Abstract

In this paper, we propose an online redundancy approach which considers fault-isolation latency and throughput trade-offs under a dynamic spare arrangement. The diagnosis process can be described by a time-varying graph which shows that our algorithm for isolation of faulty nodes incurs small throughput degradation. Fault coverage is provided for both functional modules as well as checker elements through an iterative diagnosis process aimed at improved system availability and survivability for long duration mission-critical applications. For instance, a case study of the H.263 video encoder indicates that the PSNR is maintained above 29.5dB during fault-handling phase while the diagnosis latency is 196 video frames. The results show considerable benefits over Triple Modular Redundant (TMR) arrangement in terms of fault-resiliency, power consumption, and resource-area requirements. By tackling aging-induced degradation failures in Field Programmable Gate Arrays (FPGAs), the availability is improved to 99.999%. Furthermore, the proposed setup consumes only 33% of TMR’s power for 99.999% of the mission-period, while 65.7% of TMR’s power for 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 while requiring only $\frac{1}{N}$ area overhead for $N$ PEs.

Keywords

System availability, group testing, aging-induced degradation, functional testing, reconfigurable slack, autonomous recovery.

N. Imran and R. F. DeMara are with the 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).
ACRONYMS

RN  Reconfigurable Node
RS  Reconfigurable Slack
PE  Processing Element
FE  Functional Element
DMR  Dual Modular Redundancy
TMR  Triple Modular Redundancy
NMR  N-Modular Redundancy
CED  Concurrent Error Detection
FPGAs  Field Programmable Gate Arrays
SEU  Single Event Upset
SET  Single Event Transient
SA  Stuck At
FD  Fault Detection
FI  Fault Isolation
FR  Fault Recovery
FH  Fault Handling
QP  Quantization Parameter
TDDB  Time-Dependent Dielectric Breakdown
EM  Electromigration
PSNR  Peak Signal-to-Noise Ratio
PRR  Partial Reconfiguration Region
FT  Fault Tolerance
AES  Advanced Encryption Standard
ISE  Integrated Software Environment
HDL  Hardware Description Language
NUT  Node(s) Under Test
QoS  Quality of Service
SNR  Signal-to-Noise Ratio
I. INTRODUCTION

Dynamic redundancy techniques have been widely used to increase reliability of critical systems in which reconfiguration is employed at runtime to utilize spare units in response to failures [1] [2] [3]. While some techniques rely on pre-allocation of dedicated spare units, a dynamic spare pool sharing approach can be favorable in terms of extending fault-capacity [4]. Redundancy enables fault-tolerance, however, how wisely redundancy is employed at runtime determines the sustainability of the system exposed to cumulative failures. Aggressive scaling of semiconductor technology to cope with today’s intensive computing demands leads to seek new autonomous reliability approaches for logic devices.

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

The regular structure of an FPGA-fabric is amenable to reconfiguration-based recovery. A high regularity of FPGA logic resources allows movement of a function implemented over a defective region to a fault-free region [5] [6] [7] [8]. FPGAs are popular among space exploration community for its reconfigurability [9] [10]. On the other hand, FPGAs are also susceptible to soft (transient) errors as well as hard (permanent) faults. The reliability problem of highly complex VLSI systems in sub-90 nanometer process, caused by soft and hard errors, is increasing. Therefore, the importance of addressing reliability issues is growing to achieve high integration level, performance, and transistors density on 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 level. 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 spared at runtime to perform diagnosis while others reserved at runtime can serve as backups for faulty PEs. Without loss of generality, we term these PEs as Reconfigurable Slack (RS).

II. RELATED WORK

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 the system. System-Level Diagnosis is a widely used technique for fault resilience in multiprocessor systems. In comparison diagnosis model [11] [12] [13] [14] a pair of units is evaluated subjected to the same inputs and a discrepancy indicates some failure. The impact of a topology on diagnosability of a network is thoroughly discussed in [15] [16]. For this paper, we will consider a fully connected topology so that the diagnosis can be performed between any pair of nodes. Then, after identifying a faulty node, it can be replaced by any of the available healthy nodes.

Fault tolerant systems typically employ a sequence of resolution phases including Fault Detection (FD), Fault-Diagnosis, Fault Isolation (FI), and Fault Recovery (FR). A system can be considered to be fault-tolerant if it continues operation in the presence of failures, perhaps in a degraded mode with partially restored functionality [17]. Reliability and availability are desirable qualities of a system, which are measured in terms of service continuity and operational availability in presence of interrupts, respectively [18]. In this paper, we show that reliability is attained by employing the reconfigurable modules in the Fault Handling (FH) flow, whereas the availability is maintained by the minimum interruption of the main throughput datapath.

The redundancy based methods are popular among fault-tolerant systems community, with costs of area and power
overhead. A Concurrent Error Detection (CED) setup either realizes two concurrent replicas of a design [19], or a diverse duplex to avoid common mode faults [2]. Its advantage is a very low fault detection latency. A Triple Modular Redundancy (TMR) system utilizes three instances of a datapath module. The outputs of these three instances become input 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. Another method of online fault detection involves periodic testing of the components in a given module. Although the resource requirement may not be very large, the fault detection latency may be long depending upon the chip area, assuming the system of interest is an electronic circuit. These drawbacks can be 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 as described further.

The Fault Diagnosis and Isolation 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. Our active dynamic redundancy approach combines the benefits of redundancy with a negligible computational overhead. Meanwhile, static redundancy techniques usually reserve dedicated spare resources for FH. In contrast, in the PAREs approach, the redundant modules are continually utilized in the datapath during the normal mission operation. As described below, PAREs introduces redundant resources only when needed based on runtime conditions.

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 FH. 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. Use of algorithm-level information allows PAREs to remain scalable on designs with millions of gate-equivalent circuits.

Reliability of FPGA based designs can be achieved in various ways. Offline testing methods rely on taking the device out of operation, diagnosing the faulty resources and avoiding those resources in the configured design. However, this method is less practical for real-time systems having specific timing deadlines, or mission critical systems in which device outage may be problematic to the mission. On the other hand, Online testing methods, such as online Built-in Self Test (BIST) techniques [20] check a small area of the chip in concurrence while keeping the remaining non-tested regions in operation. Resource testing typically involves pseudo-exhaustive input-space testing of the physical resources to identify faults, while functional testing methods check the fitness of the datapath functions [21]. In previous work, Dutt et. al [21] proposed a scheme in which the output of a Programmable Logic Block (PLB) is compared to that of an identically configured PLB. A discrepancy in the output flags the PLB as faulty. Gericota et. al’s active replication technique [22] concurrently creates replicas of Configurable Logic Blocks (CLBs) to improve the reliability.
In contrast, PAREs performs functional testing of the resources at higher granularity, however any prior thorough knowledge of a PE’s function is not required for FI. The scheme presented in [21] requires certain areas of the FPGA to be designated as spare for the Roving Tester (ROTE). In PAREs, the testing components remain part of the functional datapath until otherwise demanded by the FH procedure. Upon fault detection, these resources are designated for FI purposes. Later, upon the completion of FI, the reconfigurable slacks may then be recommissioned to perform priority functions. Fault recovery in FPGA based circuits is sometimes achieved via configuration level repair [23]. Some resources are kept as a standby to cope with a faulty situation. In event of a fault, these backup resources replace the used faulty resources of the design. The presented approach employs module level repair [1].

Evolvable hardware techniques focus on adapting hardware to achieve fault tolerance. These methods rely on finding a configuration which meets fitness criteria under a given fault scenario. A Competitive Runtime Reconfiguration (CRR) [19] scheme uses evolution to repair the faulty resources of a CED arrangement. We can divide these evolutionary schemes into two types 1) Design-time fault tolerance 2) Runtime fault tolerance. The focus in design time fault tolerance is to build circuits which are robust to faults in different components, yet the disadvantage is that fault recovery is limited to only anticipated faults. On the other hand, the focus in runtime fault tolerance schemes is to recover during runtime operation. Using the dynamic partial reconfiguration capability and the presented recovery sequence, PAREs is able to achieve a high degree of runtime fault tolerance.

Table I provides a comparison of previous approaches towards FH in FPGA based systems. The TMR technique involves a triplication of the design where the three copies of a system are active continuously. The fault capacity is limited to the faults within one copy only. Moreover, self-repair feature can provide increased sustainability compared to using redundancy to achieve fault tolerance [24]. In contrast to the passive techniques of fault tolerance, Active
techniques incorporate control schemes involving some intelligent actions to cope with a fault event. Evolutionary techniques [25] repair the circuit at a finer granularity, yet lack a guarantee that a recovery would be obtained within a certain number of generations. It may require hundreds of GA generations before finding an optimal solution, thus undesirably extending the recovery time. On the other hand, PAREs 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 is required to select the best individuals in a population, which may in turn necessitate the input-output truth table knowledge of the circuit. PAREs avoids both of these complications. Altogether, they allow PAREs to evaluate to the actual inputs, instead of an exhaustive set of inputs, on any commercial off-the-shelf FPGA with partial reconfiguration capability.

### III. Fault-Handling (FH) Method

The FH process is divided into various phases: FD, fault confinement, fault-diagnosis, FI, and FR. Fault-detection can be either performed by continuously observing a health metric like Signal-to-Noise Ratio (SNR), or checking the processing nodes by round-robin manner. The diagnosis phase consists of evaluating the PEs to actual inputs of the system. Given diagnosis data, the faulty elements are identified in the FI phase. An objective here is to maintain the throughput during diagnosis phase, and then the isolation phase can be completed in a very short time as we employ the on-chip processing power of FPGA device. The formulation of isolation process and the reconfiguration schedule for diagnosis process are discussed in the following sections respectively while Fig. 1 illustrates an overall organization of the paper. Section IV employs a syndrome function considering the connectivity to identify faulty nodes in a network. Section V identifies three algorithms for reconfiguration scheduling to diagnose faulty nodes under constraint of throughput availability during diagnosis phase. The algorithms are evaluated by implementing H.263 video encoder’s DCT hardware core in Hardware Description Language (HDL) as described in Section V-D. Section V-E reports fault injection and analysis results for AES encryption engine in addition to the area overhead of

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Approach</th>
<th>Emphasis</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>IMR</td>
<td>Fault-masking</td>
<td>area-cost</td>
</tr>
<tr>
<td>1</td>
<td>Divide and Conquer</td>
<td>Rapid isolation, Fault-confinement</td>
<td>Low Throughput maintainability</td>
</tr>
<tr>
<td>2</td>
<td>FaDReS</td>
<td>Availability vs. Diagnosis latency, Runtime trade-off</td>
<td>Medium Throughput maintainability</td>
</tr>
<tr>
<td>3</td>
<td>PAREs</td>
<td>High Throughput maintainability</td>
<td>Diagnosis latency</td>
</tr>
</tbody>
</table>

**Benefits**
- Online Functional Diagnosis
- Graceful degradation of PSNR
- Low energy duty-cycle
- Survivability
the RS based fault handling scheme. The energy duty cycle results as compared to the conventional TMR approach are provided in Section V-G. In the following, we will use the terms nodes and PEs interchangeably.

IV. OBJECTIVE-1: IDENTIFICATION OF FAULTY NODES

Although the comparison diagnosis model has been widely employed in fault-tolerant computing community [15], we propose time-varying topology of a flexible redundancy diagnosis model exploiting reconfiguration capability of FPGAs. Time varying topology has been previously proposed in [26] for a power system network. Given an undirected graph \( G(V, E) \) of vertex set \( V \) and edges set \( E \), the task is to identify faulty nodes. The nodes of \( G \) correspond to either PEs or processors in a multiprocessor network connected through an interconnection network. The diagnosis process is based upon the comparisons to identify discrepancy while we do not restrict the discussion to a pair-wise comparison. Instead, the fault isolation process can utilize N-Modular Redundancy (NMR) in accordance with availability of resources. 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 \( \Phi \), and the fitness states estimated based upon the fault-isolation process by vector \( \hat{\Phi} \).

The following assumptions are made in the proposed fault isolation scheme.

1) Faults are of permanent nature.

2) While comparing outputs, a faulty node manifests its fault at least once in a given Evaluation Window period in order to be observable.

3) The outcome of a comparison is positive if at least one of the nodes in a CED pair is faulty.

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 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 faults-scenario. We use the Connectivity Matrix \( C \) to show the comparison performed between two nodes in \( G \). Thus, an entry \( c_{ij} = 1 \) denotes that a comparison between node \( i \) and node \( j \) is performed. Syndrome Matrix \( \Psi \) indicates the outcome of comparisons. A value of ’1’ as an entry at index \( i \) and \( j \) of this matrix shows the discrepancy between node \( i \) and node \( j \). Both of these matrices are considered symmetric here.

\[
\Psi = \begin{bmatrix}
0 & \psi_{12} & \cdots & \psi_{1N} \\
\psi_{21} & 0 & \cdots & \psi_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
\psi_{N1} & \psi_{N2} & \cdots & 0
\end{bmatrix}
\] (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.

We used the syndrome matrix to estimate the fitness states of nodes in \( G \). Thus, the isolation of faulty nodes is performed based upon the syndrome matrix values. Following is the condition for healthiness, and estimated fitness vector is updated accordingly:
Condition: $\psi(i, j) = 0$ for any $1 \leq j \leq N$, where $i \neq j$ and $c_{ij} = 1$

Update: $\hat{\phi}(i) = 0$

In addition, the syndrome matrix is used to update the fitness of various PEs based upon diagnosis history information. When a healthy RS is found in a given slack update iteration, it indicates that previously selected slacks were faulty.

In the proposed recovery schemes, priority of tasks is taken into account while recovering from fault scenarios. For 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 $P$, 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.

V. OBJECTIVE-2: RECONFIGURATION SCHEDULE TO DEMOTE FAULTY NODES

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 Node(s) Under Test (NUT) based upon their discrepancy history.

In the following, we describe some variations of the FH phase. As a baseline method, divide and conquer approach has been evaluated. The choice of algorithm in an application depends upon the designer’s preferences about isolation latency, throughput availability requirement, and area/power trade-offs.
A. Divide and Conquer Method

Group testing schemes have been successfully employed to solve the problems of fault isolation in which the number of defective is smaller than the size of the overall pool. Dorfman [27] first proposed the idea of pool testing when an efficient algorithm was sought for screening positive blood samples of syphilis disease from a pool of samples collected from military inductees during World War II. In this approach, a large pool of samples is tested first, and if the result is positive then all the individuals are tested in the second stage; otherwise, a negative outcome of the test obviates the need to test individual samples as it implies negative or healthy nature of all the samples in the pool under test. Counterfeit coin identification problem is also similar to the problem of identifying small number of defective in a large pool [28], [29]. Litvak et. al proposes a hierarchical algorithm of group testing in which positive pools are successively divided into halves until all defective items are identified [30]. McMahan, Tebbs, and Bilder [31] consider also the case when the outcome of a test is probabilistic, and they improved the prediction accuracy by identifying misdiagnosed individuals.

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, the absence of a golden functional output element for testing an individual node makes the hierarchical testing up to the last single item infeasible. Therefore, we also rely on 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 FH phase, then halving of the suspect pool [30] offers logarithmic time diagnosis latency.
- 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 FH 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.

The number of edges in the graph of Fig. 2 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. 3 in which dotted lined boxes correspond to the checking slacks and solid lined correspond to active PEs. Algorithm 1 defines the diagnosis process.

To analyze the diagnosability of $G$ by divide and conquer method, we observe from Fig. 2(b) that every node has three adjacent nodes. In the worst case, if all the adjacent nodes of a node $i$ become faulty, then it is impossible to check the fitness of node $i$ using a comparison diagnosis model. In that case, the system is no longer diagnosable. However, if only two adjacent nodes of a presumed healthy node $j$ are faulty, then the remaining one node can be used for checking purposes. Thus, the diagnosability $t$ of a divide and conquer topology is $(d(G) - 1)$ where $d(G)$
**Algorithm 1** Fault Diagnosis Algorithm (Divide and Conquer)

**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 \( C \)
4. Update the Syndrome Matrix \( \Psi \)
5. Iterate steps from one to four \( \log(N) \) times
6. Given \( \Psi \), isolate the faulty nodes:
   \[
   \hat{\phi}_i \leftarrow 0 \text{ and } \hat{\phi}_j \leftarrow 0, \text{ if } c_{ij} = 1, \text{ and } \psi_{ij} = 0
   \]
   \[
   \hat{\phi}_i \leftarrow 1 \text{ if } \hat{\phi}_j = 0, \ c_{ij} = 1, \text{ and } \psi_{ij} = 1
   \]

is the average degree of a node in \( G \). For example, the syndrome matrix for faulty PE\(_4\) and PE\(_6\) is given by:

\[
\Psi = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
1 & 0 & 0 & x & 0 & x & x & x \\
2 & 0 & 0 & x & 1 & x & 1 & x \\
3 & 0 & x & 0 & 1 & x & 0 & x \\
4 & x & 1 & 1 & 0 & x & x & 1 \\
5 & 0 & x & x & x & 0 & 1 & 0 \\
6 & x & 1 & x & 1 & 0 & x & 1 \\
7 & x & x & 0 & x & 0 & x & 0 \\
8 & x & x & 1 & x & 1 & 0 & 0
\end{bmatrix}
\]

An entry \( \psi_{12} = 0 \) denotes the healthy nature of PE\(_1\) and PE\(_2\) while \( \psi_{42} = 1 \) shows faulty nature of at least one of the PEs in the pair under test. After fault detection, all the entries of \( \Psi \) except those on the diagonal are initialized with \( x \) implying that the health of all the PEs is suspect.
B. Fault Demotion using Reconfigurable Slack (FaDReS)

To maintain availability during the FH process, a diagnosis algorithm should avoid excessive reconfiguration of the processing datapath. Thus, we present FaDReS algorithm [32] which checks the individual nodes of the throughput path either by comparison or voting model while keeping them in the datapath.

1) Diagnosis by voting: The algorithm for diagnosis employing dynamic NMR voting on module level as given in [33] is provided below after adopting to the graph terminology presented in this paper. Fig. 4 shows various steps in the diagnosis process.

Algorithm 2 Fault Diagnosis Algorithm

**Require:** \( N, N_s, P \)

**Ensure:** \( \Phi \)

1: Initialize \( \Phi = [x \ x \ x \ ... \ x]^T \), \( i = 1 \)
2: Arrange elements of \( V \) in ascending order of \( P \)
3: while \( \{ k | k \in \Phi, k = 0 \} = \phi \) 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 N-Modular Redundancy (NMR) majority voting among NUT and update Connectivity matrix accordingly,
   Update the Syndrome matrix \( \Psi \) 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 \( N_a = N_a - N_s \), \( N_{recon} = N_{recon} + 1 \), 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 \( \psi_{ij} = 1 \)
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 \( \psi_{ij} = 0 \)

Fault diagnosis latency \( T_{diag} \) is defined as:

\[
T_{diag} = (T_{eval} + T_{rec}N_s) \sum_{j=1}^{N_{recon}} I_j
\]

(2)

Where

\( N_{recon} = \) Number of iterations before the FD completes

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

\( T_{rec} = \) Reconfiguration Latency (PR time for one PE)
\[ V_a = \{PE_1, PE_2, PE_3, PE_4, PE_5, PE_6, PE_7 \} \]
\[ V_s = \{PE_8 \} \]

\[ V_a = \{PE_1, PE_2, PE_3, PE_4, PE_5, PE_6, PE_7 \} \]
\[ V_s = \{PE_8 \} \]

\[ V_a = \{PE_1, PE_2, PE_3, PE_4, PE_5 \} \]
\[ V_s = \{PE_6 \} \]

**Fig. 4. Fault Diagnosis in the FaDReS Approach**

**Fig. 5. Upper bound on maximum number of slack updates required for various faulty scenarios, \( N = 9 \)**

\( N_s \) = Number of Reconfigurable Slacks

\( T_{eval} \) = Evaluation Window

An upper bound on the latency of the fault-diagnosis is given as:

\[ T_{diag, max} = (T_{eval} + T_{rec}N_s) \sum_{j=1}^{N_a} (N_a - j) \]  

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 previous 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.
We employ a locality constraint to quantify the distinction between the voting approach and comparison approach. In case of Xilinx FPGAs, the Internal Configuration Access Port (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. 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 $2 \times T_{rec}$ but just $(1 + \beta) \times T_{recon}$ where $\beta$ is a small number depending upon the ratio between internal on-chip memory access time and external memory access time. On the other hand, a comparison diagnosis approach requires $2 \times T_{rec}$ reconfiguration time for two slacks as both need to be configured as separate functions. The preference of one over the other method should be based upon reconfiguration time $T_{recon}$, $\beta$ factor, and evaluation window period $T_{eval}$.

C. Functional Priority Allocation using Integrated Diagnosis and Recovery Approach

The distinction between this algorithm and the FaDReS arises from the fact that after a healthy RS is identified, it is used for priority function computation immediately in the former case. On the other hand, 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.

In the PAREs 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 $G$, and is given by the equation:

$$t_r(G) = d_r(G) - 1$$
Algorithm 3 Integrated Diagnosis and Recovery Approach

Require: $N, N_s, P$

Ensure: $\hat{\Phi}, F^*$

1: Initialize $\hat{\Phi} = [x \ x \ x \ ... \ x]^T, \ i = 1$
2: Arrange elements of $V$ in ascending order of $P$
3: while $\{k | k \in \hat{\Phi}, k = x\} \neq \phi$ //Until all suspect nodes are proven to be healthy do
4: \hspace{1cm} while $\{k | k \in \hat{\Phi}_s, k = 0\} = \phi$ //Identify at least one healthy node in $V_s$ do
5: \hspace{2cm} Designate $v_s$ as checker(s) $(N_a + 1) \leq s \leq (N_a + N_s)$ ; thus $V_s = \{v_s\}$
6: \hspace{2cm} while $i \leq N_a$ do
7: \hspace{3cm} Reconfigure RS(s) with the same functionality as $v_i, N_{sup} = N_{sup} + 1$
8: \hspace{3cm} Perform N-Modular Redundancy (NMR) majority voting among NUT and update Connectivity matrix accordingly, Update the Syndrome matrix $\Psi$ 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: \hspace{3cm} $i \leftarrow i + 1$
10: \hspace{2cm} end while
11: \hspace{1cm} end while
12: \hspace{1cm} Move the RS by updating $N_a = N_a - N_s, N_{recon} = N_{recon} + 1$, Re-initialize $i = 1$
13: \hspace{1cm} end while
14: \hspace{1cm} Identify the most prioritized function computing node which is faulty, $v_{pf}$
15: \hspace{1cm} 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 $V_a$
16: end while

where $d_r(G)$ is the average degree of nodes in the graph at $r$. The topology at $r = 1$ in Fig. 4 is 0-diagnosable since a faulty RS leaves all other nodes suspect after comparisons. However, the topology defined by $C$ 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 into identification of defective nodes in a few iterations.

Fig. 8 shows an illustrative example of the fault diagnosis in PAREs approach. The faulty PEs 1 and 7 are shown by dotted line blocks. In the initialization phase of FH algorithm, the fitness state of all the PEs is suspect as shown by gray blocks. Then, PE$_s$ is reconfigured as RS by implementing function $f_1$ and its output is compared with that of PE$_1$ to check for any discrepancy. 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$_s$ is reconfigured to second priority function $f_2$ and its discrepancy check is performed with PE$_2$ which implements $f_2$. An agreement
Fig. 7. The diagnosability of a topology with various reconfiguration iterations

![Diagnosability diagram](image)

**TABLE II**

<table>
<thead>
<tr>
<th>Reconfiguration instance, $r$</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_a$ during diagnosis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algo. 1</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Algo. 2</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Algo. 3</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>No. of bitstream downloads, $N_{sup}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algo. 1</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Algo. 2</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Algo. 3</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

reveals their healthy nature. In addition, it shows that PE$_1$ was faulty as it had exhibited discrepancy with a healthy PE (i.e., PEs$_8$) 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 PEs$_8$ 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. Last, 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 FR is achieved by configuring faulty PEs by blank and healthy PEs by functions 1 through 6.

For a total of $N$ nodes in $G$, there are $N^2 - N$ possible pairings. As evident by Table II, our fault-diagnosis schemes require significantly fewer comparisons compared to the exhaustive pair evaluations. Fig. 9 shows the worst case scenarios for the three algorithms in an FE containing 8 PEs. The gray boxes correspond to faulty PEs. As shown in second and third column of the figure, as many as 3 reconfiguration iterations are required as the first two slacks selected were faulty.
D. 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 $8 \times 8$ 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. The priority of tasks 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, RS are created at design-time or generated at runtime considering the priority of tasks. As shown in Fig. 10, FH is performed at runtime with a small quality degradation during diagnosis process. The diagnosis time of Algorithm 1 is very short, however, this greedy approach incurs quality degradation during FH process. The quality degradation during FH process is improved in Algorithm 2 at the cost of some diagnosis latency. Algorithm 3 provides the best availability of the system during FH and the
Fig. 9. The worst case scenarios with two defective nodes for three algorithms

(a) FH using Algorithm 1
(b) FH using Algorithm 2
(c) FH using Algorithm 3

Fig. 10. Operational examples of the three algorithms

(a) PSNR of a video sequence
(b) Bit-rate of the video sequence

Fig. 11. PSNR and bit-rate of the encoder employing PAREs

PSNR is maintained above 29.5dB. Fig. 11 shows the PSNR and bit-rate of video stream from an operational encoder before, during, and after FH demonstrating PAREs approach.
E. Case Study-2: Fault Resilience of a Multicore Design

Next, we evaluated Advanced Encryption Standard (AES) [34] in the context of the proposed fault-diagnosis methodology. For this purpose, the encryption module of 128-bit AES is synthesized and implemented in Xilinx ISE 13.4 development environment for Virtex-7 xc7v2000t device. 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) which invokes various commands of the Xilinx flow to study fault behavior. An example of injecting SA fault to one of the LUT-inputs is shown in Fig 12. Then, the circuit is evaluated to some test inputs. The test outputs, corresponding actual fault-free outputs, and the outcome in terms of actual AES functionality are listed in Table III. For the given case of 8 inputs, a total of 4 outputs are faulty. Thus, an evaluation window of size 2 input samples is sufficient to reveal any discrepant behavior of the AES module for this particular implementation.

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 over a Virtex-4 xc4vfx60-12ff1136 device is listed in Table IV. For the synthesized design, minimum clock period is 1.821ns (Maximum Frequency 549.058MHz). Each PRM’s bitstream size is 112.8KB.

The presence of a health metric facilitates FD since the metric can be continuously observed for any contingency.
TABLE III
FAULT IMPACT IN 128 AES COMPUTATIONAL FE

<table>
<thead>
<tr>
<th>Actual Output</th>
<th>True Output</th>
<th>Test Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>66e94bd4ef0a2c3b884cfa59ca342b2e</td>
<td>66e94bd4ef8a2c3b884cfa59ca342b2e</td>
<td>Fail</td>
</tr>
<tr>
<td>3ad78e726c1ec02b7ebfe92b23d9ec34</td>
<td>3ad78e726c1ec02b7ebfe92b23d9ec34</td>
<td>Pass</td>
</tr>
<tr>
<td>45bc707d2968204d8dfbaf08b0cad9b</td>
<td>45bc707d29e8204d8dfbaf28b0cad9b</td>
<td>Fail</td>
</tr>
<tr>
<td>161556838018f52805c6db6202002e3f</td>
<td>161556838018f52805c6db6202002e3f</td>
<td>Pass</td>
</tr>
<tr>
<td>f55693ab626d11efdedb0a64c6854af55693ab626d11efdedb0a64c6854a</td>
<td>f55693ab626d11efdedb0a64c6854a</td>
<td>Fail</td>
</tr>
<tr>
<td>64e82b50e501fbd7dd4116921159b83e</td>
<td>64e82b50e501fbd7dd4116921159b83e</td>
<td>Pass</td>
</tr>
<tr>
<td>baac12fbb613a7de11150375c74034041</td>
<td>baac12fbb613a7de11150375c74034041</td>
<td>Pass</td>
</tr>
<tr>
<td>bcf176a7ea2d8085ebace362462a281</td>
<td>bcf176a7eaad8085ebace362462a281</td>
<td>Fail</td>
</tr>
</tbody>
</table>

TABLE IV
UTILIZATION SUMMARY OF THE AES DESIGN

<table>
<thead>
<tr>
<th>Logic Resource</th>
<th>Utilization by a PE</th>
<th>Utilization by a Reconfigurable PE</th>
<th>Capacity of a PRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>416</td>
<td>1021</td>
<td>1024</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>625</td>
<td>1778</td>
<td>4096</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>726</td>
<td>1236</td>
<td>4096</td>
</tr>
<tr>
<td>Number of FIFO16/RAMB16s</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Number of DSP48s</td>
<td>16</td>
<td>0</td>
<td>varies</td>
</tr>
</tbody>
</table>

For example, in case of a video encoder, Peak Signal-to-Noise Ratio (PSNR) measure of the image in frame buffer indicates the correctness of computation. However, for the applications that do not have such measures, a round-robin diagnosis of the computational hardware may be performed continuously or periodically. This provision, however, incurs some cost in terms of power consumption and fault-detection latency. Considering the diagnosis latency and power consumption trade-off as demonstrated in [7], the period of the comparison checking may be selected.

F. Time Complexity of the PAREs algorithm

An upper bound on the diagnosis time is defined in terms of the maximum slack update iterations required to isolate \( N_d \) faulty nodes in a network of \( N \) nodes employing a single RS, and is given by:

\[
N_{\text{sup,max}} = 1 + \sum_{s=0}^{N-N_d} s
\]

For example, given a network of size \( N = 8 \) and \( N_s = 1 \), maximum number of slack updates occur in the case of faulty PE7 and PE8. Thus, \( N_{\text{sup,max}} = 1 + 7 + 6 \). The constant term 1 is added to include the reconfiguration required to identify a healthy slack. Fig. 14 shows the upper bound on diagnosis latency using \( N_s = 1 \). 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. Fig. 15 shows the diagnosis latency when using two slacks.
for a network of size $N = 8$. To diagnose a single defective node, as few as two slack updates are required as compared to the previous case requiring a maximum of 8 slack updates when only one slack was employed.

G. Energy Duty Cycle

Time-Dependent Dielectric Breakdown (TDDB) and Electromigration (EM) are two significant causes of permanent faults in long-term mission critical systems [35]. To quantify the survivability of the system employing the proposed FH flow, FD, FI, and FR times are considered here. The availability is generally defined as:

$$\text{Availability} = \frac{MTTF}{MTTF + MTTR}$$ (4)

In this analysis, we consider hard faults only caused by TDDB and EM. The impact of radiations and aging induced degradation on reliability of FPGA-based circuits has been analyzed by authors in [35], [36], [37] [38], [39]. 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 [35] for MCNC benchmark circuits simulating their 12 years behavior. Considering, 312 utilized
LUTs in a PE spanning one Partial Reconfiguration Region (PRR) which contains total 1152 LUTs, the 10.2% failure rate means 32 LUTs fail per year. In the worst case scenario, suppose that the failure rate is uniformly distributed over time. Thus, for a minimum of one LUT failure, the MTTF is 11 days.

The MTTR is the sum of times required for FD, diagnosis, FI, and FR. To assess the FD time, faults are injected into the DCT module at frame number 50 of the news.qcif video sequence. As a result, the PSNR drops below 31dB threshold at frame number 59. Thus, FD time is 0.3 seconds for 30fps frame rate. For a systems of $N = 8$ PEs, the latency of fault-diagnosis can be computed by using eq. 3. Using one slack, the maximum cost is 196 frames or 6.5 second for a frame rate of 30 fps. The FI time 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 reconfiguration during FR period is 1.6 seconds. Thus, total time to repair for this particular example is 8.4 seconds. With these values of MTTF and MTTR, the availability is 99.999%.

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, an alternative approach of Fault Tolerance (FT), that is, TMR would consume about 216mW dynamic power throughout the mission-lifetime if the voter’s overhead is neglected. Thus, the proposed scheme consumes only 33% of this power for 99.999% of the mission-period, while 65.7% of TMR’s power for 0.001% mission.

VI. Conclusions

We proposed a novel scheme of fault-handling to meet survivability objective for digital systems. Active dynamic redundancy realizes autonomicity while gracefully maintaining desired quality of service measure under area resource, power, and energy constraints. The uniplex assertion applies if a signal-to-noise metric is known, as well as applications that do not possess a readily correlated metric to identify anomalous behavior. In addition, readily available processor cores allow dynamic fault identification by executing a software specification of the signal processing algorithm which is used to periodically validate critical outputs of the high-speed hardware circuit within tolerances. As a future work, we plan to extend our scheme to achieve fault-resilience in general multiprocessor networks and reconfigurable smart grid power systems.

REFERENCES


