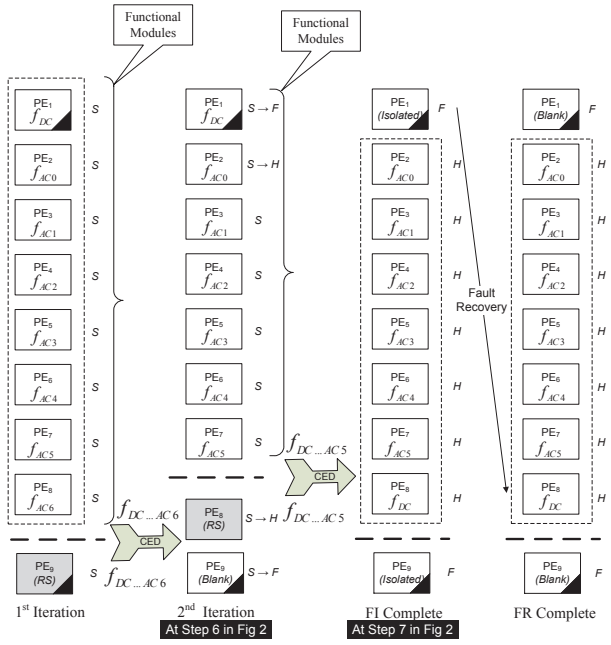


Figure 6.9: An example of the fault isolation and recovery scheme

Once the fault isolation process is complete and a healthy RS is identified throughout the evaluation window, that RS is used for error checking of all the functional modules. If there are N healthy PEs available, fault recovery is made without quality degradation. However, if there are fewer healthy modules than the current mode of DCT, then a graceful degradation strategy is employed where

the most important functions are prioritized to be retained in the proven resources. In the DCT case, the most influential function to retain is the DC coefficient's computation.

Fault detection latency of the proposed approach based upon PSNR is negligible, whereas latency of isolating faulty PEs is bounded as our algorithm follows a deterministic flow. The idea is to time-multiplex the RS for different functions and compare the output of the RS with those from the active modules in the datapath. A discrepancy between the outputs of two modules results in them remaining in the *Suspect* pool, whereas the agreement marks them as *Healthy*. Fig. 6.8 shows upper bound on the number of iterations when using different number of slacks for various fault situations. For all these figures, the total number of PEs is 9 but can be extended without loss of generality. As an example, we require 2 iterations at most to isolate 7 faulty modules out of 9, when using 4 slacks.

Experimental Results

Performance Improvement

The proposed fault handling scheme is validated by observing the fault behavior of H.263 video encoder's DCT block. Faults are injected into the PE_1 and PE_9 at frame 50. For simulating the fault behavior, SA faults are injected at logic resources' inputs in the post-place and route simulation model of the circuit. For this purpose, we employed Fault Injection and Analysis Toolkit (FIAT) [175] which modifies the *User-Constraints File (UCF)* in the Xilinx design flow. As shown in Fig. 6.10, the PSNR of the video sequence drops below the threshold δ = 28dB at frame 59 and the fault isolation scheme is invoked. After a few frames, the DCT block is recovered, improving PSNR back to a normal condition. It may be noted, however, that we simulated a relatively demanding scenario here where the number of healthy PEs available is less than N.

Otherwise, there would be no quality degradation. Also, it is worthwhile observing that during fault isolation, considerable throughput is available. The input video sequence to the encoder is `forman.qcif` with the resolution of 176×144. It was observed that the lowest fault-free PSNR was 29.96dB throughout the video sequence.

The PSNR measure of the image is a direct indication of its visual quality. Fig. 6.11 illustrates the qualitative results of the approach. The recovered image in Fig 6.11(c) after applying the fault handling scheme is visually much better than Fig. 6.11(b) where a faulty PE in the DCT block disrupts the image in the frame buffer.

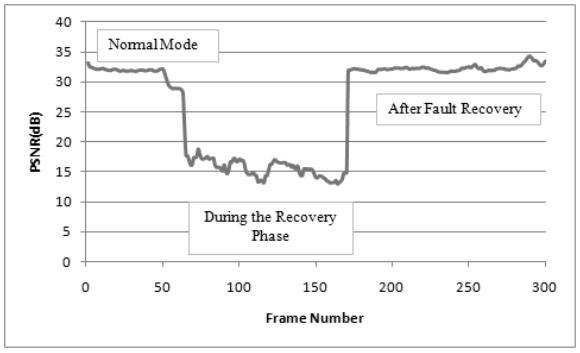


Figure 6.10: An operational example of the video encoder

(a) Reconstructed frame (without faults)

(b) The effect of a faulty PE

(c) Recovered frame (after fault-handling)

Figure 6.11: Qualitative results on sequence from ASU video library [3]

Power Analysis

To analyze the power-overhead of the proposed scheme, Xilinx XPower Estimator (XPE) 11.1 has been utilized. The static non-reconfigurable design power requirement is 1541mW which consists of 591mW Quiescent and 950mW Dynamic Power. The individual components of the total dynamic power are listed in Table 6.1. In addition, the power estimates of *Partial Reconfiguration Modules (PRMs)* are listed in Table 6.2. An estimate of power consumption for reconfiguration is based upon the resource count of HWICAP core reported in the Xilinx datasheet [176]. The measurement and estimation of power consumption in Virtex FPGAs is thoroughly demonstrated in [177]. The PEs employ a power-efficient design since the utilized DSP48 multipliers generally consume only 2.3mW operating at 100MHz at a typical toggle rate of 38% [178]. As illustrated by Table 6.2, considering the example of 8×8 DCT module, fault-tolerant version consumes more power (i.e., 142mW) than that required by a simple DCT module (i.e., 72mW). However, the extra cost of power is considerably less than that required by TMR which would consume a factor of three times the power of a single module. Table 6.3 lists the *Dynamic Energy Consumption, E* of the *Partial Reconfigurable (PR)* design throughout the isolation phases for various fault-rates described in terms of number of faulty modules, N_f. Thus, operating in uniplex mode with a single reconfigurable slack, the FaDReS power consumption including the HWICAP totals only 65.7% of TMR during fault-handling and roughly only 33% of TMR during fault-free operation.

Table 6.1: Dynamic power consumption of the static design

Resource	**Utilization Count**	**Dynamic Power (W)**
Clock	2	0.113
Logic	8812 LUTs *	0.328
I/Os	144	0.361
BRAM	1	0.002
DCM	1	0.082
PowerPC Processor	1	0.064
Total		0.950

* Includes dynamic power of 5656 FFs, 445 Shift Registers, and 71 Select RAMs

Table 6.2: Dynamic power consumption of the reconfigurable design

Module	**Resource Count**	**Dynamic Power (W)**
PRM_1	82 LUTs, 44 FFs, 1 DSP48	0.021
PRM_2	117 LUTs, 78 FFs, 1 DSP48	0.023
PRM_3	103 LUTs, 64 FFs, 1 DSP48	0.021
PRM_4	120 LUTs, 81 FFs, 1 DSP48	0.023
PRM_5	94 LUTs, 56 FFs, 1 DSP48	0.021
PRM_6	122 LUTs, 83 FFs, 1 DSP48	0.023
PRM_7	107 LUTs, 68 FFs, 1 DSP48	0.023
PRM_8	113 LUTs, 74 FFs, 1 DSP48	0.023
HWICAP	2115 LUTs, 728 FFs	0.061
4×4 DCT		0.044
6×6 DCT		0.059
8×8 DCT		0.072
8×8 DCT with one RS		0.081

Table 6.3: Dynamic energy consumption of the PR design during FI phase

$\mathbf{N}_f$	1	2	3	4	5	6	7
E (Joules)	1.52	2.89	4.37	5.82	7.08	8.04	8.64

Diagnosis by voting

The algorithm for diagnosis employing dynamic NMR voting on module level is given in Algorithm 8. Fig. 6.12 shows various steps in the diagnosis process.

Algorithm 8 FaDReS (Greedy fault-diagnosis with subsequent recovery)

Require: N, N_s, $\mathbf{P}$

Ensure: $\hat{\Phi}$

1: Initialize $\hat{\Phi} = [x\ x\ x\ ...\ x]^T$, $i = 1$, $N_a = N - Ns$
2: Arrange elements of V in ascending order of $\mathbf{P}$
3: **while** $(\{k|k \in \hat{\Phi}, k = 0\} = \phi)$ **do**
4: Designate v_s as checker(s) $(N_a + 1) \leq s \leq (N_a + N_s)$; thus $V_s = \{v_s\}$
5: **while** $i \leq N_a$ **do**
6: Reconfigure RS(s) with the same functionality as v_i, $N_{sup} = N_{sup} + 1$
7: Perform NMR majority voting among NUTs when $N_s > 1$, or CED between NUTs when $N_s = 1$, then update Connectivity matrix accordingly,
Update the Syndrome matrix $\mathbf{\Psi}$ based upon discrepancy information,
$\hat{\phi}_i \leftarrow 0$ for v_i which shows no discrepancy then go to step-12, $\hat{\phi}_i \leftarrow x$ otherwise
8: $i \leftarrow i + 1$
9: **end while**
10: Move the RS by updating $N_a = N_a - N_s$, $N_r = N_r + 1$, Re-initialize $i = 1$
11: **end while**
12: Update the fitness state of the previous RS(s): $\hat{\phi}_j \leftarrow 1$; for $(s + 1) \leq j \leq N$ and $\psi_{j.} = 1$
13: Use a healthy RS to check all other nodes in V_a, $\hat{\phi}_i \leftarrow 0$; if $\hat{\phi}_j = 0$, $c_{ij} = 1$, and $\psi_{ij} = 0$

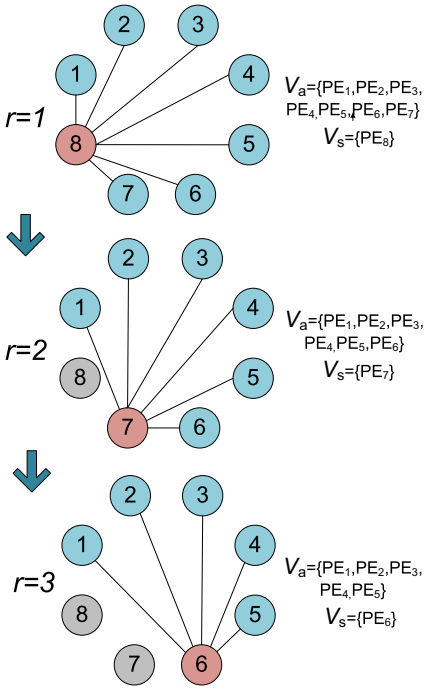


Figure 6.12: Fault-diagnosis in the FaDReS approach

Fault diagnosis latency T_{diag} is defined as:

$$T_{diag} = (T_{eval} + T_{rec}N_s) \sum_{j=1}^{N_r} I_j \tag{6.5}$$

where

N_r = Number of testing arrangement iterations during detection

I_j = Number of times a j_{th} RS is reconfigured

T_{rec} = Reconfiguration Latency (PR time for one PE)

N_s = Number of Reconfigurable Slacks

T_{eval} = Duration of Evaluation Window

By substituting $N_r = N_a$ and the worst case reconfiguration count, the upper bound on the latency of the fault-diagnosis is obtained as:

$$T_{diag,max} = (T_{eval} + T_{rec}N_s) \sum_{j=0}^{N_a-1} (N_a - j) \tag{6.6}$$

Diagnosis by Comparison

A variation of Algorithm 2 is made in which a NUT is assigned to only one RS for checking; whereas more than one RS(s) may be allocated to a NUT in the diagnosis-by-voting case. For example, in diagnosis by comparison approach with $N_s = 2$, the first RS is configured with f_1 and the second RS with f_2 in the first iteration. Upon failure of identifying a healthy RS, these slacks are reconfigured to f_3 and f_4, respectively and so on.

In case of Xilinx FPGAs, the ICAP, on-chip memory called Block-RAM, and Compact Flash external memory form a memory hierarchy for reconfiguration functions. The bitstreams which define the functions configured to various PEs are initially stored in external memory. We employ a local-

ity constraint to quantify the distinction between the voting approach and comparison approach. If an RS is to be configured with a function, the corresponding bitstream needs to be fetched from the external memory for the first time. However, if another RS needs to be configured with the same function, a bitstream fetch operation is not required as the access can be granted from on-chip memory. Thus, if two RS's are to be configured with the same functionality, the reconfiguration penalty is not $2 * T_{rec}$ but just $(1 + \beta) * T_{recon}$ where $0 \leq \beta \leq 1$ depends upon the ratio between internal on-chip memory access time and external memory access time. On the other hand, a comparison diagnosis approach requires $2 * T_{rec}$ reconfiguration time for two slacks as both need to be configured as separate functions. The preference of one method over the other should be based upon reconfiguration time T_{recon}, β factor, and evaluation window period T_{eval}. For devices with fast on-chip memory access provision, β is a small number and hence comparison-by-voting can be more advantageous approach. For Virtex-4 device with external compact-flash and internal block-RAM, we observed a value of $\beta = 0.0013$ when operating the reconfiguration port at 100MHz clock frequency.

Reconfiguration Algorithm 3: PURE

PURE achieves dynamic prioritization of available resources by assigning healthy slacks to the highest priority functions. The distinction between the PURE algorithm and FaDReS arises from the fact that after a healthy RS is identified, PURE configures it for priority function computation immediately. An identified healthy RS is used for checking purposes to isolate all other PEs. Thus, the Algorithm 9 can be used to prioritize throughput while the FaDReS Algorithm 8 can be used to prioritize fault diagnosis completion.

Algorithm 9 PURE (Fault-diagnosis with integrated priority-driven recovery)

Require: N, N_s, $\mathbf{P}$
Ensure: $\hat{\Phi}$, F^*

Initialize $\hat{\Phi} = [x\ x\ x\ ...\ x]^T$, $i = 1$, $N_a = N - N_s$
Arrange elements of V in ascending order of $\mathbf{P}$
while $(\{k|k \in \hat{\Phi}, k = x\} \neq \phi)$ //Until all suspect nodes are proven to be healthy **do**
  while $(\{k|k \in \hat{\Phi}_s, k = 0\} = \phi)$ //Identify at least one healthy node in V_s **do**
    Designate v_s as checker(s) $(N_a + 1) \leq s \leq (N_a + N_s)$; thus $V_s = \{v_s\}$
    while $i \leq N_a$ **do**
      Reconfigure RS(s) with the same functionality as v_i, $N_{sup} = N_{sup} + 1$
      Perform NMR majority voting among NUTs when $N_s > 1$, or CED between NUTs when $N_s = 1$, then update Connectivity matrix accordingly, Update the Syndrome matrix $\mathbf{\Psi}$ based upon discrepancy information,
      $\hat{\phi}_i \leftarrow 0$ for v_i which shows no discrepancy then go to step-13, $\hat{\phi}_i \leftarrow x$ otherwise
      $i \leftarrow i + 1$
    end while
    Move the RS by updating $N_a = N_a - N_s$, $N_r = N_r + 1$, Re-initialize $i = 1$
  end while
  Identify the most prioritized function computing node which is faulty, v_{pf}
  Use the identified healthy RS to compute a priority function, $F_s^* \leftarrow F_{pf}$ thus RS is removed from V_s and added to V_a
end while

Diagnostic Flow

In the PURE approach, the diagnosability of the system is incrementally improved by reconfiguration. The diagnosability $t_r(G)$ at a reconfiguration instant, r is defined by the average degree of active nodes in $\mathbf{G}$, and is given by the equation:

$$t_r(G) = d_r(G) - 1 \tag{6.7}$$

where $d_r(G)$ is the average degree of nodes in the graph at r. The topology at $r = 1$ in Fig. 6.12 is 0-diagnosable since a faulty RS leaves all other nodes suspect after comparisons. However, the topology defined by $\mathbf{C}$ at $r = 2$ which combines diagnosis information of $C(1)$ and $C(2)$ is 1-

diagnosable since a single faulty node is guaranteed to be identified. In general, the diagnosability at the completion of algorithm is $N - 2$ after every possible pair combination is evaluated and the resultant topology is a fully connected graph. Fig. 6.13 shows the diagnosability at various reconfiguration instants for a network of 8 nodes. As it can be seen, an increase in the number of slacks results in identification of defective nodes within a few iterations.

Fig. 6.14 shows an illustrative example of the fault diagnosis in the PURE approach. The fitness state of PEs which are suspect is depicted by rounded-corner blocks. In this example, PE1 and PE7 are afflicted with faults. Upon initialization of the fault-handling algorithm, all PEs are suspect. Then, PE_8 is reconfigured as RS by implementing function f_1 and its output is compared with that of PE_1 to check for any discrepancy. An RS is shown by dashed block. In this example, PE_1 is also faulty; therefore, this first comparison does not provide any useful information about the health of PEs and they remain suspect. Next, PE_8 is reconfigured to second priority function f_2 and its discrepancy check is performed with PE_2 which implements f_2. An agreement reveals their healthy nature. In addition, it shows that PE_1 was faulty as it had exhibited discrepancy with a healthy PE (i.e., PE_8) in the previous step. As soon as a healthy RS is identified, a faulty PE implementing a priority function is moved to the RS. Thus, PE_1 is configured as blank by downloading a blank bitstream while PE_8 is configured with function f_1 to maintain throughput. Next, PE_7 is chosen as RS whose discrepancy with a healthy PE_2 shows its faulty nature. Lastly, a healthy PE_6 serving as RS accomplishes the diagnosis procedure to update the fitness state of PEs 2 through 5 to healthy. Overall, the fault recovery is achieved by configuring faulty PEs by blank and healthy PEs by functions 1 through 6.

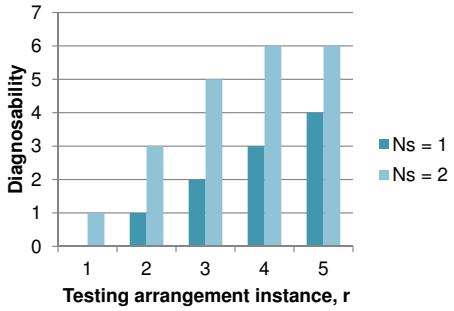


Figure 6.13: The diagnosability of a topology with various reconfiguration iterations

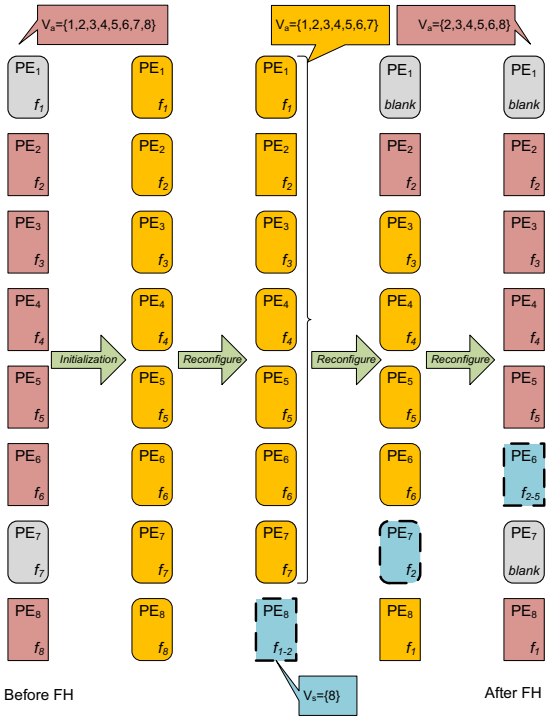


Figure 6.14: An example of fault diagnosis in PURE approach

Another scenario can be considered for the above example in which two checker PEs are utilized in the diagnostic stage. As the intermediate results have to be written into data buffer as in Fig. 1,

so that the CPU can evaluate for discrepancy check, the data buffer writing timing would be different than the previous scenario. In general, for a given faulty-scenario, an increase in N_s can help reducing the latency of diagnosis completion. On the other hand, to improve the fault-diagnosis latency, such a choice of larger N_s can incur more throughput degradation during the diagnosis phase. Thus, the choice of N_s should be made according to maximum tolerable throughput degradation during the diagnosis phase and the desired latency of fault-handling.

For a total of N nodes in $\mathbf{G}$, there are $N^2 - N$ possible pairings. As evident by Table 6.4, our fault-diagnosis schemes require significantly fewer comparisons compared to the exhaustive pair evaluations where the values are scaled to % of total resources available during diagnosis. Fig. 6.15 shows the worst case scenario for the PURE algorithm in an FE containing 8 PEs. The round corner blocks correspond to faulty PEs. As shown in Fig. 6.15, as many as 3 reconfiguration iterations are required as the first two slacks selected were faulty.

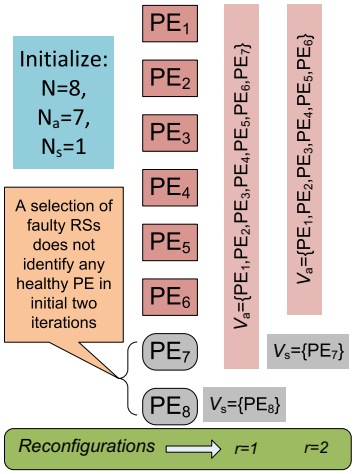


Figure 6.15: The worst case scenario for the diagnostic phase with two defective nodes

Table 6.4: Latency vs. throughput comparisons

Metric	Approach	Testing arrangement instance, r		
		1	2	3
N_a during diagnosis	Algo. 1 (Divide & Conquer)	50%	25%	12.5%
	Algo. 2 (FaDReS)	87.5%	75%	62.5%
	Algo. 3 (PURE)	87.5%	75%	62.5%
No. of bitstream downloads, N_{sup}	Algo. 1 (Divide & Conquer)	50%	50%	50%
	Algo. 2 (FaDReS)	87.5%	75%	62.5%
	Algo. 3 (PURE)	87.5%	75%	62.5%

Fault Detection Criteria

PURE adapts the configuration of the processing datapath based on the correctness and performance of recent throughput by incorporating a health metric.

PSNR as a Health Metric

PSNR is well-established metric to assess the relative quality of video encoding [179]. The PSNR of a $M \times M$ frame of n pixel-depth is computed based upon the algebraic difference of the input frame and the image in the frame buffer in $O(M^2)$ steps. In the PURE technique, the PSNR of each frame is computed in the background using the On Chip Processor already embedded in fabric without decreasing the throughput of the PE array. In the experiments herein, the computation of PSNR was measured to take 4.23msec for the DCT input image luma resolution 176×144, and thus incurs only 2.79% time utilization of the embedded PowerPC. Likewise, the power consumption overhead during PE reconfiguration is 70mW considering ICAP and RS utilized power [45]. Thus, this approach can be advantageous in terms of power and area requirements by detecting anomalies without incurring redundancy within the PE datapath. Meanwhile, PSNR computations on the processor proceed concurrently with DCT computations in the PE array. PSNR computation is

performed as a health metric and is not on the PE array's critical path of the DCT core. Thus, the interval of time between successive calculations of PSNR can be selected independently to be sufficient for health assessment without impacting the DCT core's throughput.

The occurrence of hardware errors resulting in a decrease in PSNR has been validated in the literature [56] [180] [20],[47]. For example, in [56] the authors developed an alternative resilience approach called Soft NMR. It used real-time signal difference to compensate for anomalies exposed by voltage over-scaling, and they evaluated the resilience of their circuits using PSNR. In [180] and [47], PSNR is used to quantify the graceful degradation achieved in a Motion Estimation engine, DCT application, and an Inverse DCT circuit as the supply voltage is reduced. Their research investigated supply voltage reduction from 1.2V to 0.71V causing errors that decreased PSNR 34.9dB to 24.8dB and deemed the maintenance of PSNR above 20dB as achieving acceptable performance. The impact that faults have on PSNR and the resulting image quality are also visually apparent. For instance, Fig. 6.16 depicts PSNR of 35.27dB, 7.07dB, 29.86dB, and 34.78dB resulting from error-free, PE_1 faulty, PE_2 faulty, PE_7 faulty respectively, for a typical frame from the `city` sequence.

(a) Image in frame buffer computed using healthy PEs, PSNR=35.27dB

(b) Image in frame buffer computed using DCT with a faulty PE_1, PSNR=7.07dB

(c) Image in frame buffer computed using DCT with a faulty PE_2, PSNR=29.86dB

(d) Image in frame buffer computed using DCT with a faulty PE_7, PSNR=34.78dB

Figure 6.16: The impact of faults on PSNR and image quality

While these previous approaches utilize PSNR for assessing resilient architecture performance, the novelty of the PURE technique is to escalate resources based on their impact on PSNR. In particular, the PURE scheme maintains quality above a certain user-specified tolerance by adapting the datapath. Taking a broad view, a system boundary is defined so that external factors such as environment, occlusions, or signal transmission errors reside outside of the signal processing task. For example within the system boundary of a video encoding task, PSNR reflects the compression quality if the input noise is considered to be part of the input signal. Thus, the PSNR reflects the effectiveness of the signal processing system in terms of its underlying hardware resources. However, even in the absence of faults, PSNR varies depending on the algorithms ability to perform

lossy compression and reconstruction in accordance with the nature of the scene's content. For example in the PURE results shown in Fig. 6.19 of the following section, PSNR is seen to decline from 33dB down to 32dB during frames 1 through 50. When PSNR drops abruptly at frame 51, due to a hardware fault, it triggers the Fault Detection phase of the PURE algorithm.

The PURE algorithm differentiates failure-induced changes in PSNR from ambient changes in PSNR using a user-selected maximum tolerable quality degradation during Fault Detection (FD), denoted as Δ_{FD}. The quantity Δ_{FD} represents an allowable runtime percentage change in PSNR which would invoke the PURE diagnostic flow. A sliding window of recent PSNR values is used to accommodate differences in the changing nature of the scene's content. Δ_{PSNR} is defined as:

$$\Delta_{PSNR} = 100 \times \frac{(PSNR_{avg} - PSNR_{current})}{PSNR_{avg}} \tag{6.8}$$

For example, Table 6.5 and Table 6.6 indicate the feasibility of selecting $\Delta_{FD} = 3\%$ for the `city` input sequence with a sliding window of 6 frames. Although the nominal PSNR value may vary, Table 6.5 and Table 6.6 together show how a desirable Δ_{FD} value could tradeoff both false positive and false negative detections. Finally, selection of the sliding window size can take into account the product of reconfiguration time and frame rate yielding $\lceil T_{recon} \times Framerate \rceil$, e.g. $\lceil 180ms \times 30fps \rceil = 6$ frames. Table 6.7 lists the effectiveness of using these detection parameters with a variety of input benchmarks using PEs with 5% degraded output at frame 51, QP=5.

Table 6.5: Effect of $\Delta_{FD} = 3\%$ tolerance using Failure-Free Resources for city.qcif

Frame	Δ_{PSNR}	Action	Interpretation
7	-0.32%	no change	correct
23	0.26%	no change	correct
...	...	no change	correct
47	7.53%	reconfiguration triggered	false positive†
...	...	no change	correct
70	2.01%	no change	correct

† reconfiguration is triggered

Table 6.6: Effect of $\Delta_{FD} = 3\%$ tolerance using PEs with 5% degraded output

Faulty PE	Δ_{PSNR}	Action	Interpretation
1	6.63%	reconfig. triggered at 51	correct
2	4.07%	reconfig. triggered at 51	correct
3	4.76%	reconfig. triggered at 52	correct
4	4.63%	reconfig. triggered at 52	correct
5	3.99%	reconfig. triggered at 53	correct
6	3.01%	reconfig. triggered at 55	correct
7	$< 3\%$	no change	false negative†
8	$< 3\%$	no change	false negative†

† innocuous fault below threshold, reconfiguration is not triggered.

Table 6.7: Fault detection performance ($\Delta_{FD} = 3\%$)

	Faulty PE_1			Faulty PE_2		
Sequence	Trigger Frame	Δ_{PSNR}	Interpretation	Trigger Frame	Δ_{PSNR}	Interpretation
Akiyo	51	6.54%	correct, latency = 0 frames	none	-	false negative
Carphone	52	4.81%	correct, latency = 1 frames	52	3.33%	correct, latency = 1 frames
City	51	6.63%	correct, latency = 0 frames	51	4.07%	correct, latency = 0 frames
Claire	53	3.04%	correct, latency = 2 frames	55	3.01%	correct, latency = 4 frames
Football	51	41.55%	correct, latency = 0 frames	51	16.77%	correct, latency = 0 frames

Table 6.8 summarizes the combinations of conditions under which PSNR is a reliable indicator of faults. The first row indicates that when no fault is present and tolerance is not exceeded then fault diagnosis is not invoked. The last row corresponds to the scenario whereby fault diagnosis is initiated in response to a fault detected by exceeding detection tolerance. Both of these scenarios invoke the expected response to maintain the quality objective by seeking a repair only when

needed. Conditions corresponding to the middle two rows of Table 6.8 also maintain the desired quality objective, due to the non-intrusive nature of the PURE reconfiguration process. For instance in the second row, PURE still minimizes the impact of inadvertent triggering of reconfiguration by temporarily deallocating the least priority function or reconfiguring the RS. In the third row, the failure is an *innocuous fault* in the sense that it does not manifest a degradation in signal quality sufficient to necessitate repair. In summary, PURE allows the designer to specify the tolerable range of signal degradation by selecting Δ_{FD} to allow fluctuations up to that value without triggering the diagnostic flow. Finally, even though PSNR calculation and the Reconfiguration Controller are not part of the throughput datapath and thus do not impact signal quality, handling of possible faults in these PURE components can be addressed using techniques identified in [181].

Table 6.8: Quality-oriented fault-diagnosis

Hardware Faults	$\Delta_{PSNR} > \Delta_{FD}$	**FD** asserted	**Quality objective met?**
No	No	No	Yes
No	Yes	Yes → False Positive	Yes†
Yes	No	No → False Negative	Yes††
Yes	Yes	Yes	Yes

† Small power overhead involved
†† Innocuous fault

Output Discrepancy as a Health Metric

When a health metric such as PSNR is readily available, it can be used to reduce area and power overheads. However, for applications where such health metric is not feasible, PURE can utilize CED and priority information without loss of generality. Thus, to detect hardware faults at the local DCT level instead of an entire encoder level, a periodic checking scheme is employed. Here a single RS is used which can be either a design-time spare or the least priority PE. In either case, an RS is sequentially configured with active functions of the throughput datapath to serve as a replica for discrepancy checking. A discrepancy between an active PE and RS indicates a hardware fault in

one of them, but does not indicate which one. Once suspect PEs are identified, the same diagnostic flow can be invoked that was previously described for the PSNR metric. Afterwards, the PURE diagnostic flow is initiated to analyze and isolate the faulty PEs.

When using Output Discrepancy as a health metric, PURE gives precedence to checking the highest priority PEs. For example, in the case of DCT the PE which computes the DC coefficient is prioritized first, then the PE computing the AC_0 coefficient, and so on. The choice of how frequently an RS is configured and the number of RS utilized, both affect the fault-detection latency. We will discuss in Section 6 how the fault-handling latency is improved by increasing the number of utilized RS. Both the above mentioned parameters, i.e., reconfiguration interval and number of RS, affect the power consumption. An in-depth discussion of using output discrepancy as a health metric is presented in [14] where a low area overhead estimator is used in lieu of multiple instances of the fully redundant datapath.

In summary, use of either a PSNR-based or discrepancy-based health metric can be used to initiate the PURE diagnostic flow. Nonetheless, PURE provides the designer with the freedom to choose the number of RS and the period between reconfigurations based on area, power, and fault-detection latency tradeoffs in order to meet the specific design objectives.

PURE Functional Testing as Compared to Physical Resource Testing

There are a number of distinctions between PURE and physical resource testing techniques. For instance, the STARs approach mentioned in Section 2 provides a useful and established online BIST approach to diagnosis of FPGA Logic Resources by Abramovinci, Stroud, and Emmert [67],[89]. Both techniques focus on providing fault coverage while maintaining useful throughput. However, they have significant differences including: *test and recovery granularity*, *test input vector overhead*, *support for heterogeneous resources*, *detection latency*, and *dormant fault coverage*.

With respect to test and recovery granularity, the techniques differ significantly. Both PURE and STARs can utilize CED for fault detection. STARs uses CED to compare the outputs of each fine-grained physical resource individually, whereby every Programmable Logic Block (PLB) is repeatedly reconfigured for testing against some other PLB. On the other hand, PURE employs CED at the coarse-grained application level to compare functional outputs, whereby each function is composed of an arbitrarily large number of PLBs. Thus for signal processing architectures, PURE is able to take advantage of information from the application-level, such as pipeline stage organization of the DCT core or video encoder. In the case of PSNR as a health metric, PURE provides the advantage of needing to reconfigure only when a fault is present and observable. In terms of scalability, in contrast to fine granularity BIST-style approaches which require reconfigurations proportional to the number of physical resources, PURE diagnosis flow executes linearly with respect to the number of PEs.

With respect to test input vector overhead, PURE avoids exhaustive test inputs by leveraging the throughput input data to detect discrepancies, as described above. STARs, on the other hand, requires additional inputs which function only as test vectors, but do not contribute to throughput. It employs a Test Pattern Generator (TPG) and an Output Response Analyzer (ORA) to test a block under test. STARs utilizes pseudo-exhaustive test inputs which configure every PLB to every possible logic function individually to verify correctness. While both PURE and STARs require periodic reconfiguration, PURE reconfiguration consists of only loading the bitstream for a PE which is invariant and predefined. STARs reconfigures each PLB in a vast range of arrangements which must be stored separately or created dynamically. However, this does allow STARs to locate and remap the fault at the finest possible granularity. This conserves resources which can be recycled, although contemporary reconfigurable devices have a vast number of resources available. Nonetheless, this does allow STARs to provide dormant fault coverage even if the PLB is not active. In PURE, dormant faults are expunged after the region is configured for comparison by the

diagnosis flow.

With respect to support for heterogeneous resources, PURE's use of functional testing can be advantageous. For instance, considering that many commercial FPGAs provide an abundance of dedicated functional units embedded in the fabric such as hardware multipliers. Since PURE uses functional performance for both PSNR and CED based fault detection, testing of the embedded resources such as a Xilinx DSP48 multiplier become intrinsic in the technique. On the other hand, resource-oriented tests must seek out special-purpose pseudo-exhaustive tests of these heterogeneous resources to avoid combinatorial explosion of the input space.

With respect to detection latency, exhaustive resource testing exhibits a detection latency proportional to the number of PLBs rather than number of PEs. For a $N \times N$ array of PLBs, the expected value of detection latency for STARs is $\frac{N^2}{2} \times t_{test}^{PLB}$ where t_{test}^{PLB} denotes the testing time of a PLB plus overheads incurred by stopping the clock to capture the register states. Use of a hybrid functional CED technique to detect faults and then STARs to diagnosis and recover from them has been proposed as an enhanced version [182]. For PURE, the expected value of detection latency varies linearly with the number of PEs. More precisely, an upper bound on the diagnosis time is defined in terms of the maximum slack-update iterations required to isolate N_d number of faulty nodes in a network of N nodes employing a single RS, and is given by:

$$N_{sup,max} = 1 + \sum_{s=N-N_d}^{N-1} s \tag{6.9}$$

For example, given a network of size $N = 8$ and $N_s = 1$, the maximum number of slack updates occur in the case when PE_7 and PE_8 are faulty as depicted in Fig. 9. Thus, $N_{sup,max} = 1 + 7 + 6$. The constant term 1 is added to include the reconfiguration required to identify a healthy slack. Fig. 6.17 shows the upper bound on diagnosis latency using $N_s = 1$. Fig. 6.18 shows the diagnosis

latency when using two slacks for a network of size $N = 8$. Although, an increase in number of nodes results in increased diagnosis latency due to the reconfigurations involved, the number of defective nodes impact the latency more significantly. To diagnose a single defective node with $N_s = 2$, as few as one slack update is required as compared to the previous case requiring a maximum of 8 slack updates when only one slack was employed.

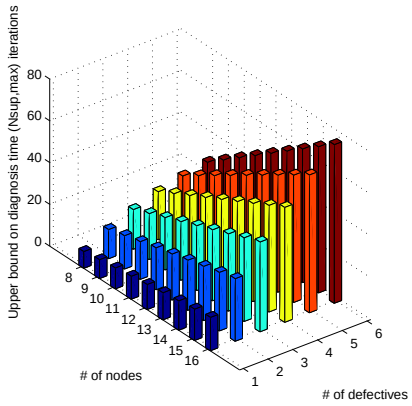


Figure 6.17: Diagnosis latency of the PURE approach for $N_s = 1$

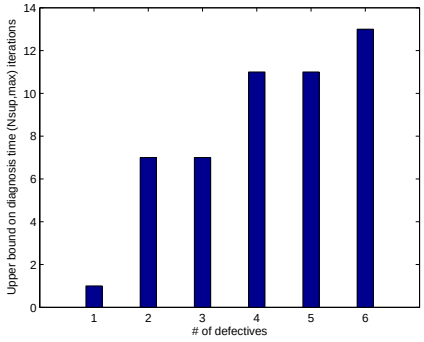


Figure 6.18: Diagnosis latency of the PURE approach for $N_s = 2$, $N = 8$

Table 6.9: Configuration bitstream sizes in DCT core

Function	PRR Location	.bit Size
f_{DC}	SLICE_X54Y224:SLICE_X71Y255	32KB
f_{AC0}	SLICE_X54Y192:SLICE_X71Y223	35KB
f_{AC1}	SLICE_X54Y160:SLICE_X71Y191	34KB
f_{AC2}	SLICE_X54Y128:SLICE_X71Y159	35KB
f_{AC3}	SLICE_X54Y96:SLICE_X71Y127	34KB
f_{AC4}	SLICE_X54Y64:SLICE_X71Y95	36KB
f_{AC5}	SLICE_X54Y32:SLICE_X71Y63	37KB
f_{AC6}	SLICE_X54Y0:SLICE_X71Y31	34KB

Table 6.9 lists the configuration bitstream sizes for various PEs in DCT core which can be used to assess the configuration memory size requirement. The following factors are involved in the reconfiguration flow, and hence add to the overhead of the diagnostic provision in PURE approach.

PRR Size: For Virtex-4 device, the minimum PRR height that can be defined is 16 CLBs [165] while the maximum height can span an entire column in the chip. To effectively utilize the PRR capacity, the resource utilization of the mapped function should also be considered when choosing the PRR size. For example, each PRR should have a sufficient number of LUTs, FFs, and DSP multipliers to implement a DCT-coefficient computation function in the DCT core.

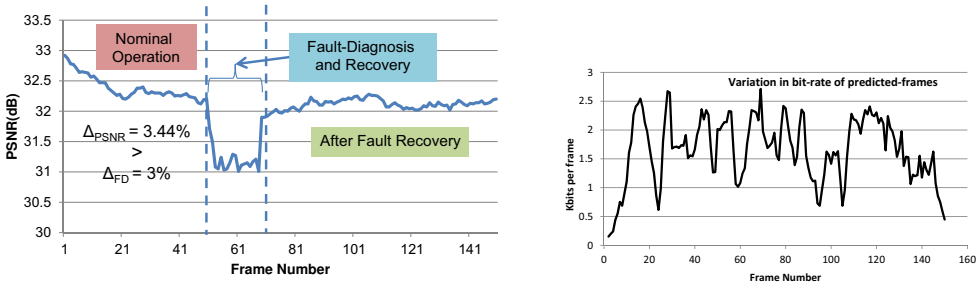
Number of PRRs (M): The total number of reconfigurable partitions defined at design-time depend upon number of functions, throughput requirements, fault-handling capacity to multiple failures, and desired diagnostic latency. Fault-detection and diagnosis latency can be improved by utilizing more PEs for the comparison purposes at runtime.

External Reconfiguration Memory Size: Each PRR can perform the computation of a function while each function mapping generates a partial reconfiguration bitstream. To realize full mapping capability at runtime so that any function can be mapped to any PE, as many as $N \times M$ number of configurations equivalent memory is needed. Thus, the compact-flash memory size requirement increases significantly with both N and M.

On-chip Reconfiguration Memory Size: For a tractable number of nodes such as 8, the on-chip configuration memory size requirement can be fulfilled with today's FPGAs. However, the on-chip memory of FPGAs may not scale well for the increased the number of PEs. In such a scenario, a bitstream relocation approach[183][110] can benefit in saving the memory requirement. In [183], the authors reported a 50% reduction in number of partial bitstreams in a software defined radio prototype while a 79.4% saving of the overall bitstream storage size was achieved in [110] by exploiting the relocatable modules.

Experimental Results

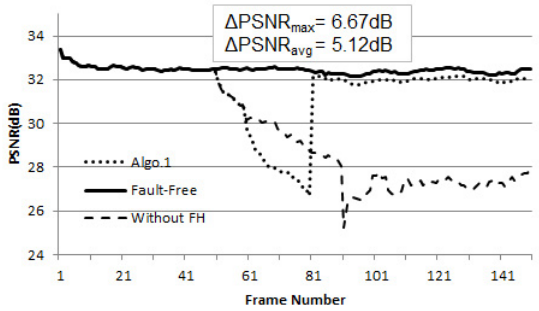
To assess the resilience and power consumption of the PURE algorithm, case studies were evaluated with various benchmarks, using either PSNR or Output Discrepancy as a health metric.



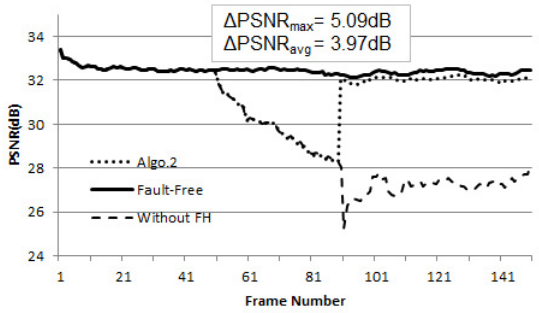
(a) PSNR of `silent.qcif` video sequence

(b) Bit-rate of `silent.qcif` video sequence

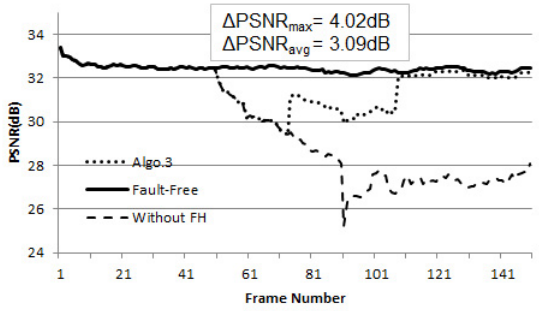
Figure 6.19: PSNR and bit-rate of the encoder employing PURE



(a) Fault-handling using Algorithm 1: Divide-and-Conquer



(b) Fault-handling using Algorithm 2: FaDReS



(c) Fault-handling using Algorithm 3: PURE

Figure 6.20: Operational examples of the three algorithms

Case Study-1: Prioritized elements of the DCT core

To demonstrate the effectiveness of the proposed approach, first consider the case of H.263 video encoder's DCT module. The 8×8 DCT is computed by 8 PEs. Each PE performs the 1D-DCT of a row of input pixels to produce an output coefficient. For example, PE_1 computes the DC-coefficient from 8 pixels in a row of frame memory. In the current prototype to evaluate PURE approach, the video encoder application is run on the on-chip processor. All the sub-blocks except the DCT block are implemented in software, the later being implemented in hardware. The image data of video sequences is written by the processor to the frame buffer. In order to facilitate 2-D DCT operation, the frame buffer also serves as transposition memory and is implemented by Virtex-4 dual port Block-RAM. Upon completion of the DCT operation, it is read back from the frame buffer to the PowerPC through the Xilinx General Purpose Input-Output (GPIO) core. By the pipeline design of the DCT core, the effective throughput of the DCT core is one pixel per clock. Internally, the PEs utilize DSP48 blocks available in Virtex-4 FPGAs. A 100MHz core operation can provide maximum throughput 100M-pixels per second while in order to meet the real-time throughput requirement for 176×144 resolution video frames at 30 frames per second, the minimum computational rate should be 760K pixels per second. The PSNR computation time is much longer than that consumed by PEs processing data stream in parallel, i.e., 0.25msec per frame. It is worth mentioning, however, that a failure to meet real-time deadline in PSNR computation due to a slow speed processor will only impact the fault-detection/handling latency rather than the computational throughput of the concurrently operating PEs-array implemented in a hardware fabric.

The priority of functions is naturally in descending order as the DC-coefficient contains most content information of a natural image. These 8 PEs in the processing throughput datapath are covered by the proposed resilience scheme. For this purpose, depending upon area/power margin available,

RSs are created at design-time or generated at runtime considering the priority of functions. As shown in Fig. 6.20, fault-handling is performed at runtime with a small quality degradation during diagnosis process. The diagnosis time of Algorithm 1 is very short, however, this greedy approach incurs quality degradation during fault-handling process. The quality degradation during fault-handling process is improved in Algorithm 2 at the cost of some diagnosis latency. Algorithm 3 provides the best availability of the system during fault-handling and the PSNR is maintained above 29.5dB. Fig. 6.19 shows the PSNR and bit-rate of the video stream from an operational encoder before, during, and after fault-handling using the PURE approach.

During diagnosis, PURE can achieve higher useful throughput than alternative approaches due to escalation of healthy resources to the top priority functional assignments. Fault-handling results with a video encoder show that average PSNR in PURE's case is only 3.09dB below that of a fault-free encoder, compared to a divide-and-conquer approach which incurs average PSNR loss of 5.12dB during the fault diagnosis phase. This metric provides a useful indication of quality during refurbishment. The PURE approach maintains throughput by retaining viable modules in the datapath while divide-and-conquer does not take them into account. Moreover, latency of diagnosis phase can be reduced by employing multiple dynamic slacks. For instance, a 90% of diagnosable conditions can be identified in a single reconfiguration using $N_s = 3$ slacks while $N_s = 1$ slack identifies only 50% diagnosable conditions. A 90% CPDC is achieved in more than 3 testing arrangement instances when using a single slack. Compared to a static topology scheme where PEs arrangement is fixed at design-time, diagnosability can be increased from a single defective node to six defective nodes using as few as $r = 4$ reconfigurations and $N_s = 2$ slacks. In general, the diagnosability at the completion of PURE's algorithm is $N - 2$ after every possible pair combination is evaluated since at least one healthy pair is necessary to eliminate suspect status.

Case Study-2: Fault Resilience of a Multi-PE Design

Next, to evaluate the PURE approach to applications which do not possess a PSNR-like health metric, we consider AES [184] in the context of the proposed fault-diagnosis methodology using a verilog core [185]. For this purpose, the encryption module of 128-bit AES is synthesized and implemented in Xilinx ISE 13.4 development environment for Virtex-7 `xc7v2000t` device. SA faults are injected in the simulation model of circuit generated by the Xilinx Xtool flow. We utilized our previously developed *Fault Injection and Analysis Toolkit (FIAT)* [80] which invokes various commands of the Xilinx flow to study fault behavior. Then, the circuit is evaluated using test inputs. The test outputs, corresponding actual fault-free outputs, and the outcome in terms of actual AES functionality are listed in Table 6.10. For the given case of 8 inputs, a total of 4 outputs being faulty are observed.

Table 6.10: Fault impact in 128 AES Computational FE

Actual Output	True Output	Test Outcome
66e94bd4ef0a2c3b884cfa59ca342b2e	*66e94bd4ef8a2c3b884cfa59ca342b2e*	*incorrect*
3ad78e726c1ec02b7ebfe92b23d9ec34	3ad78e726c1ec02b7ebfe92b23d9ec34	correct
45bc707d2968204d88dfba2f0b0cad9b	*45bc707d29e8204d88dfba2f0b0cad9b*	*incorrect*
161556838018f52805cdbd6202002e3f	161556838018f52805cdbd6202002e3f	correct
f5569b3ab626d11efde1bf0a64c6854a	*f5569b3ab6a6d11efde1bf0a64c6854a*	*incorrect*
64e82b50e501fbd7dd4116921159b83e	64e82b50e501fbd7dd4116921159b83e	correct
baac12fb613a7de11450375c74034041	baac12fb613a7de11450375c74034041	correct
bcf176a7ea2d8085ebacea362462a281	*bcf176a7eaad8085ebacea362462a281*	*incorrect*

To analyze the latency, area, and power consumption of the fault-resilient architecture of the AES module, we used Xilinx ISE 9.2i for synthesis and implementation flow. The utilization summary for the design implemented on a `xc4vfx60-12ff1136` device is listed in Table 6.11. For the synthesized design, minimum clock period is 1.821ns (Maximum Frequency 549.058MHz). The size of each partial reconfiguration bitstream file is 112.8KB. Fig. 6.21 shows the floorplan of AES core. This allows the PURE approach to occupy only $\frac{1}{N}$ area overhead for N PEs.

Table 6.11: Utilization summary of the AES design

Logic Resource	Utilization		Capacity of a PRR
	PE	Reconfigurable PE	
Number of Slices	416	1021	1024
Number of Slice Flip Flops	625	1778	4096
Number of 4 input LUTs	726	1236	4096
Number of FIFO16/RAMB16s	16	16	16

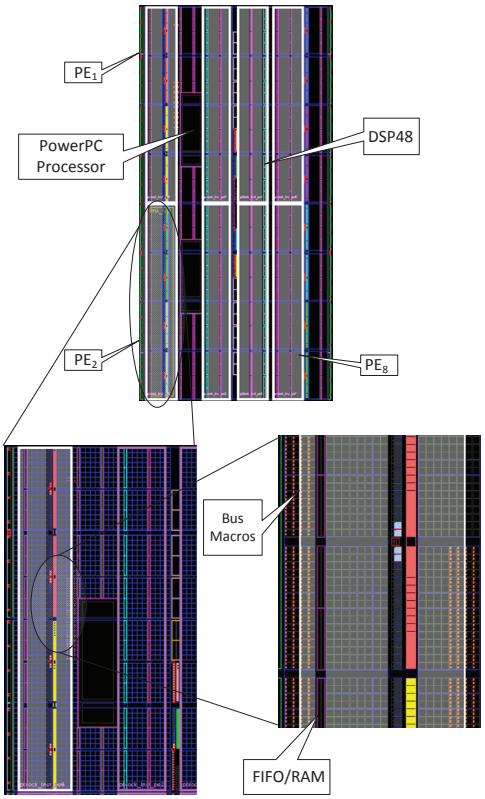


Figure 6.21: Floorplan of the AES core for Virtex-4 chip

Energy Duty Cycle

Time-Dependent Dielectric Breakdown (TDDB) and Electromigration (EM) are two significant causes of permanent faults over the device lifetime [186]. To quantify the *survivability* of the system employing the PURE fault-handing flow, the fault detection, diagnosis, and recovery times are considered here. The *availability* is generally defined in terms of Mean-Time-To-Failure (MTTF) and Mean-Time-To-Repair (MTTR). The impact of radiation and aging-induced degradation on reliability of FPGA-based circuits has been analyzed by authors in [186], [187], [17].

In this analysis, we use the TDDB failure rate of 10% LUT per year and EM failure rate of 0.2% per year as demonstrated in [186] for MCNC benchmark circuits simulating their 12-year behavior. Considering a DCT core, 312 utilized LUTs in a PE spanning one Partial Reconfiguration Region (PRR) exhibiting a 10.2% failure rate means 32 LUTs fail per year. If the failure rate is uniformly distributed over time, then a worst case scenario would correspond to a MTTF of 11 days between LUT failures.

The MTTR is the sum of times required for fault detection, diagnosis, and recovery. To assess the fault detection latency, faults are injected into the DCT module at frame number 50 of the `news.qcif` video sequence [3]. As a result, the PSNR drops at frame number 59. Thus, the fault detection time is 0.3 seconds for a 30fps frame rate. For a system of $N = 8$ PEs, the latency of fault-diagnosis can be computed by using eq. 6.6. Using one slack, the maximum cost is 196 frames or 6.5 second for a frame rate of 30 fps. Given the diagnosis data, the time to identify faulty nodes is negligible as the on-chip processor operating at 100 MHz clock rate can mark faulty nodes in very short time once the syndrome matrix is computed. Similarly, time required for 8 reconfigurations during fault recovery is 1.6 seconds. Thus, total time to refurbishment for this particular example is 8.4 seconds. With these values of MTTF and MTTR, the PURE's availability is 99.999%. Moreover, significant throughput is maintained during fault-diagnosis phase as evident

by the minimum values of PSNR in Fig. 6.20. Thus, the impact on signal quality even during the period of unavailability is minimal.

Next, we analyze the dynamic power duty cycle of the proposed scheme. An instance of the simple DCT module consumes 72mW dynamic power. However, after adding the fault-resilience overhead, the consumed dynamic power is 142mW. On the other hand, a TMR arrangement would consume about 216mW dynamic power in addition to the voter, during the 12-year mission-lifetime. By tackling aging-induced degradation failures in FPGAs, the availability is improved from 6.027% for TMR to 99.999% for PURE given pessimistic device failure rates. This average availability measure for TMR is based upon failure of two TMR paths without recovery, as a system failure may occur in the worst case upon incidence of the second fault. Since conventional TMR provides no repair mechanism, in the worst case the system becomes unavailable upon occurrence of a second failure, as the correct functioning datapath cannot be discerned by majority voting. Furthermore, the PURE arrangement consumes only 33% of TMR configuration power for 99.999% of the mission-period. Meanwhile, it consumes 65.7% of TMR arrangement for only 0.001% mission. A second case study with an AES encryption core implemented on a Xilinx Virtex-4 FPGA indicates detection and recovery of repeated stuck-at faults using diagnosis-by-comparison at the module-level while requiring only $\frac{1}{N}$ area overhead for N PEs when $N_s = 1$ slack is used.

CHAPTER 7: CONCLUSION

With reduced feature sizes and power consumption, future nano-scale semiconductor devices are attractive candidates for signal processing platforms to meet computational demands of current and future video applications. However the reliability of high density devices decreases significantly when used for computationally-intensive complex signal processing tasks. Thus, autonomous fault-handling becomes highly desirable to sustain performance in a seamless fashion from the viewpoint of the middleware and application software. The techniques developed in this work are summarized in the following section. Next, the scope and limitations of the current work are identified in Section 7.2, and then possible future directions are discussed in Section 7.3. Finally, Section 7.4 concludes the dissertation by envisioning a path for the road ahead in a broader perspective.

Technical Summary

Figure 7.1 shows various evaluation metrics and the techniques developed in this dissertation to meet the objectives set forth in Chapter 1. Ideally, a fault-tolerance scheme should have a very small, δ area-overhead. However, traditional schemes like TMR arrangement incur significant area cost. A FaDReS arrangement operates with almost uniplex area requirement by employing the reconfiguration strategy for fault-diagnosis. Similarly, a fault-detecting method is favorable if it incurs minimal throughput degradation. The traditional redundancy-based CED method reduces the throughput by half to enable comparison-based detection. In the hCED scheme proposed herein, the throughput degradation is improved by multiplexing the hardware fabric for primary operation and a down-sampled fault detection phase. Furthermore, PURE maintains the throughput of the datapath by computing the PSNR off the critical path in concurrent with main operation.

Metrics	From existing achievements towards ideal case	Results obtained from the dissertation
Area overhead	Reduce from 200% in *TMR* towards 0%+δ for uniplex plus control	Only 65.7% of TMR during fault-handling and 34% of TMR during fault-free operation in FaDReS
Throughput Reduction	Reduce from 100% in CED towards 0%+δ for uniplex plus estimator	• 11% in temporal hCED by multiplexing the fabric to compute the anticipated value of output • 2.79% time utilization of the embedded processor for PSNR calculation in PURE's fault detection phase
Detection Latency	Less-than-linear increase with number of resources in BIST to instantaneous fault detection	167msec latency of detecting operationally significant faults in PURE
Isolation Latency	Avoid taking the device offline as in *Offline Testing*, rather maintain throughput during fault isolation	Maintain PSNR degradation < 3% during a 60 frame fault isolation interval
Recovery Time and Certainty	Reduce from non-deterministic number of reconfigurations in *Evolutionary* approaches to a bounded recovery latency	The expected value of number of reconfiguration iterations to a complete recovery = 2.54 in FaDReS
Recovery Quality	Avoid catastrophic failure in presence of multiple faults to achieve a graceful degradation strategy	Sustain throughput despite multiple hardware failures in SCDR, DRFI, Online MOGA, FaDRes, and PURE

Figure 7.1: The techniques developed herein to meet evaluation criteria

Starting with the first objective in Figure 7.1, area management techniques for fault handling in reconfigurable logic devices were presented in Chapter 3. These avoid resource testing of test vectors and instead utilize discrepancy information during normal throughput computation. The fault coverage achieved with these techniques spans the utilized logic resources. For instance, the SCDR adaptive fault-handling scheme configures the throughput datapath in reconfigurable fabric based systems. The improved SNR as a result of the recovery scheme compared to that of a faulty system demonstrates the effectiveness of the approach. We evaluated the scheme using a typical application that is decomposable into distinct pipelined stages with favorable results by metrics of Signal-to-Noise Ratio (SNR) and PSNR.

To efficiently manage the available area for fault-detection purposes, two forms of the CED method of fault detection in FPGA-based designs were introduced. The *spatial heterogeneous CED* form exhibits reduced resource requirements over a conventional CED technique. Thus, area and power

are conserved using the proposed approach at the cost of a negligible fault detection latency overhead. The *temporal heterogeneous CED* forms error detection capability of fault coverage includes permanent faults in logic resources, in addition to transient faults. Moreover, the temporal error detection form has uniplex area requirement avoiding redundancy in the resources. It has the capability to manifest permanent faults due to diverse inputs. These results are significant contribution in the sense that fault coverage is enhanced with negligible resource overhead at the cost of reduced throughput.

A fault handling mechanism using Amorphous Slack is introduced which has advantages of continuous throughput with small degradation and low area overhead. Dynamic PR is used with hardware modularity to provide autonomous capability for survivable systems. Experiments with video coding and image processing applications indicate that fault resilience is achievable in an area efficient manner using Amorphous Slack. For example, TMR will require 24 modules for 8×8 DCT computation and the fault capacity would be limited to errors in only one voting path. However, the Amorphous Slack approach allows additional modules during normal operations, and can handle even the case when 6 out of 8 modules are faulty. Thus, compared to the TMR scheme, the area and power requirements are about one third, yet fault tolerance is improved. Moreover, fault-handling can be adjusted by the DSP circuit designer based upon the tradeoff desired between detection latency and the area overhead incurred.

In another adaptive area-management technique, a pool of hardware configurations for a reconfigurable platform is generated at design-time by utilizing distinct arrangement of hardware resources. Once faults occur affecting multiple circuit realizations, the PageRank algorithm is used to identify the most functional realizations. The experiments indicate that the approach is effective at identifying the correct configuration in a fraction of the comparisons needed by unguided search, thereby offering considerably improved throughput. In addition, *graceful degradation* is realized by promoting the bitstreams in situations where all configurations are faults-affected.

Next, a throughput-driven runtime resource configuring scheme to realize soft-resiliency in self-repairing computational platforms for signal processing is presented. A health metric-based feedback method is used by the multi-objective online evolution to dynamically adapt the processing blocks to achieve the desired levels of power and quality. The scheme is validated by implementation on a commercial off-the-shelf Xilinx Virtex FPGA to validate the feasibility of a fault-tolerant and energy-efficient design. Moreover, the scheme is not dependent upon the technology model of a specific device. Nonetheless, a dynamic reconfiguration capability of the devices is essential to implement the proposed fault handling flow.

To improve the fault-handling latency of the evolutionary process, FHME is developed which is capable of accommodating hard faults using a reconfiguration strategy while maintaining useful throughput during recovery. Input signal characteristics are exploited to intelligently manage the computational resources with the objective of power efficiency, graceful degradation, and recovery time. In addition to leveraging data parallelism, the priorities are identified in sub-modules of an ME engine and utilized accordingly to recover from fault scenarios. The FHME core prototyped on a Xilinx Virtex-4 device demonstrates power reduction and resilience in video datasets available from [3]. The concept of dynamic adaptation is evaluated by considering the tradeoff between QoS, power consumption, and reliability levels.

Moving on to the last row of Figure 7.1, the issue of recovery quality is addressed. During diagnosis, PURE can achieve higher useful throughput than alternative approaches due to escalation of healthy resources to the top priority functional assignments. Fault-handling results with a video encoder show that average PSNR in PURE's case is only 3.09 dB below that of a fault-free encoder, compared to a divide and conquer approach which incurs average PSNR loss of 5.12 dB during fault diagnosis phase. This metric provides a useful indication of quality during refurbishment. The PURE approach maintains throughput by retaining viable modules in the datapath while divide and conquer and randomized pairing do not take them into account. Moreover, latency of

diagnosis phase can be reduced by employing multiple dynamic slacks. For instance, a 90% probability of diagnosis can be achieved in a single reconfiguration using $N_s = 3$ slacks while $N_s = 1$ slack provides only 50% probability of diagnosis. A 90% probability of diagnosis is achieved in more than 3 testing arrangement instances when using a single slack. Compared to a fixed topology scheme, diagnosability can be increased from a single defective node to six defective nodes using as few as $r = 4$ reconfigurations and $N_s = 2$ slacks. In general, the diagnosability at the completion of PURE's algorithm is $N - 2$ after every possible pair combination is evaluated. System's availability and survivability, which are important characteristics of mission-critical systems in presence of environmental-effects/aging-induced permanent faults, are significantly improved. By tackling aging-induced degradation failures in FPGAs, the availability is improved from 6% for TMR to 99.999% given pessimistic device failure rates.

Compared to a divide-and-conquer approach which incurs peak PSNR loss of 6.67dB during the fault diagnosis phase, PURE performance of a video encoder achieves peak PSNR degradation of only 4.02dB, when subjected to identical video inputs and failure conditions. By tackling aging-induced degradation failures in FPGAs, the availability is improved to 99.999% even for pessimistic device failure rates. Thus, detection and isolation latency, recovery latency, and recovery quality are improved by innovations proposed in this dissertation.

A priority-aware deterministic flow fault-handling algorithm, FaDReS is developed which uses PSNR as an indicative measure of the hardware's health. The experimental testing of the FaDReS algorithm shows advantageous results for fault handling scenarios. The number of iterations in fault isolation phase is bounded by the number of PEs and RS condition. The PSNR degradation is minimal throughout this phase by fully utilizing input signal characteristics to reduce DCT size. The priority of functions is taken into account to achieve the best possible solution in a fault scenario. Although the FaDReS finite state machine is a critical component of the proposed scheme, some other means of software fault tolerance may be employed for the PowerPC processor [181]

which is responsible for sequencing the fault handling mechanism. Although a video encoder was considered as a case study, a broader range of applications could be feasible. In particular, in absence of a uniplex health metric (such as PSNR), the designer can tradeoff periodic temporal CED or spatial CED redundant computations based on throughput/cost/reliability constraints. Alternatively, the slack can be periodically configured in a round-robin manner to check the correctness of PEs as demonstrated in [18].

To further improve the partial throughput during recovery period PURE scheme is proposed which also employs a health metric in the feedback loop. PURE provides an adaptive approach to fault-handling for meeting survivability objectives using dynamic reconfiguration. Dynamic redundancy realizes autonomicity while gracefully maintaining a defined quality of service measure within area-resource, power, and energy constraints. PURE achieves these objectives at reduced area and power overheads compared to static redundancy schemes by adapting a uniplex instance of the datapath when aberrant behavior occurs. A uniplex configuration is sufficient if a signal-to-noise metric is known, as well as in applications which possess a readily correlated metric to identify anomalous behavior.

Scope and Limitations

While PURE provides an adaptive dynamic reconfiguration approach to achieve survivability with benefits in terms of most metrics, its scope of applicability and limitations are described here. The foremost is the availability of a reconfigurable fabric. Dynamic reconfiguration of redundancy permits autonomous operation while maintaining a defined quality measure within area-resource, power, and energy constraints. PURE achieves these objectives at reduced area and power overheads compared to static redundancy schemes by adapting a uniplex instance of the datapath when aberrant behavior occurs. A uniplex configuration is shown to be sufficient for applications such

as DCT when a signal-to-noise metric such as PSNR is available. Yet without loss of generality, PURE is suitable for any application which possesses a definitive condition identifying anomalous behavior such as output discrepancy using diagnosis-by-comparison.

Even though logic resources are focused on for fault-coverage, interconnect resources are covered to some degree. In particular, within the routing permutations available for the pre-defined RS regions mapped over the PRRs of the reconfigurable fabric. The fault coverage provided includes logic resources as well as routing resources as their performance is intrinsic to the observed quality metric. The malfunctioning of any of them will result in the utilizing PE to be flagged as faulty, and then its assigned function is moved to another area in the chip only if it is found to exhibit a sufficient operational priority on the output quality. This self-organizing hardware architecture maintains energy efficiency and quality under various operating conditions by sacrificing non-critical computations based on input signal characteristics and escalating critical tasks to healthy computational resources.

For reconfigurable fabrics, an autonomous soft-resilience approach can be advantageous to the tradeoffs of accuracy and energy efficiency especially if cessation of throughput is acceptable to the application. For example, a multi-objective GA approach is promising in solving such large search space problems using the proposed guidance function along the pareto front if it can operate within a sufficient time window to perform the search. The proposed scheme performs well for a synthetic node case study as well as SVM and DCT computations. The recovery results demonstrate self-healing capability, as well as power efficient circuits with provision of the adaptive resource escalation approach. An interesting future consideration would be to develop a scheme for priority estimation at runtime for other applications where task priority information is not known a-priori.

In summary, the proposed techniques appear to be preferable by several metrics, for applications

having a reconfigurable logic fabric as compared to previous approaches for such domains. For use in Application-Specific ICs, however, large-scale reconfiguration is not possible, and traditional fault-handling techniques such as static TMR are recommended. These are still preferred when reconfiguration is not an option or fault-masking capability is desired in order to mask even a single error to propagate through the output.

Table 7.1: A summary of the dissertation and lessons learned

What worked well?
•Uniformly partitioned SCDR pipeline •Full search based FHME •Multi-objective GA to find pareto solution in quality-energy space •Xilinx platform for dynamic reconfiguration and power estimation •Video encoder application to demonstrate graceful degradation •PSNR-based health metric to trigger fault-handling
What did not work as well?
•Vendors proprietary file formats and tool flow support for fault injection •Fault-handling latency of the evolvable hardware scheme •A divide and conquer approach to isolate faulty modules (throughput) •Reconfiguration approach to mitigate soft errors (latency)
Lessons learned
•GAs are better at large scale problems •Deterministic flows (FaDRes, PURE) perform good for rapid fault recovery when using comparison diagnosis models •Reconfiguration is essential to mitigate permanent faults
What challenges remain?
•Extending SCDR to a Processor Pipeline •Designing the datapaths to support architectural adaptations of the FHME •Extending the multi-objective approach to large systems •Extending the reconfiguration concepts to ASICs to mitigate aging, PV and supporting NTV operation •Memory Errors

Table 7.1 summarizes the main points of the dissertation along with lessons learned and future directions. The schemes developed herein require dynamic reconfiguration capability of the un-

derlying platform to dynamically adapt the architecture according to runtime characteristics. It may be noted that the method concentrates on local permanent damage rather than soft-errors which can be effectively mitigated by *Scrubbing* [11]. Scrubbing for reconfigurable devices has been addressed extensively in the literature, including [10],[76], and [188], and it provides a suitable approach to narrowly-focused transient soft-errors which would not require the robustness nor complexity of the techniques developed herein. Especially the need to maintain throughput which is typically disabled during scrubbing.

Future Directions

The research presented in this dissertation can serve as a unified framework to build advance schemes in order to enable breakthroughs and to pursue a diverse set research directions for using autonomous reconfiguration to increase reliability. Similar to the work developed herein is made possible by previous research efforts, the new horizons of technology breakthroughs call upon current research efforts. The work presented in this dissertation can be extended in following ways:

Area Management Techniques:

SCDR: Although, the FIR case study is a very regular circuit, there is no loss of generality as long as the circuit can be fitted into various PRR stages. A future extension can be the development of a fault-tolerant pipelined microprocessor core using the SCDR scheme.

hCED: The latency of fault detection in spatial mode may be improved by randomly checking some of the coefficients in the kernel computation. An important area of future work is the derivation of necessary fault-free conditions for a generic FE design.

DRFI: An interesting direction can be analyzing the effect of varying the granularity of diagnosis

by using the PR model developed in [189]. Also, the TMR model of diagnosis can be helpful to accelerate fault-isolation.

Soft-Resilience Using An Online Multi-Objective GA:
The objective function can incorporate process-variation models and feature size characteristics to extend the scheme beyond FPGAs. The reconfiguration capability in full-custom ASICs could be realized by introducing some programmable logic and routing components.

Health Metric Based Dynamic Resource Allocation:
Although the proposed scheme is evaluated for FS-based ME, energy can be reduced with other parallelizable architectures. Another future direction is extending the proposed technique of exploiting free APEs by considering higher resolutions, higher frame rates, advanced motion models like in H.264 or HEVC which have much higher computational workload demands.

Future work can be to extend these schemes to achieve fault-resilience in general multiprocessor networks which have inherent reconfigurablity, although the freedom of task mapping and increased granularity lessens the focus on maintaining throughput. Another main branch of the work presented in this dissertation can be to extend the Resource Escalation approach to accommodate inter/intra-die process variation and voltage scaling, which can adaptively achieve reliable computation at low power consumption.

The Road Ahead

As the Moore's law sustains in the near future, we can expect future chips with a very large number of transistors. In addition, the reconfigurable fabric is typically present in heterogeneous systems. Many applications are being mapped to reconfigurable fabric for energy efficiency of computation, and video applications are good candidates being highly data and computation-intensive. We

think it is important to tackle hardware issues such as process variations, power consumption, and hardware faults, etc., so that they can be handled at the architectural and algorithmic level. A possible solution is by designing the datapaths as adaptable as possible in order to realize the runtime feasibility of adding or removing computational elements in a given environment.

Other directions reach beyond traditional digital design paradigms. For example, intelligent self-healing capability is desirable in micro-electronics based systems which can be achieved through more advanced biologically-inspired design paradigms which are in their infancy for digital implementation. Adaptive designs seek to increase sustainability of circuit operation when subject to aging-induced degradation which is increasingly prominent with reduced feature size.

The concept of health metric based dynamic architectural adaptations may be extended to realize a Metamorphic Data Encoding (MDE) paradigm. For instance, the number of redundant bits in ECC codeword can be dynamically adjusted based upon runtime conditions. For power efficiency purposes, the computation block can be run at error critical voltage in the vicinity of NTV. Then, using health metric to instantaneously adapt codeword width to be just sufficient to maintain quality by power gating the ECC bits can result in energy savings. So, instead of adapting device conditions when soft errors occur or metamorphism of architecture (FaDReS), Metamorphic Data Encoded representation is inherently robust to operationally significant errors.

Metamorphic Data Encoding can be a good candidate scheme to mitigate aging by introducing some ECC bits on demand. By adapting the number of ECC bits dynamically, current operating conditions of CUT intrinsically select the amount of data redundancy needed to maintain the desired quality under current conditions. In a broader perspective, the MDE can be thought of a way to adapt what the meaning of bits represent as necessary to maintain quality. In addition, trading precision for resilience, or other novel ways that a data coding could self-adapt in a Metamorphic Computational Ecosystem seems an interesting future research direction to pursue.

LIST OF REFERENCES

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[2] TNT, "Video test sequences, institute for information processing, leibniz university of hannover," Retrieved on May 26, 2013 [Online] `ftp://ftp.tnt.uni-hannover.de/pub/svc/testsequences/`.

[3] Trace, "Video trace library: YUV 4:2:0 video sequences," Retrieved on January 20, 2012 [Online] `http://trace.eas.asu.edu/yuv/`.

[4] J. A. Blackard and D. J. Dean, "The forest covertype dataset," http://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.info.

[5] R. Hyman Jr., K. Bhattacharya, and N. Ranganathan, "Redundancy mining for soft error detection in multicore processors," *Computers, IEEE Transactions on*, vol. 60, pp. 1114 –1125, Aug. 2011.

[6] J. Emmert, C. Stroud, and M. Abramovici, "Online fault tolerance for FPGA logic blocks," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, pp. 216 –226, Feb. 2007.

[7] K. Paulsson, M. Hubner, and J. Becker, "Strategies to on-line failure recovery in self-adaptive systems based on dynamic and partial reconfiguration," in *First NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, June 2006.

[8] W. Rao, C. Yang, R. Karri, and A. Orailoglu, "Toward future systems with nanoscale devices: Overcoming the reliability challenge," *Computer*, vol. 44, pp. 46–53, Feb. 2011.

[9] SPP1500, "Dependable embedded systems," [Online] http://spp1500.itec.kit.edu.

[10] M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K. LaBel, M. Friendlich, H. Kim, and A. Phan, "Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis," *Nuclear Science, IEEE Transactions on*, vol. 55, pp. 2259–2266, Aug. 2008.

[11] P. Ostler, M. Caffrey, D. Gibelyou, P. Graham, K. Morgan, B. Pratt, H. Quinn, and M. Wirthlin, "SRAM FPGA reliability analysis for harsh radiation environments," *Nuclear Science, IEEE Transactions on*, vol. 56, pp. 3519 –3526, Dec. 2009.

[12] N. Imran and R. F. DeMara, "A fault-handling methodology by promoting hardware configurations via pagerank," in *Presentations at the ReSpace/MAPLD Conference*, (Albuquerque, NM), 2011.

[13] N. Imran and R. F. DeMara, "Cyclic NMR-based fault tolerance with bitstream scrubbing via Reed-Solomon codes," in *Presentations at the ReSpace/MAPLD Conference*, (Albuquerque, New Mexico), Aug. 2011.

[14] N. Imran and R. F. DeMara, "Heterogeneous concurrent error detection (hCED) based on output anticipation," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, pp. 61–66, Dec. 2011.

[15] N. Imran and R. F. DeMara, "A self-configuring TMR scheme utilizing discrepancy resolution," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, pp. 398 –403, Nov. 30 -Dec. 2 2011.

[16] A. Stoica, D. Keymeulen, R. Zebulum, M. Mojarradi, S. Katkoori, and T. Daud, "Adaptive and evolvable analog electronics for space applications," in *Proceedings of the 7th inter-*

national conference on Evolvable systems: from biology to hardware, ICES'07, (Berlin, Heidelberg), pp. 379–390, Springer-Verlag, 2007.

[17] C. Bolchini and C. Sandionigi, "Fault classification for SRAM-Based FPGAs in the space environment for fault mitigation," *Embedded Systems Letters, IEEE*, vol. 2, pp. 107 –110, Dec. 2010.

[18] R. F. DeMara, K. Zhang, and C. A. Sharma, "Autonomic fault-handling and refurbishment using throughput-driven assessment," *Applied Soft Computing*, vol. 11, pp. 1588–1599, March 2011.

[19] R. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's Law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, pp. 253–266, Feb. 2010.

[20] G. Karakonstantis, N. Banerjee, and K. Roy, "Process-variation resilient and voltage-scalable DCT architecture for robust low-power computing," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, pp. 1461–1470, Oct. 2010.

[21] R. Gonzalez, B. Gordon, and M. Horowitz, "Supply and threshold voltage scaling for low power cmos," *Solid-State Circuits, IEEE Journal of*, vol. 32, pp. 1210–1216, Aug. 1997.

[22] E. Mintarno, J. Skaf, R. Zheng, J. Velamala, Y. Cao, S. Boyd, R. Dutton, and S. Mitra, "Self-tuning for maximized lifetime energy-efficiency in the presence of circuit aging," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, pp. 760–773, May 2011.

[23] R. Al-Haddad, R. Oreifej, R. A. Ashraf, and R. F. DeMara, "Sustainable modular adaptive redundancy technique emphasizing partial reconfiguration for reduced power consumption," *International Journal of Reconfigurable Computing*, vol. 2011, 2011.

[24] A. Jacobs, G. Cieslewski, A. D. George, A. Gordon-Ross, and H. Lam, "Reconfigurable fault tolerance: A comprehensive framework for reliable and adaptive fpga-based space computing," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, pp. 21:1–21:30, Dec. 2012.

[25] F. Cancare, D. B. Bartolini, M. Carminati, D. Sciuto, and M. D. Santambrogio, "On the evolution of hardware circuits via reconfigurable architectures," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, pp. 22:1–22:22, Dec. 2012.

[26] A. Nabina and J. L. Nunez-Yanez, "Adaptive voltage scaling in a dynamically reconfigurable fpga-based platform," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, pp. 20:1–20:22, Dec. 2012.

[27] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, pp. 11–33, Jan-March 2004.

[28] M. Tahoori, "High resolution application specific fault diagnosis of FPGAs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, pp. 1775 –1786, Oct. 2011.

[29] D. P. Siewiorek and R. S. Swarz, *Reliable computer systems: design and evaluation*. Natick, MA 01760: A. K. Peters, third ed., 1998.

[30] S. Mitra and E. McCluskey, "Which concurrent error detection scheme to choose ?," in *International Test Conference*, pp. 985–994, 2000.

[31] S. Mitra, W.-J. Huang, N. Saxena, S.-Y. Yu, and E. McCluskey, "Reconfigurable architecture for autonomous self-repair," *Design Test of Computers, IEEE*, vol. 21, pp. 228–240, May-June 2004.

[32] W. Barker, D. Halliday, Y. Thoma, E. Sanchez, G. Tempesti, and A. Tyrrell, "Fault tolerance using dynamic reconfiguration on the POEtic tissue," *Evolutionary Computation, IEEE Transactions on*, vol. 11, pp. 666–684, oct. 2007.

[33] P. C. Haddow and A. M. Tyrrell, "Challenges of evolvable hardware: past, present and the path to a promising future," *Genetic Programming and Evolvable Machines*, vol. 12, pp. 183–215, Sep. 2011.

[34] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti, "Toward robust integrated circuits: The embryonics approach," *Proceedings of the IEEE*, vol. 88, pp. 516–543, April 2000.

[35] P. J. Layzell and A. Thompson, "Understanding inherent qualities of evolved circuits: Evolutionary history as a predictor of fault tolerance," in *Third International Conference on Evolvable Systems (ICES)*, (London, UK), pp. 133–144, Springer-Verlag, 2000.

[36] D. Keymeulen, R. S. Zebulum, Y. Jin, and A. Stoica, "Fault-tolerant evolvable hardware using field-programmable transistor arrays," *Reliability, IEEE Transactions on*, vol. 49, no. 3, pp. 305–316, 2000.

[37] J. Lohn, G. Larchev, and R. DeMara, "A genetic representation for evolutionary fault recovery in virtex FPGAs," in *in Proceedings of the Fifth International Conference on Evolvable Systems (ICES)*, pp. 47–56, 2003.

[38] F. Lombardi, N. Park, M. Al-Hashimi, and H. Pu, "Modeling the dependability of n-modular redundancy on demand under malicious agreement," in *Dependable Computing, 2001. Proceedings. 2001 Pacific Rim International Symposium on*, pp. 68 –75, 2001.

[39] M. G. Parris, C. A. Sharma, and R. F. DeMara, "Progress in autonomous fault recovery of field programmable gate arrays," *ACM Comput. Surv.*, vol. 43, pp. 31:1–31:30, Oct. 2011.

[40] S. Vigander, "Evolutionary fault repair of electronics in space applications," tech. rep., Dissertation, Norwegian University of Science and Technology, Trondheim, Norway, 2001.

[41] M. Liu and J. He, "An evolutionary negative-correlation framework for robust analog-circuit design under uncertain faults," *Evolutionary Computation, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2012.

[42] R. A. Ashraf and R. F. DeMara, "Scalable FPGA refurbishment using netlist-driven evolutionary algorithms," *IEEE Transactions on Computers*, vol. 62, no. 8, pp. 1526–1541, 2013.

[43] S. Kim, H. Chu, I. Yang, S. Hong, S. H. Jung, and K.-H. Cho, "A hierarchical self-repairing architecture for fast fault recovery of digital systems inspired from paralogous gene regulatory circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, pp. 2315 –2328, dec. 2012.

[44] M. Makhzan, A. Eltawil, and F. Kurdahi, "Architectural and algorithm level fault tolerant techniques for low power high yield multimedia devices," in *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS*, pp. 124–131, July 2008.

[45] N. Imran, J. Lee, and R. F. DeMara, "Fault demotion using reconfigurable slack (FaDReS)," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 7, pp. 1364–1368, 2013.

[46] H. Cho, L. Leem, and S. Mitra, "ERSA: Error resilient system architecture for probabilistic applications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, pp. 546–558, April 2012.

[47] G. Karakonstantis, D. Mohapatra, and K. Roy, "Logic and memory design based on unequal error protection for voltage-scalable, robust and adaptive DSP systems," *Journal of Signal Processing Systems (JSPS)*, vol. 68, pp. 415–431, 2012.

[48] P. Whatmough, S. Das, D. Bull, and I. Darwazeh, "Circuit-level timing error tolerance for low-power DSP filters and transforms," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 6, pp. 989–999, 2013.

[49] W. Rao, A. Orailoglu, and R. Karri, "Nanofabric topologies and reconfiguration algorithms to support dynamically adaptive fault tolerance," in *VLSI Test Symposium*, p. 6, april-4 may 2006.

[50] M. A. V.-R. Juan A. Gmez-Pulido and J. M. Snchez-Prez, "High-speed reconfigurable parallel system to design good error correcting codes in communications," *Journal of Signal Processing Systems*, vol. 66, pp. 147–152, 2012.

[51] M. Kthiri, H. Loukil, A. Atitallah, P. Kadionik, D. Dallet, and N. Masmoudi, "FPGA architecture of the LDPS Motion Estimation for H.264/AVC Video Coding," *Journal of Signal Processing Systems*, vol. 68, pp. 273–285, 2012.

[52] R. Rubin and A. DeHon, "Choose-your-own-adventure routing: lightweight load-time defect avoidance," in *Proceedings of the ACM/SIGDA international symposium on FPGAs*, (New York, NY, USA), pp. 23–32, ACM, 2009.

[53] M. Pereira, L. Braun, M. Hubner, J. Becker, and L. Carro, "Run-time resource instantiation for fault tolerance in FPGAs," in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp. 88–95, June 2011.

[54] K. Siozios and D. Soudris, "Low-cost fault tolerant targeting FPGA devices," in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Special session on dependability by reconfigurable hardware*, June 2012.

[55] R. Abdallah and N. Shanbhag, "Minimum-energy operation via error resiliency," *Embedded Systems Letters, IEEE*, vol. 2, pp. 115–118, Dec. 2010.

[56] E. Kim and N. Shanbhag, "Soft N-Modular redundancy," *Computers, IEEE Transactions on*, vol. 61, pp. 323–336, March 2012.

[57] S. Narayanan, G. Varatkar, D. Jones, and N. Shanbhag, "Computation as estimation: A general framework for robustness and energy efficiency in SoCs," *Signal Processing, IEEE Transactions on*, vol. 58, pp. 4416–4421, Aug. 2010.

[58] G. Varatkar and N. Shanbhag, "Error-resilient motion estimation architecture," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, pp. 1399–1412, Oct. 2008.

[59] G. Greenwood, "On the practicality of using intrinsic reconfiguration for fault recovery," *Evolutionary Computation, IEEE Transactions on*, vol. 9, pp. 398–405, Aug. 2005.

[60] J.-C. Laprie, "Dependable computing and fault tolerance : Concepts and terminology," in *25th International Symposium on Fault-Tolerant Computing - Highlights from Twenty-Five Years*, Jun 1995.

[61] M. Malek, "A comparison connection assignment for diagnosis of multiprocessor systems," in *Proceedings of the 7th annual symposium on Computer Architecture (ISCA)*, (New York, NY, USA), pp. 31–36, ACM, 1980.

[62] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of diagnosable systems," *Electronic Computers, IEEE Transactions on*, vol. EC-16, pp. 848–854, Dec. 1967.

[63] C. Carmichael, "Triple module redundancy design techniques for virtex FPGAs," *Xilinx Application Note: Virtex Series XAPP197 (v1.0.1), July 6, 2006*, 2006.

[64] F. Kastensmidt, L. Sterpone, L. Carro, and M. Reorda, "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs," in *Design, Automation and Test in Europe*, pp. 1290–1295 Vol. 2, 7-11 2005.

[65] S. Dutt, V. Verma, and V. Suthar, "Built-in-self-test of FPGAs with provable diagnosabilities and high diagnostic coverage with application to online testing," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, pp. 309 –326, Feb. 2008.

[66] M. Gericota, G. Alves, M. Silva, and J. Ferreira, "Reliability and availability in reconfigurable computing: A basis for a common solution," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, pp. 1545–1558, Nov. 2008.

[67] M. Abramovici, J. M. Emmert, and C. E. Stroud, "Roving STARs: an integrated approach to on-line testing, diagnosis, and fault tolerance for FPGAs in adaptive computing systems," in *The Third NASA/DoD Workshop on Evolvable Hardware*, pp. 73–92, 2001.

[68] M. Gao, H.-M. S. Chang, P. Lisherness, and K.-T. T. Cheng, "Time-multiplexed online checking," *IEEE Transactions on Computers*, vol. 60, no. 9, pp. 1300–1312, 2011.

[69] J. Russell and C. Kime, "System fault diagnosis: Closure and diagnosability with repair," *Computers, IEEE Transactions on*, vol. C-24, pp. 1078–1089, Nov. 1975.

[70] A. Friedman and L. Simoncini, "System-level fault diagnosis," *Computer*, vol. 13, pp. 47–53, March 1980.

[71] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, pp. 382–401, July 1982.

[72] Z. Ma and A. Krings, "Dynamic hybrid fault modeling and extended evolutionary game theory for reliability, survivability and fault tolerance analyses," *Reliability, IEEE Transactions on*, vol. 60, pp. 180 –196, March 2011.

[73] Z. Ma and A. W. Krings, "Survival analysis approach to reliability, survivability and prognostics and health management (phm)," in *Aerospace Conference, 2008 IEEE*, pp. 1 –20, March 2008.

[74] W.-S. Hong and S.-Y. Hsieh, "Strong diagnosability and conditional diagnosability of augmented cubes under the comparison diagnosis model," *Reliability, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2011.

[75] P.-L. Lai, "A systematic algorithm for identifying faults on hypercube-like networks under the comparison model," *Reliability, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2012.

[76] M. Garvie and A. Thompson, "Scrubbing away transients and jiggling around the permanent: long survival of FPGA systems through evolutionary self-repair," in *IEEE International On-Line Testing Symposium (IOLTS)*, pp. 155–160, 2004.

[77] H. Tan and R. F. DeMara, "A multilayer framework supporting autonomous run-time partial reconfiguration," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, pp. 504–516, May 2008.

[78] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Low overhead fault-tolerant FPGA systems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 6, no. 2, pp. 212–221, 1998.

[79] C. A. Sharma and R. F. DeMara, "A combinatorial group testing method for FPGA fault location," in *ACST'06: Proceedings of the 2nd IASTED international conference on Advances in computer science and technology*, (Anaheim, CA, USA), pp. 55–60, ACTA Press, 2006.

[80] C. A. Sharma, A. Sarvi, A. Alzahrani, and R. F. DeMara, "Self-healing reconfigurable logic using autonomous group testing," *Microprocessors and Microsystems*, vol. 37, no. 2, pp. 174–184, 2013.

[81] N. Steiner and P. Athanas, "Hardware autonomy and space systems," in *Aerospace conference, 2009 IEEE*, pp. 1–13, March 2009.

[82] H. Flatt, H. Blume, and P. Pirsch, "Mapping of a real-time object detection application onto a configurable RISC/Coprocessor architecture at full HD resolution," in *Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*, (Quintana Roo), pp. 452 –457, Dec. 2010.

[83] D. Bouldin, "Enhancing electronic systems with reconfigurable hardware," *Circuits and Devices Magazine, IEEE*, vol. 22, pp. 32 –36, May-June 2006.

[84] A. Jara-Berrocal and A. Gordon-Ross, "VAPRES: a virtual architecture for partially reconfigurable embedded systems," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 837–842, 2010.

[85] M. Kuehnle, A. Brito, C. Roth, K. Dagas, and J. Becker, "The study of a dynamic reconfiguration manager for systems-on-chip," *Annual Symposium on VLSI*, 2011.

[86] L. Shannon, "Leveraging reconfigurability in the design process," *International Conference on Field Programmable Logic and Applications*, pp. 731–732, 2005.

[87] J. Sloan, D. Kesler, R. Kumar, and A. Rahimi, "A numerical optimization-based methodology for application robustification: Transforming applications for error tolerance," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 161 –170, July 2010.

[88] E. Stott, P. Sedcole, and P. Cheung, "Fault tolerant methods for reliability in FPGAs," in *International Conference on Field Programmable Logic and Applications (FPL)*, pp. 415–420, 8-10 2008.

[89] M. Abramovici, C. Stroud, and J. Emmert, "Online BIST and BIST-based diagnosis of fpga logic blocks," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 12, pp. 1284–1294, 2004.

[90] R. F. DeMara and K. Zhang, "Autonomous FPGA fault handling through competitive run-time reconfiguration," in *NASA/DoD Conference on Evolvable Hardware*, pp. 109–116, July 2005.

[91] K. Zhang, R. F. DeMara, and C. A. Sharma, "Consensus-based evaluation for fault isolation and on-line evolutionary regeneration," in *International Conference in Evolvable Systems (ICES'05)*, pp. 12–24, 2005.

[92] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*, (Hillsdale, NJ, USA), pp. 101–111, L. Erlbaum Associates Inc., 1985.

[93] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *Computers, IEEE Transactions on*, vol. C-23, pp. 90–93, Jan. 1974.

[94] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ., 3rd ed., 2008.

[95] H. Jian, P. Matthew, L. Jooheung, and F. D. Ronald, "Scalable FPGA-based architecture for DCT computation using dynamic partial reconfiguration," *ACM Trans. Embed. Comput. Syst.*, vol. 9, no. 1, pp. 1–18, 2009. 1596541.

[96] Xilinx, "Embedded system tools reference manual," 2008. Retrieved on January 8, 2012 [Online] http://www.xilinx.com/.../edk10_est_rm.pdf.

[97] R. F. DeMara, J. Lee, R. Al-Haddad, R. S. Oreifej, R. Ashraf, B. Stensrud, and M. Quist, "Dynamic partial reconfiguration approach to the design of sustainable edge detectors.," in *ERSA'10*, pp. 49–58, 2010.

[98] J. Canny, "A computational approach to edge detection," *Pattern Anal. and Mach. Intell., IEEE Trans. on*, vol. PAMI-8, pp. 679 –698, Nov. 1986.

[99] T. Kim, H. Adeli, C. Ramos, and B.-H. Kang, *Signal Processing, Image Processing, and Pattern Recognition*. Springer-Verlag, Springer Heidelberg Dordrecht London New York: Springer, 2011.

[100] VGG, "Oxford visual geometry group (vgg)'s images dataset : Aerial views," 2012. Retrieved on Feb. 13, 2012 [Online] http://www.robots.ox.ac.uk/~vgg/data/.

[101] I. Koren and S. Su, "Reliability analysis of n-modular redundancy systems with intermittent and permanent faults," *Computers, IEEE Transactions on*, vol. C-28, pp. 514–520, July 1979.

[102] A. Leon-Garcia, *Probability, statistics, and random processes for Electrical Engineering*. Pearson/Prentice Hall, 2008.

[103] C. A. B. Smith, "The counterfeit coin problem," *The Mathematical Gazette*, vol. 31, no. 293, pp. pp. 31–39, 1947.

[104] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos in the wild," in *IEEE International Conference on Computer Vision and Pattern Recognition(CVPR)*, (Miami), 2009.

[105] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Springer, 2009.

[106] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, "Diffusion maps, spectral clustering and eigenfunctions of Fokker-Planck operators," in *in Advances in Neural Information Processing Systems 18*, pp. 955–962, MIT Press, 2005.

[107] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[108] S. Yang, "Logic synthesis and optimization benchmarks version 3," tech. rep., Microelectronics Center of North Carolina, 1991.

[109] D. Gleich, "pagerank at mathworks.com." `http://www.stanford.edu/~dgleich/programs-old.html`, 2006.

[110] J. Huang and J. Lee, "Reconfigurable architecture for ZQDCT using computational complexity prediction and bitstream relocation," *Embedded Systems Letters, IEEE*, vol. 3, no. 1, pp. 1–4, 2011.

[111] K. V. Palem, L. N. Chakrapani, Z. M. Kedem, A. Lingamneni, and K. K. Muntimadugu, "Sustaining moore's law in embedded computing through probabilistic and approximate design: retrospects and prospects," in *International conference on Compilers, architecture, and synthesis for embedded systems*, CASES, (New York, NY, USA), pp. 1–10, ACM, 2009.

[112] V. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. Chakradhar, "Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency," in *47th ACM/IEEE Design Automation Conference (DAC)*, pp. 555–560, June 2010.

[113] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator," in *14th ACM/IEEE interna-*

tional symposium on Low power electronics and design (ISLPED), (New York, NY, USA), pp. 195–200, ACM, 2009.

[114] N. Forbes, "Biologically inspired computing," *Computing in Science and Engineering*, vol. 2, no. 6, pp. 83–87, 2000.

[115] H. Schmeck, C. Mller-Schloer, E. akar, M. Mnif, and U. Richter, "Adaptivity and self-organisation in organic computing systems," in *Organic Computing A Paradigm Shift for Complex Systems* (C. Mller-Schloer, H. Schmeck, and T. Ungerer, eds.), vol. 1 of *Autonomic Systems*, pp. 5–37, Springer Basel, 2011.

[116] SPP1183, "German research foundation (DFG) organic compututing research program," [Online] `http://projects.aifb.kit.edu/effalg/oc/`.

[117] B. Hargreaves, H. Hult, and S. Reda, "Within-die process variations: How accurately can they be statistically modeled?," in *Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 524–530, March 2008.

[118] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold voltage (NTV) design: opportunities and challenges," in *Proceedings of the 49th Annual Design Automation Conference*, DAC'12, (New York, NY, USA), pp. 1153–1158, ACM, 2012.

[119] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, pp. 850–862, April 2010.

[120] R. N. Al-Haddad, *An adaptive modular redundancy technique to self-regulate availability, area, and energy consumptions in mission-critical applications*. Phd dissertation, University of Central Florida, Orlando, Florida, 2011.

[121] R. W. Butler, "A primer on architectural level fault tolerance," Tech. Report NASA/TM-2008-215108, The NASA STI Program Office, Feb. 2008.

[122] R. Hegde and N. Shanbhag, "Soft digital signal processing," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 9, pp. 813–823, Dec. 2001.

[123] G. Varatkar and N. Shanbhag, "Energy-efficient motion estimation using error-tolerance," in *International Symposium on Low Power Electronics and Design, ISLPED*, pp. 113–118, Oct. 2006.

[124] C. Lisboa, L. Carro, C. Argyrides, and D. Pradhan, "Algorithm level fault tolerance: A technique to cope with long duration transient faults in matrix multiplication algorithms," in *26th IEEE VLSI Test Symposium, VTS*, pp. 363–370, May 2008.

[125] A. Fathy, I. Tarrad, H. Hamed, and A. Awad, "Advanced encryption standard algorithm: Issues and implementation aspects," in *Advanced Machine Learning Technologies and Applications* (A. Hassanien, A.-B. Salem, R. Ramadan, and T.-h. Kim, eds.), vol. 322 of *Communications in Computer and Information Science*, pp. 516–523, Springer Berlin Heidelberg, 2012.

[126] G. Gao, Y. Wang, J. Cui, and R. Yao, "Research on multi-objective on-line evolution technology of digital circuit based on fpga model," in *Proceedings of the 7th international conference on Evolvable systems: from biology to hardware*, ICES'07, pp. 67–76, 2007.

[127] S. Zhao and L. Jiao, "Multi-objective evolutionary design and knowledge discovery of logic circuits based on an adaptive genetic algorithm," *Genetic Programming and Evolvable Machines*, vol. 7, pp. 195–210, 2006.

[128] H. Kutami, Y. Fukushima, M. Fukushi, I. Yairi, and T. Hattori, "Route-aware task mapping method for fault-tolerant 2d-mesh network-on-chips," in *Defect and Fault Tolerance in VLSI*

and Nanotechnology Systems (DFT), IEEE International Symposium on, pp. 472–480, Oct. 2011.

[129] O. Derin, D. Kabakci, and L. Fiorin, "Online task remapping strategies for fault-tolerant network-on-chip multiprocessors," in *Fifth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pp. 129–136, May 2011.

[130] A. Maheshwari, W. Burleson, and R. Tessier, "Trading off transient fault tolerance and power consumption in deep submicron (DSM) VLSI circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, pp. 299–311, March 2004.

[131] H. Singh, K. Agarwal, D. Sylvester, and K. Nowka, "Enhanced leakage reduction techniques using intermediate strength power gating," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 11, pp. 1215–1224, 2007.

[132] B. Zatt, M. Shafique, S. Bampi, and J. Henkel, "A low-power memory architecture with application-aware power management for motion & disparity estimation in multiview video coding," in *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '11, (Piscataway, NJ, USA), pp. 40–47, IEEE Press, 2011.

[133] Xilinx, "Planahead 10.1 user guide," 2008.

[134] MATLAB, "Genetic algorithm options," *MathWorks Documentation Center*, 2013.

[135] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[136] D. Decoste and B. Schlkopf, "Training invariant support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 161–190, 2002.

[137] P. M. Kuhn and K. P. M., *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*. Norwell, MA, USA: Kluwer Academic Publishers, 1st ed., 1999.

[138] B. Zatt, M. Shafique, F. Sampaio, L. Agostini, S. Bampi, and J. Henkel, "Run-time adaptive energy-aware motion and disparity estimation in multiview video coding," in *Proceedings of the 48th Design Automation Conference (DAC)*, DAC '11, (New York, NY, USA), pp. 1026–1031, ACM, 2011.

[139] V. Do and K. Yun, "A low-power vlsi architecture for full-search block-matching motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 8, pp. 393–398, Aug. 1998.

[140] S. Saponara and L. Fanucci, "Data-adaptive motion estimation algorithm and vlsi architecture design for low-power video systems," *Computers and Digital Techniques, IEE Proceedings*, vol. 151, pp. 51–59, Jan. 2004.

[141] C. Kalaycioglu, O. Ulusel, and I. Hamzaoglu, "Low power techniques for motion estimation hardware," in *International Conference on Field Programmable Logic and Applications (FPL)*, pp. 180–185, Sep. 2009.

[142] I. S. Chong and A. Ortega, "Dynamic voltage scaling algorithms for power constrained motion estimation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, pp. 101–104, April 2007.

[143] I. Ahmad, W. Zheng, J. Luo, and M. Liou, "A fast adaptive motion estimation algorithm," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, pp. 420–438, March 2006.

[144] L.-W. Lee, J.-F. Wang, J.-Y. Lee, and J.-D. Shie, "Dynamic search-window adjustment and interlaced search for block-matching algorithm," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 3, pp. 85–87, Feb. 1993.

[145] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Nat. Telecomm. Conf., New Orleans, LA*, 1981.

[146] R. Li, B. Zeng, and M. Liou, "A new three-step search algorithm for block motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 4, pp. 438–442, Aug 1994.

[147] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *Image Processing, IEEE Transactions on*, vol. 9, pp. 287–290, Feb 2000.

[148] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Visual Communications and Image Processing*, (San Jose, CA), pp. 1069–1079, Jan. 2002.

[149] X. Lee and Y.-Q. Zhan, "A fast hierarchical motion-compensation scheme for video coding using block feature matching," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, pp. 627–635, Dec. 1996.

[150] S. Na and C.-M. Kyung, "Activity-based motion estimation scheme for h.264 scalable video coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, pp. 1475–1485, Nov. 2010.

[151] H. Yin, H. Jia, H. Qi, X. Ji, X. Xie, and W. Gao, "A hardware-efficient multi-resolution block matching algorithm and its vlsi architecture for high definition mpeg-like video encoders," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, pp. 1242–1254, Sep. 2010.

[152] A. Paul, Y.-C. Jiang, J.-F. Wang, and J.-F. Yang, "Parallel reconfigurable computing-based mapping algorithm for motion estimation in advanced video coding," *ACM Trans. Embed. Comput. Syst.*, vol. 11, pp. 40:1–40:18, Aug. 2012.

[153] O. Tasdizen, H. Kukner, A. Akin, and I. Hamzaoglu, "A high performance reconfigurable motion estimation hardware architecture," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 882–885, April 2009.

[154] G. Pastuszak and M. Jakubowski, "Adaptive computationally-scalable motion estimation for the hardware h.264/avc encoder," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2012.

[155] A. Celebi, H.-J. Lee, and S. Erturk, "Bit plane matching based variable block size motion estimation method and its hardware architecture," *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 3, pp. 1625–1633, 2010.

[156] C. wei, H. Hui, T. jiarong, L. Jinmei, and M. Hao, "A high-performance reconfigurable vlsi architecture for vbsme in H.264," *Consumer Electronics, IEEE Transactions on*, vol. 54, no. 3, pp. 1338–1345, 2008.

[157] L. Lu, J. McCanny, and S. Sezer, "Reconfigurable system-on-a-chip motion estimation architecture for multi-standard video coding," *Computers Digital Techniques, IET*, vol. 4, no. 5, pp. 349–364, 2010.

[158] C.-H. Cheng, Y. Liu, and C.-L. Hsu, "Design of an error detection and data recovery architecture for motion estimation testing applications," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, pp. 665–672, April 2012.

[159] C.-L. Hsu, C.-H. Cheng, and Y. Liu, "Built-in self-detection/correction architecture for motion estimation computing arrays," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, pp. 319–324, Feb. 2010.

[160] G. V. Varatkar and N. R. Shanbhag, "Variation-tolerant motion estimation architecture," in *Signal Processing Systems, IEEE Workshop on*, pp. 126–131, Oct. 2007.

[161] H. Chung and A. Ortega, "Analysis and testing for error tolerant motion estimation," in *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pp. 514–522, Oct 2005.

[162] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *Circuits and Systems, IEEE Transactions on*, vol. 36, pp. 1301–1308, Oct. 1989.

[163] C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, T.-C. Wang, and L.-G. Chen, "Analysis and architecture design of variable block-size motion estimation for H.264/AVC," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 53, no. 3, pp. 578–593, 2006.

[164] Xilinx, "Planahead user guide," UG632 (v14.3) October 16, 2012.

[165] Xilinx, "Partial reconfiguration user guide," UG702 (v14.3) October 16, 2012.

[166] Xilinx, "Partial reconfiguration tutorial: Planahead design tool," UG743 (v14.1) May 8, 2012.

[167] W.-J. Huang, N. Saxena, and E. McCluskey, "A reliable LZ data compressor on reconfigurable coprocessors," in *Field-Programmable Custom Computing Machines, 2000 IEEE Symposium on*, pp. 249–258, Apr. 2000.

[168] N. Imran, R. DeMara, J. Lee, and J. Huang, "Self-adapting resource escalation for resilient signal processing architectures," *Journal of Signal Processing Systems*, pp. 1–24, 2013.

[169] E. Mizan, T. Amimeur, and M. Jacome, "Self-imposed temporal redundancy: An efficient technique to enhance the reliability of pipelined functional units," in *19th International Symposium on Computer Architecture and High Performance Computing*, pp. 45–53, Oct.

[170] R. Dorfman, "The detection of defective members of large populations," *The Annals of Mathematical Statistics*, vol. 14, no. 4, pp. pp. 436–440, 1943.

[171] E. Litvak, X. M. Tu, and M. Pagano, "Screening for the presence of a disease by pooling sera samples," *Journal of the American Statistical Association*, vol. 89, no. 426, pp. pp. 424–434, 1994.

[172] Xilinx, "PowerPC 405 processor block reference guide (ug018)," 2010. Retrieved on January 8, 2012 [Online] `http://www.xilinx.com/.../ug018.pdf`.

[173] Xilinx, "Os and libraries document collection EDK 9.2i," 2007. Retrieved on January 8, 2012 [Online] `http://www.xilinx.com/.../edk92i_oslib_rm.pdf`.

[174] Xilinx, "Virtex-4 FPGA configuration user guide (ug071)," 2009. Retrieved on January 8, 2012 [Online] `http://www.xilinx.com/.../ug071.pdf`.

[175] R. N. Al-Haddad, C. A. Sharma, and R. F. DeMara, "Performance evaluation of two allocation schemes for combinatorial group testing fault isolation.," in *ERSA'07*, pp. 269–272, July 2007.

[176] Xilinx, "LogiCORE IP XPS HWICAP datasheet," Retrieved on January 11, 2012 [Online] `http://www.xilinx.com/.../xps_hwicap.pdf`.

[177] J. Becker, M. Huebner, and M. Ullmann, "Power estimation and power measurement of Xilinx Virtex FPGAs: trade-offs and limitations," in *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on*, pp. 283 – 288, Sept. 2003.

[178] Xilinx, "XtremeDSP 48 slice," Retrieved on January 11, 2012 [Online] http://www.xilinx.com/technology/dsp/xtremedsp.htm.

[179] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, pp. 688–703, July 2003.

[180] G. Karakonstantis, D. Mohapatra, and K. Roy, "System level DSP synthesis using voltage overscaling, unequal error protection & adaptive quality tuning," in *IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 133–138, Oct. 2009.

[181] M. Bucciero, J. P. Walters, and M. French, "Software fault tolerance methodology and testing for the embedded PowerPC," in *Proceedings of the IEEE Aerospace Conference*, (Big Sky, MT), pp. 1–9, March 2011.

[182] W.-J. Huang, S. Mitra, and E. McCluskey, "Fast run-time fault location in dependable FPGA-based applications," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pp. 206–214, 2001.

[183] T. Becker, W. Luk, and P. Y. K. Cheung, "Enhancing relocatability of partial bitstreams for run-time reconfiguration," in *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 35–44, 2007.

[184] NIST, "FIPS PUB 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Department of Commerce, November 2001.," [Online] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[185] S. Drimer, *Security for volatile FPGAs*. Phd dissertation, Univeristy of Cambridge, 15 JJ Thomson Avenue Cambridge CB3 0FD United Kingdom, 2009.

[186] S. Srinivasan, R. Krishnan, P. Mangalagiri, Y. Xie, V. Narayanan, M. Irwin, and K. Sarpatwari, "Toward increasing FPGA lifetime," *Dependable and Secure Computing, IEEE Transactions on*, vol. 5, pp. 115 –127, april-june 2008.

[187] C. Poivey, M. Berg, S. Stansberry, M. Friendlich, H. Kim, D. Petrick, and K. LaBel, "Heavy ion SEE test of Virtex-4 FPGA XC4VFX60 from Xilinx," June 2007.

[188] J. Heiner, N. Collins, and M. Wirthlin, "Fault tolerant ICAP controller for high-reliable internal scrubbing," in *Aerospace Conference, 2008 IEEE*, pp. 1–10, 2008.

[189] K. Papadimitriou, A. Dollas, and S. Hauck, "Performance of partial reconfiguration in fpga systems: A survey and a cost model," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 4, pp. 36:1–36:24, Dec. 2011.