J Sign Process Syst manuscript No.
(will be inserted by the editor)

Self-adapting Resource Escalation for Resilient Signal Processing Architectures

Naveed Imran · Ronald F. DeMara · Jooheung Lee · and Jian Huang

Received: Jul 15, 2012 / Revised: April 01, 2013

Abstract 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 Peak Signal-to-Noise Ratio (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.

N. Imran and R. F. DeMara
Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL, 32816 USA
E-mail: naveed@knights.ucf.edu, demara@mail.ucf.edu

J. Lee
Department of Electronic and Electrical Engineering, Hongik University, Sejong 339-701, Korea
E-mail: joolee@hongik.ac.kr

J. Huang Advanced Micro Devices (AMD)
E-mail: harryhuang2000@gmail.com

Keywords Survivability · Reconfigurable hardware for video compression · Semiconductor aging · Process variations · Autonomous recovery · Mission-critical systems · FPGAs

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
$\boldsymbol{\Psi}$	Syndrome Matrix
Φ	Fitness State Vector
$\hat{\Phi}$	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	Total number of nodes
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

1 Introduction

In the domain of Digital Signal Processing (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 [1] [2]. 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 [3]. 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 reliability [4] [5]. 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 [1] [6]. 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.

Field Programmable Gate Arrays (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 [7] [8]. 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 [9] [10]. This characteristic has already made FPGAs popular for application in the space exploration community [11] [12] [13]. On the other hand, SRAM-based FPGAs are also susceptible to soft (transient) errors as well as hard (permanent) faults [14] that can be addressed using the techniques developed in this paper.

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.

In this paper, we present a strategy 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 is shown in Fig. 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 *Discrete Cosine Transform (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

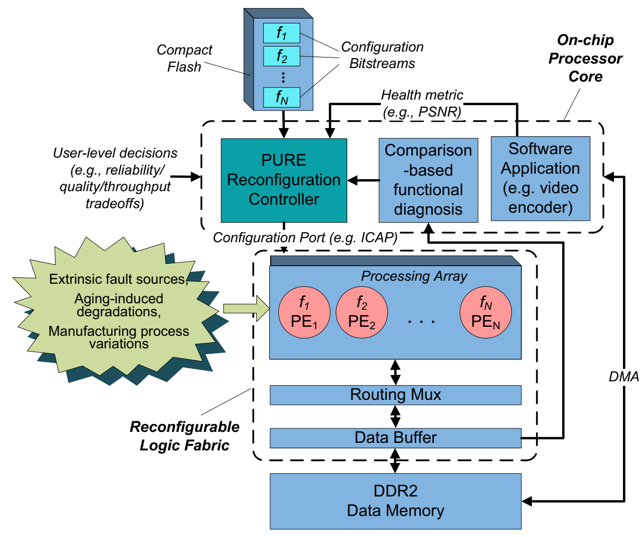

Compact Flash
f_1
f_2
f_N
Configuration Bitstreams
On-chip Processor Core
Health metric (e.g., PSNR)
User-level decisions (e.g., reliability/ quality/throughput tradeoffs)
PURE Reconfiguration Controller
Comparison -based functional diagnosis
Software Application (e.g. video encoder)
Configuration Port (e.g. ICAP)
Extrinsic fault sources: Aging-induced degradations, Manufacturing process variations
Processing Array
f_1 PE$_1$
f_2 PE$_2$
f_N PE$_N$
Routing Mux
Data Buffer
DMA
Reconfigurable Logic Fabric
DDR2 Data Memory

Fig. 1 A system-level block diagram illustrating the self-adapting resource escalation of the FPGA device

alternative partial Configuration Bitstreams for the PEs which are stored in a Compact Flash external memory device. A Configuration Port, such as the *Internal Configuration Access Port* (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.

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 [15], [16], [17]. 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.

2 Related Work

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 [2] [18]. The reliability issues of modern signal processing architectures due to voltage scaling are being addressed in recent research [19] [20]. 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 [1] 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 [21]. Likewise, to minimize the power overhead of error resilience while maintaining signal quality, the scheme proposed in [2] exposes only less crucial blocks to process variation and channel noise.

Fault-handling (FH) systems typically employ a sequence of resolution phases including *Fault Detection*, *Fault-Diagnosis*, and *Fault Recovery*. 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 [22]. 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 [23]. In this paper, reliability is attained by employing the reconfigurable modules in the fault-handling flow, whereas availability is maintained by minimum interruption of the main throughput data-path.

The redundancy based fault detection methods are popular among fault-tolerant systems community, with costs of area and power overhead. In the *Comparison Diagnosis Model* [24] [25], 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 [26], or a diverse duplex design to reduce common mode faults [4]. Its advantage is a very low fault detection latency. A *Triple Modular Redundancy (TMR)* system [27] [28] 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 PURE 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 PURE approach, the redundant modules are continually utilized in the datapath during the normal mission operation.

While reconfiguration and redundancy are fundamental components of a fault recovery 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.

Reliability of FPGA based designs [29] can be achieved in various ways. Table. 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* [30] [31] approaches to increase sustainability, such as refurbishing the failed instance using *jiggling* [32]. Other *active recovery* techniques incorporate control schemes which realize intelligent actions to cope with a failure. Evolutionary techniques [33] 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 Algorithms (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 partial reconfiguration 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 *Competitive Runtime Reconfiguration (CRR)* [26] 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 runtime, 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 *Built-in Self Test (BIST)* techniques [34] 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 [35]. In [36], a pair of blocks configured with identical operating modes are subjected to resource-oriented test patterns. This *Self-Testing AReas (STARs)* approach keeps a relative small area of the device off-line and being tested, while the rest of the device is online and

Table 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*

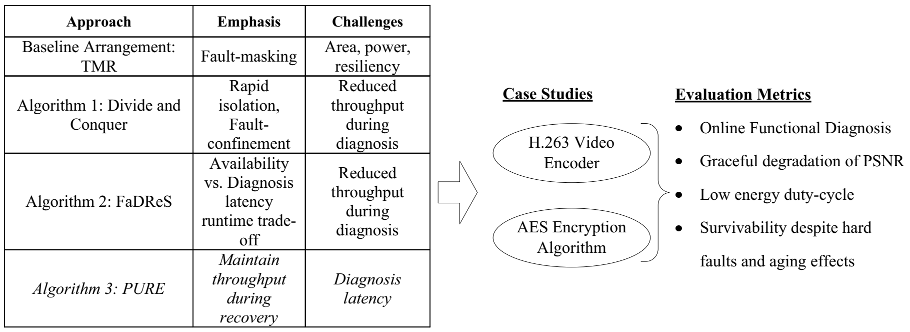
continues its operation. STARs compares the output of each Programmable Logic Block (PLB) to that of an identically configured PLB. This utilizes the property that a discrepancy between the output flags the PLB as suspect as outlined by Dutt et. al's *Roving Tester (ROTE)* technique [35] and used in Gericota et. al's active replication technique [37] which concurrently creates replicas of *Configurable Logic Blocks (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, PURE performs 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 PURE, 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.

3 Fault-Handling Method

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 [38] [39] or spatial CED [4] 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 **G** is called *Fault Diagnosis*. The maximum number of faulty nodes which a scheme guarantees to identify is known as *diagnosability* of **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 paper, 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.



Approach	Emphasis	Challenges
Baseline Arrangement: TMR	Fault-masking	Area, power, resiliency
Algorithm 1: Divide and Conquer	Rapid isolation, Fault-confinement	Reduced throughput during diagnosis
Algorithm 2: FaDReS	Availability vs. Diagnosis latency runtime trade-off	Reduced throughput during diagnosis
Algorithm 3: PURE	*Maintain throughput during recovery*	*Diagnosis latency*

Case Studies
H.263 Video Encoder
AES Encryption Algorithm
Evaluation Metrics
Online Functional Diagnosis
Graceful degradation of PSNR
Low energy duty-cycle
Survivability despite hard faults and aging effects

Fig. 2 Overview of recovery algorithms evaluated herein and the evaluation approach

Fig. 2 illustrates the scope, approaches, and metrics of this paper. 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.

4 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 [19] [40].

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.

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

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.

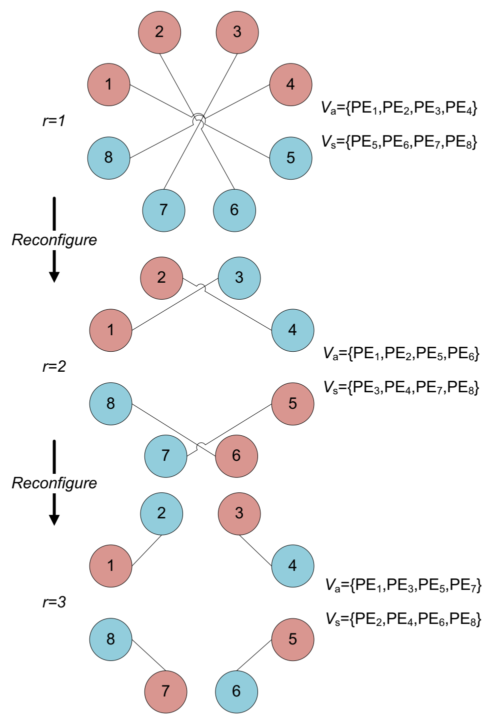
5 Reconfiguration Algorithm 1: Divide-and-Conquer Method

Group testing schemes [41] [42] [43] 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.

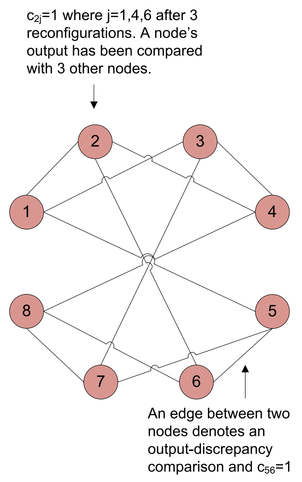
We identify two scenarios in which this hierarchical divide-and-conquer strategy may be more appealing to be employed than the two algorithms discussed in further sections:

– If there are no restrictions on throughput or availability during the fault-handling phase, then halving of the suspect pool [42] 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. 3 illustrates the topologies in the diagnostic flow at various reconfiguration iterations. The number of edges in the graph of Fig. 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. 4 in which dotted lined boxes correspond to the checking slacks and solid lined boxes correspond to active PEs. Algorithm 1 defines the diagnosis process.



(a) Time varying topologies at various reconfiguration instants


c_{2j}=1 where j=1,4,6 after 3 reconfigurations. A node's output has been compared with 3 other nodes.
An edge between two nodes denotes an output-discrepancy comparison and c_{56}=1

(b) Graph represented by **C** after 3 reconfigurations

Fig. 3 Divide-and-conquer method for fault diagnosis

To measure the diagnosability of **G** obtained by the divide-and-conquer reconfiguration method, we observe from Fig. 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

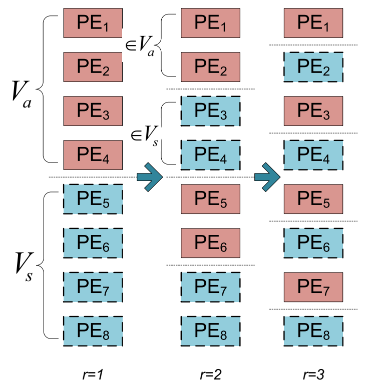


Fig. 4 Various reconfiguration instants in the divide-and-conquer approach

Algorithm 1 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 $\psi_{ij} = 1$
```

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 **G**. 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 Fig. 3a is given by:

$$\mathbf{\Psi} = \left[\begin{array}{c:cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hdashline 1 & 0 & 0 & 0 & x & 0 & x & x & x \\ 2 & 0 & 0 & x & 1 & x & 1 & x & x \\ 3 & 0 & x & 0 & 1 & x & x & 0 & x \\ 4 & x & 1 & 1 & 0 & x & x & x & 1 \\ 5 & 0 & x & x & x & 0 & 1 & 0 & x \\ 6 & x & 1 & x & x & 1 & 0 & x & 1 \\ 7 & x & x & 0 & x & 0 & x & 0 & 0 \\ 8 & x & x & x & 1 & x & 1 & 0 & 0 \end{array}\right] \quad (2)$$

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.

6 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 [44]. 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:

$$P(X) = \frac{Number\ of\ favorable\ scenarios}{Total\ number\ of\ diagnosable\ fault\ scenarios} \tag{3}$$

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

$$CPDC(X \geq 1) = \sum_{r=1}^{N} P(X = r) \tag{4}$$

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. 4.

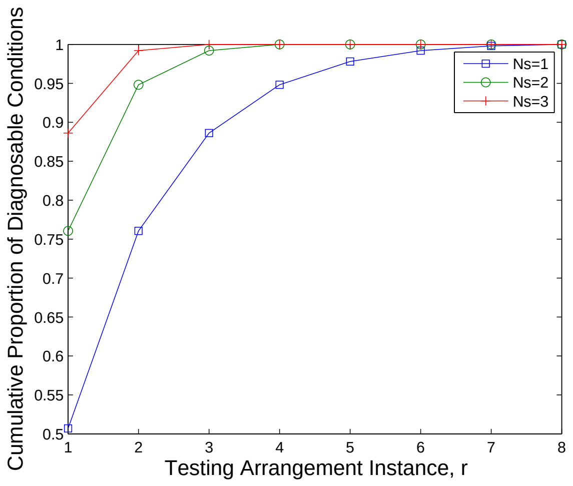


Fig. 5 Cumulative Proportion of Diagnosable Conditions demonstrating diagnosis benefit of additional slacks

Fig. 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 $N_s = 1$.

6.1 Diagnosis by voting

The algorithm for diagnosis employing dynamic NMR voting on module level is given in Algorithm 2. Fig. 6 shows various steps in the diagnosis process.

Fault diagnosis latency T_{diag} is defined as:

$$T_{diag} = (T_{eval} + T_{rec}N_s) \sum_{j=1}^{N_r} I_j \tag{5}$$

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:

$$T_{diag,max} = (T_{eval} + T_{rec}N_s) \sum_{j=0}^{N_a-1} (N_a - j) \tag{6}$$

Algorithm 2 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 $\psi_{ij} = 0$
```

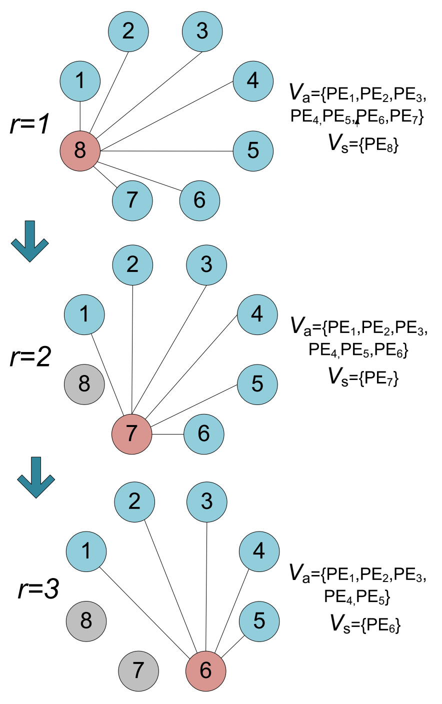

r=1
Va={PE1,PE2,PE3, PE4,PE5,PE6,PE7}
Vs={PE8}
r=2
Va={PE1,PE2,PE3, PE4,PE5,PE6}
Vs={PE7}
r=3
Va={PE1,PE2,PE3, PE4,PE5}
Vs={PE6}

Fig. 6 Fault Diagnosis in the FaDReS Approach

6.2 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 locality 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.

7 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 3 can be used to prioritize throughput while the FaDReS Algorithm 2 can be used to prioritize fault diagnosis completion.

Algorithm 3 PURE (Fault Diagnosis with Integrated Priority-driven Recovery)

```
Require: $N$, $N_s$, $\mathbf{P}$
Ensure: $\hat{\Phi}$, $F^*$
1: Initialize $\hat{\Phi} = [x\ x\ x\ ...\ x]^T$, $i = 1$, $N_a = N - N_s$
2: Arrange elements of $V$ in ascending order of $\mathbf{P}$
3: while ($\{k|k \in \hat{\Phi}, k = x\} \neq \phi$) //Until all suspect nodes are proven to be healthy do
4:     while ($\{k|k \in \hat{\Phi}_s, k = 0\} = \phi$) //Identify at least one healthy node in $V_s$ do
5:         Designate $v_s$ as checker(s) $(N_a + 1) \leq s \leq (N_a + N_s)$ ; thus $V_s = \{v_s\}$
6:         while $i \leq N_a$ do
7:             Reconfigure RS(s) with the same functionality as $v_i$, $N_{sup} = N_{sup} + 1$
8:             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
9:             $i \leftarrow i + 1$
10:        end while
11:        Move the RS by updating $N_a = N_a - N_s$, $N_r = N_r + 1$, Re-initialize $i = 1$
12:    end while
13:    Identify the most prioritized function computing node which is faulty, $v_{pf}$
14:    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 $V_a$
15: end while
```

7.1 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:

$$t_r(G) = d_r(G) - 1 \tag{7}$$

where $d_r(G)$ is the average degree of nodes in the graph at r. The topology at $r = 1$ in Fig. 6 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. 7 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.

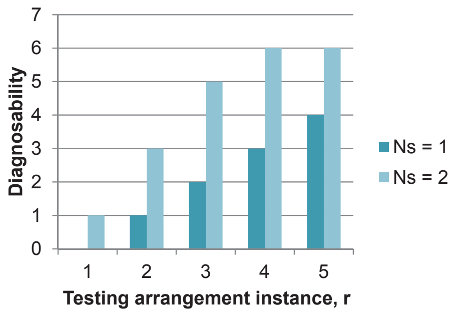


Fig. 7 The diagnosability of a topology with various reconfiguration iterations

Fig. 8 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.

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

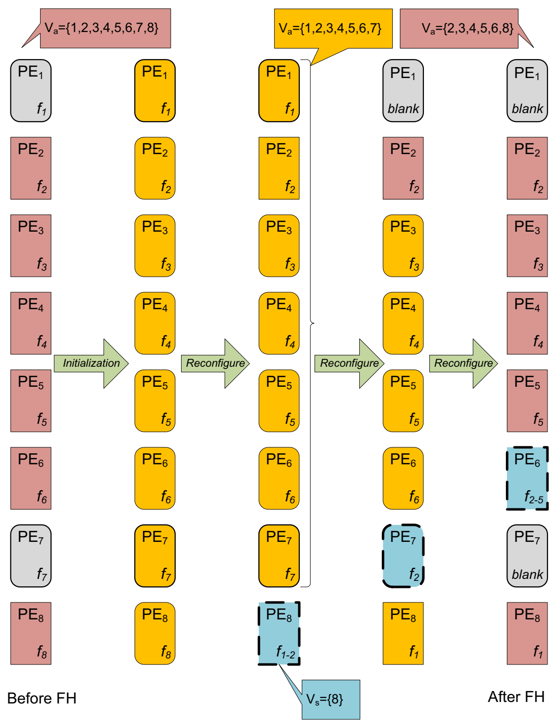


Fig. 8 An example of fault diagnosis in PURE approach

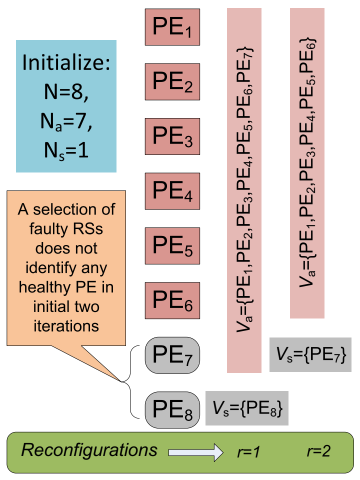

Initialize: N=8, Na=7, Ns=1
A selection of faulty RSs does not identify any healthy PE in initial two iterations
Va={PE1,PE2,PE3,PE4,PE5,PE6,PE7}
Va={PE1,PE2,PE3,PE4,PE5,PE6}
Vs={PE7}
Vs={PE8}
Reconfigurations
r=1
r=2

Fig. 9 The worst case scenario for the diagnostic phase with two defective nodes

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 2, 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. 9 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. 9, as many as 3 reconfiguration iterations are required as the first two slacks selected were faulty.

7.2 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.

7.2.1 PSNR as a Health Metric

PSNR is well-established metric to assess the relative quality of video encoding [45]. 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 [44]. 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 [19] [46] [47], [2]. For example, in [19] 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 [46] and [2], PSNR is used to quantify the graceful degradation achieved in a Motion Estimation engine, DCT application, and an Inverse DCT circuit as the

Table 2 Latency vs. throughput comparisons for the fault-handling schemes in terms of % resources available during diagnosis

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%

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. 10 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.

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. 13 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:

$$\Delta_{PSNR} = 100 \times \frac{(PSNR_{avg} - PSNR_{current})}{PSNR_{avg}} \quad (8)$$

For example, Table 3 and Table 4 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 3 and Table 4 together show how a desirable Δ_{FD} value could trade-off 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 5 lists the effectiveness of using these detection parameters with a variety of input benchmarks.

Table 3 Effect of $\Delta_{FD} = 3\%$ tolerance using Failure-Free Resources for city.qcif, QP=5

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 4 Effect of $\Delta_{FD} = 3\%$ tolerance using PEs with 5% degraded output for city.qcif, QP=5

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 summarizes the combinations of conditions under which PSNR is a reliable indicator of faults.

Table 5 Fault detection performance ($\Delta_{FD} = 3\%$) using PEs with 5% degraded output at frame 51, QP=5

	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 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

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 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 [48].

7.2.2 Output Discrepancy as a Health Metric

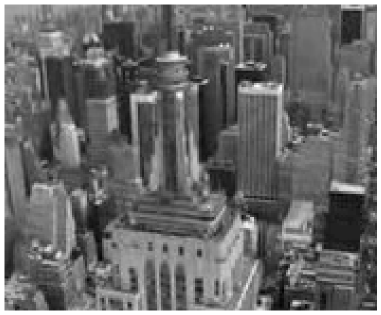
When a health metric such as PSNR is readily available, it can be used to reduce area and power overheads. However, for applications where such health metric is not feasible, PURE can utilize CED and priority information without loss of generality. Thus, to detect hardware faults at the local DCT level 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 7.3 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 [49] where a low area overhead estimator is used in lieu of multiple instances of the fully redundant datapath.

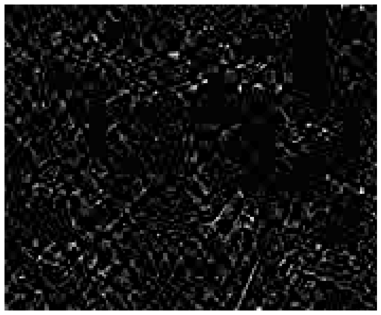
In summary, use of either a PSNR-based or discrepancy-based health metric can be used to initiate the PURE diagnostic flow. Nonetheless, PURE provides the designer with the freedom to choose the number of RS and the period between reconfigurations based on area, power, and fault-detection latency tradeoffs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

7.3 PURE Functional Testing as Compared to Physical Resource Testing

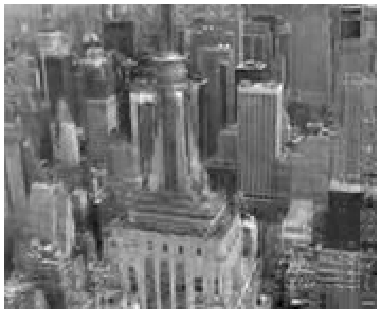
There are a number of distinctions between PURE and physical resource testing techniques. For instance, the STARs approach mentioned in Section 2 provides a useful and established online BIST approach to diagnosis of FPGA Logic Resources by Abramovinci, Stroud, and Emmert [50], [36]. Both techniques focus on providing fault coverage while maintaining useful throughput.



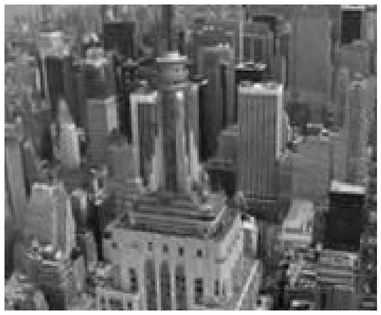
(a) Image in frame buffer computed using healthy PEs, PSNR=35.27dB



(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

Fig. 10 The impact of faults on PSNR and image quality

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 [51]. 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:

$$N_{sup,max} = 1 + \sum_{s=N-N_d}^{N-1} s \tag{9}$$

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. 11 shows the upper bound on diagnosis latency using $N_s = 1$. Fig. 12 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.

Table 7 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 [52] 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

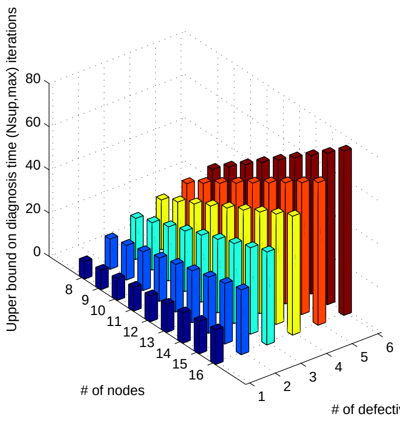


Fig. 11 Diagnosis latency of the PURE approach for $N_s = 1$

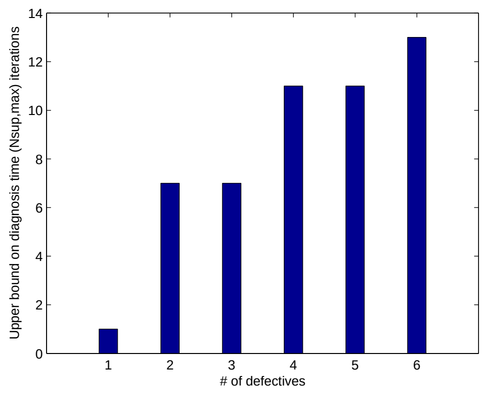


Fig. 12 Diagnosis latency of the PURE approach for $N_s = 2$, $N = 8$

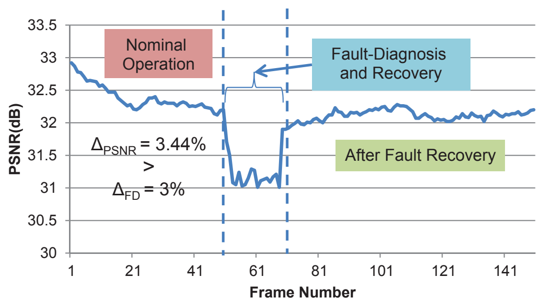
Table 7 Configuration bitstream sizes in DCT core

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

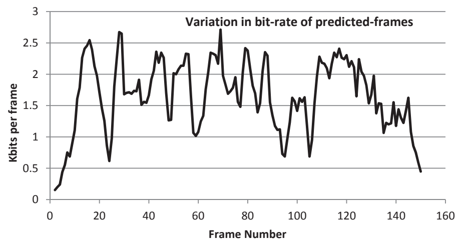
latency can be improved by utilizing more PEs for the comparison purposes at runtime.

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 [53] [54] can benefit in saving the memory requirement. In [53], 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 [54] by exploiting the relocatable modules.



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



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

Fig. 13 PSNR and bit-rate of the encoder employing PURE

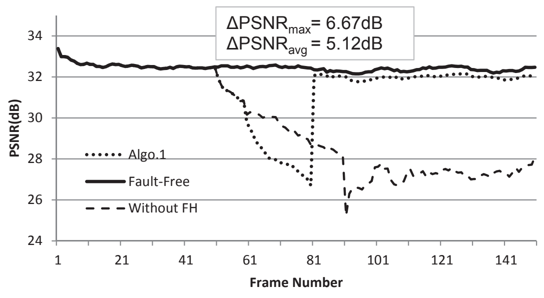
8 Experimental Results

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.

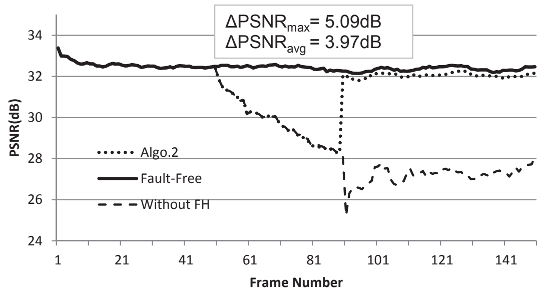
8.1 Case Study-1: Prioritized elements of the DCT core

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.

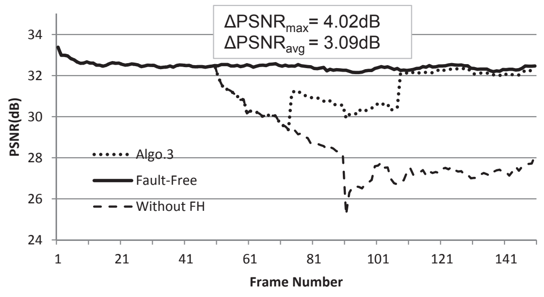
The priority of functions is naturally in descending order as the DC-coefficient contains most content information of a natural image. These 8 PEs in the processing throughput datapath are covered by the proposed resilience scheme. For this purpose, depending upon area/power margin available, RSs are created at design-time or generated at runtime considering the priority of functions. As shown in Fig. 14, fault-handling is performed at runtime with a small quality degradation during diagnosis process. The diagnosis time of Algorithm 1 is very short, however,



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



(b) Fault-handling using Algorithm 2: FaDReS



(c) Fault-handling using Algorithm 3: PURE

Fig. 14 Operational examples of the three algorithms indicating reduced PSNR degradation for PURE

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. 13 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.

8.2 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 [55] in the context of the proposed fault-diagnosis methodology using a verilog core [56]. For this purpose, the encryption module of 128-bit AES is synthesized and implemented in Xilinx Integrated Software Environment (ISE) 13.4 development environment for Virtex-7 `xc7v2000t` device. *Stuck-At* 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)* [57] 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 8. For the given case of 8 inputs, a total of 4 outputs being faulty are observed.

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 9. For the synthesized design, minimum clock period is 1.821ns (Maximum Frequency 549.058MHz). The size of each partial reconfiguration bitstream file

Table 8 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*

Table 9 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

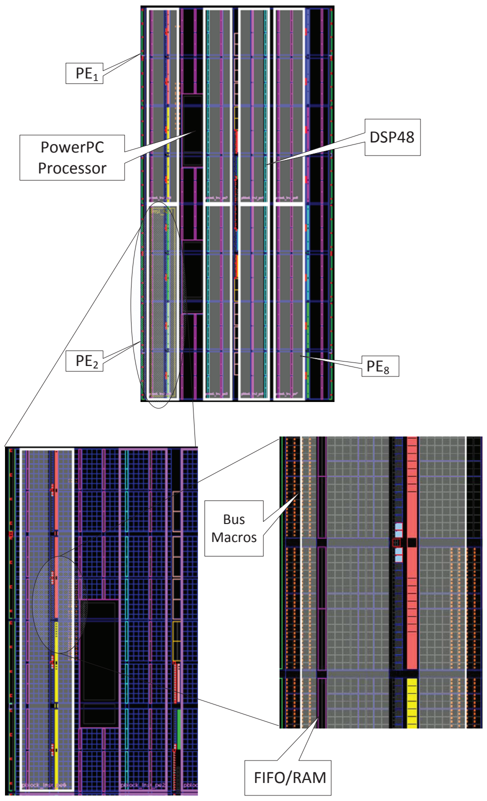


Fig. 15 Floorplan of the AES core for Virtex-4 chip

is 112.8KB. Fig. 15 shows the floorplan of AES core. This allows the PURE approach to occupy only $\frac{1}{N}$ area overhead for N PEs.

8.3 Energy Duty Cycle

Time-Dependent Dielectric Breakdown (TDDB) and Electromigration (EM) are two significant causes of permanent faults over the device lifetime [58]. 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 [58], [59], [60].

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 [58] 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 [61]. 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. 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. 14. 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.

9 Conclusions

PURE provides an adaptive dynamic reconfiguration approach to achieve survivability. 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.

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. Future work is 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.

References

1. Cho, Hyungmin, Larkhoon Leem, and S. Mitra. 2012. ERSA: Error resilient system architecture for probabilistic applications. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 31 (4): 546–558.
2. Karakonstantis, Georgios, Debabrata Mohapatra, and Kaushik Roy. 2012. Logic and memory design based on unequal error protection for voltage-scalable, robust and adaptive DSP systems. *Journal of Signal Processing Systems (JSPS)* 68: 415–431.
3. Whatmough, P. N., S. Das, D. M. Bull, and I. Darwazeh. 2013. Circuit-level timing error tolerance for low-power DSP filters and transforms. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 21 (6): 989–999.
4. Mitra, S., and E. J. McCluskey. 2000. Which concurrent error detection scheme to choose ? In *International test conference*, 985–994.
5. Mitra, Subhasish, W. J. Huang, N. R. Saxena, S. Y. Yu, and E. J. McCluskey. 2004. Reconfigurable architecture for autonomous self-repair. *Design Test of Computers, IEEE* 21 (3): 228–240.
6. Rao, Wenjing, A. Orailoglu, and R. Karri. 2006. Nanofabric topologies and reconfiguration algorithms to support dynamically adaptive fault tolerance. In *VLSI test symposium*, 6.
7. Juan A. Gmez-Pulido, Miguel A. Vega-Rodrguez, and Juan M. Snchez-Prez. 2012. High-speed reconfigurable parallel system to design good error correcting codes in communications. *Journal of Signal Processing Systems* 66: 147–152.
8. Kthiri, Moez, Hassen Loukil, Ahmed Atitallah, Patrice Kadionik, Dominique Dallet, and Nouri Masmoudi. 2012. FPGA architecture of the LDPS Motion Estimation for H.264/AVC Video Coding. *Journal of Signal Processing Systems* 68: 273–285.
9. Imran, Naveed, and Ronald F. DeMara. 2011. A self-configuring TMR scheme utilizing discrepancy resolution. In *International conference on reconfigurable computing and FPGAs (ReConFig)*, 398–403.
10. Rubin, Raphael, and André DeHon. 2009. Choose-your-own-adventure routing: lightweight load-time defect avoidance. In *Proceedings of the ACM/SIGDA international symposium on FPGAs*, 23–32. New York, NY, USA: ACM. ISBN 978-1-60558-410-2.
11. Berg, M., C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K. A. LaBel, M. Friendlich, H. Kim, and A. Phan. 2008. Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis. *Nuclear Science, IEEE Transactions on* 55 (4): 2259–2266.
12. Stoica, Adrian, Didier Keymeulen, Ricardo Zebulum, Mohammad Mojarradi, Srinivas Katkoori, and Taher

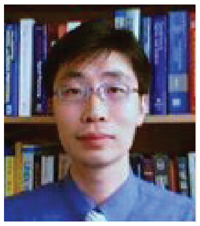
Daud. 2007. Adaptive and evolvable analog electronics for space applications. In *Proceedings of the 7th international conference on evolvable systems: from biology to hardware. ICES'07*, 379–390. Berlin, Heidelberg: Springer. ISBN 3-540-74625-0, 978-3-540-74625-6.

13. Pereira, M. M., L. Braun, M. Hubner, J. Becker, and L. Carro. 2011. Run-time resource instantiation for fault tolerance in FPGAs. In *NASA/ESA conference on adaptive hardware and systems (AHS)*, 88–95.
14. Siozios, Kostas, and Dimitrios Soudris. 2012. Low-cost fault tolerant targeting FPGA devices. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS), special session on dependability by reconfigurable hardware.*
15. Palem, Krishna V., Lakshmi N. B. Chakrapani, Zvi M. Kedem, Avinash Lingamneni, and Kirthi Krishna Muntimadugu. 2009. 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*, 1–10. New York, NY, USA: ACM. ISBN 978-1-60558-626-7.
16. Chippa, V. K., D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar. 2010. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In *47th ACM/IEEE design automation conference (DAC)*, 555–560.
17. Mohapatra, Debabrata, Georgios Karakonstantis, and Kaushik Roy. 2009. Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator. In *14th ACM/IEEE international symposium on low power electronics and design (ISLPED)*, 195–200. New York, NY, USA: ACM. ISBN 978-1-60558-684-7.
18. Abdallah, R. A., and N. R. Shanbhag. 2010. Minimum-energy operation via error resiliency. *Embedded Systems Letters, IEEE* 2 (4): 115–118.
19. Kim, E. P., and N. R. Shanbhag. 2012. Soft N-Modular redundancy. *Computers, IEEE Transactions on* 61 (3): 323–336.
20. Narayanan, S., G. V. Varatkar, D. L. Jones, and N. R. Shanbhag. 2010. Computation as estimation: A general framework for robustness and energy efficiency in SoCs. *Signal Processing, IEEE Transactions on* 58 (8): 4416–4421.
21. Varatkar, G. V., and N. R. Shanbhag. 2008. Error-resilient motion estimation architecture. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 16 (10): 1399–1412.
22. Greenwood, G. W. 2005. On the practicality of using intrinsic reconfiguration for fault recovery. *Evolutionary Computation, IEEE Transactions on* 9 (4): 398–405.
23. Laprie, J-C. 1995. Dependable computing and fault tolerance : Concepts and terminology. In *25th international symposium on fault-tolerant computing - highlights from twenty-five years.*
24. Malek, Miroslaw. 1980. A comparison connection assignment for diagnosis of multiprocessor systems. In *Proceedings of the 7th annual symposium on computer architecture (ISCA)*, 31–36. New York, NY, USA: ACM.
25. Preparata, Franco P., Gernot Metze, and Robert T. Chien. 1967. On the connection assignment problem of diagnosable systems. *Electronic Computers, IEEE Transactions on* EC-16 (6): 848–854.
26. DeMara, Ronald F., Kening Zhang, and Carthik A. Sharma. 2011. Autonomic fault-handling and refurbishment using throughput-driven assessment. *Applied Soft Computing* 11: 1588–1599.
27. Carmichael, Carl. 2006. Triple module redundancy design techniques for virtex FPGAs. *Xilinx Application Note: Virtex Series XAPP197 (v1.0.1), July 6, 2006.*
28. Kastensmidt, F. L., L. Sterpone, L. Carro, and M. S. Reorda. 2005. On the optimal design of triple modular redundancy logic for SRAM-based FPGAs. In *Design, Automation and Test in Europe*, 1290–12952.
29. Sloan, J., D. Kesler, R. Kumar, and A. Rahimi. 2010. A numerical optimization-based methodology for application robustification: Transforming applications for error tolerance. In *IEEE/IFIP international conference on dependable systems and networks (DSN)*, 161–170.
30. Gao, Ming, Hsiu-Ming (Sherman) Chang, Peter Lisherness, and Kwang-Ting (Tim) Cheng. 2011. Time-multiplexed online checking. *IEEE Transactions on Computers* 60 (9): 1300–1312.
31. Stott, E., P. Sedcole, and P. Cheung. 2008. Fault tolerant methods for reliability in FPGAs. In *International conference on field programmable logic and applications (FPL)*, 415–420.
32. Garvie, M., and A. Thompson. 2004. 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)*, 155–160.
33. Keymeulen, D., R. S. Zebulum, Y. Jin, and A. Stoica. 2000. Fault-tolerant evolvable hardware using field-programmable transistor arrays. *Reliability, IEEE Transactions on* 49 (3): 305–316.
34. Emmert, J. M., C. E. Stroud, and M. Abramovici. 2007. Online fault tolerance for FPGA logic blocks. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 15 (2): 216–226.
35. Dutt, S., V. Verma, and V. Suthar. 2008. 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* 27 (2): 309–326.
36. Abramovici, M., C. E. Stroud, and J. M. Emmert. 2004. Online BIST and BIST-based diagnosis of fpga logic blocks. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 12 (12): 1284–1294.
37. Gericota, M. G., G. R. Alves, M. L. Silva, and J. M. Ferreira. 2008. Reliability and availability in reconfigurable computing: A basis for a common solution. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 16 (11): 1545–1558.
38. Mizan, E., T. Amimeur, and M. F. Jacome. Oct. 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*, 45–53.
39. Paulsson, K., M. Hubner, and J. Becker. 2006. 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).*
40. Koren, I., and S. Y. H. Su. 1979. Reliability analysis of n-modular redundancy systems with intermittent and permanent faults. *Computers, IEEE Transactions on* C-28 (7): 514–520.
41. Dorfman, Robert. 1943. The detection of defective members of large populations. *The Annals of Mathematical Statistics* 14 (4): 436–440.
42. Litvak, Eugene, Xin M. Tu, and Marcello Pagano. 1994. Screening for the presence of a disease by pooling sera

samples. *Journal of the American Statistical Association* 89 (426): 424–434.
43. Smith, Cedric A. B. 1947. The counterfeit coin problem. *The Mathematical Gazette* 31 (293): 31–39.
44. Imran, N., J. Lee, and R. F. DeMara. 2013. Fault demotion using reconfigurable slack (FaDReS). *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 21 (7): 1364–1368.
45. Wiegand, T., H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan. 2003. Rate-constrained coder control and comparison of video coding standards. *Circuits and Systems for Video Technology, IEEE Transactions on* 13 (7): 688–703.
46. Karakonstantis, G., D. Mohapatra, and K. Roy. 2009. System level DSP synthesis using voltage overscaling, unequal error protection & adaptive quality tuning. In *IEEE workshop on signal processing systems (SiPS)*, 133–138.
47. Karakonstantis, G., N. Banerjee, and K. Roy. 2010. Process-variation resilient and voltage-scalable DCT architecture for robust low-power computing. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 18 (10): 1461–1470.
48. Bucciero, M., J. P. Walters, and M. French. 2011. Software fault tolerance methodology and testing for the embedded PowerPC. In *Proceedings of the IEEE aerospace conference*, 1–9. Big Sky, MT.
49. Imran, Naveed, and Ronald F. DeMara. 2011. Heterogeneous concurrent error detection (hCED) based on output anticipation. In *International conference on reconfigurable computing and FPGAs (ReConFig)*, 61–66.
50. Abramovici, M., J. M. Emmert, and C. E. Stroud. 2001. 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*, 73–92.
51. Huang, Wei-Je, S. Mitra, and E. J. McCluskey. 2001. Fast run-time fault location in dependable FPGA-based applications. In *IEEE international symposium on defect and fault tolerance in vlsi systems (DFT)*, 206–214.
52. Xilinx. Partial reconfiguration user guide. UG702 (v14.3) October 16, 2012.
53. Becker, T., W. Luk, and P. Y K Cheung. 2007. Enhancing relocatability of partial bitstreams for run-time reconfiguration. In *15th annual IEEE symposium on field-programmable custom computing machines (FCCM)*, 35–44.
54. Huang, Jian, and Jooheung Lee. 2011. Reconfigurable architecture for ZQDCT using computational complexity prediction and bitstream relocation. *Embedded Systems Letters, IEEE* 3 (1): 1–4.
55. 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`.
56. Drimer, Saar. 2009. Security for volatile FPGAs. Phd dissertation, Univeristy of Cambridge, 15 JJ Thomson Avenue Cambridge CB3 0FD United Kingdom.
57. Sharma, Carthik A., Alireza Sarvi, Ahmad Alzahrani, and Ronald F. DeMara. 2013. Self-healing reconfigurable logic using autonomous group testing. *Microprocessors and Microsystems* 37 (2): 174–184.
58. Srinivasan, S., R. Krishnan, P. Mangalagiri, Yuan Xie, V. Narayanan, M. J. Irwin, and K. Sarpatwari. 2008. Toward increasing FPGA lifetime. *Dependable and Secure Computing, IEEE Transactions on* 5 (2): 115–127.
59. Poivey, Christian, Melanie Berg, Scott Stansberry, Mark Friendlich, Hak Kim, Dave Petrick, and Ken LaBel. June 2007. Heavy ion SEE test of Virtex-4 FPGA XC4VFX60 from Xilinx. Retrieved on January 8, 2012 [Online] `http://radhome.gsfc.nasa.gov/radhome/papers/T021607_XC4VFX60.pdf`.
60. Bolchini, C., and C. Sandionigi. 2010. Fault classification for SRAM-Based FPGAs in the space environment for fault mitigation. *Embedded Systems Letters, IEEE* 2 (4): 107–110.
61. Trace. Video trace library: YUV 4:2:0 video sequences. Retrieved on January 20, 2012 [Online] `http://trace.eas.asu.edu/yuv/`.

Naveed Imran received the M.S. degree in Electrical Engineering from the University of Central Florida (UCF), Orlando, Florida, USA in 2010. Currently, he is a Ph.D. candidate in the Department of Electrical Engineering and Computer Science at UCF. His research interests include hardware design of DSP systems, FPGAs, reconfigurable hardware for image/video applications, and reliable VLSI architectures.

Ronald F. DeMara received the Ph.D. degree in Computer Engineering from the University of Southern California. Since 1993, he has been a full-time faculty member at the University of Central Florida. His research interests are in Computer Architecture with emphasis on Evolvable Hardware and Distributed Architectures for Intelligent Systems. He has published approximately 130 articles on these topics and holds one patent. He is a Senior Member of IEEE and a Member of ACM, and ASEE, and has served on the Editorial Board of IEEE Transactions on VLSI Systems. In 2008, he received the Outstanding Engineering Educator Award in the Southeastern United States from IEEE.



Jooheung Lee has been working on various topics in the areas of multimedia signal processing algorithms and low power VLSI systems design. His research interests include image and video coding algorithms, multimedia systems, power aware and reliable VLSI systems design, and reconfigurable computing for signal processing applications. Previously, he worked at the Wireless Multimedia Communications Laboratory at the R&D Complex of LG Electronics in 1998, where he worked on low power video codec ASIC design for mobile applications. After completing his Ph.D. at the Pennsylvania State University in 2006, he joined the Department of Electrical Engineering and

Computer Science at the University of Central Florida, Orlando, Florida, USA, where he was a full-time faculty member. Currently, he is an Associate Professor of the Department of Electronic and Electrical Engineering at Hongik University, Republic of Korea.

Jian Huang is currently with Advanced Micro Devices. He got Ph.D. degree in electrical engineering at the University of Central Florida in 2010. He got his master degree (M.S.) in computer engineering from the University of North Texas in 2007. His research interests include VLSI Architecture for cryptography, unified and scalable architecture for image and video coding, and reconfigurable computing.