Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2014, Article ID 279673, 21 pages
http://dx.doi.org/10.1155/2014/279673

Research Article

Distance-Ranked Fault Identification of Reconfigurable Hardware Bitstreams via Functional Input

Naveed Imran and Ronald F. DeMara

Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA

Correspondence should be addressed to Naveed Imran; naveed@knights.ucf.edu

Received 29 September 2013; Revised 26 December 2013; Accepted 9 January 2014; Published 17 March 2014

Academic Editor: Walter Stechele

Distance-Ranked Fault Identification (DRFI) is a dynamic reconfiguration technique which employs runtime inputs to conduct online functional testing of fielded FPGA logic and interconnect resources without test vectors. At design time, a diverse set of functionally identical bitstream configurations are created which utilize alternate hardware resources in the FPGA fabric. An ordering is imposed on the configuration pool as updated by the PageRank indexing precedence. The configurations which utilize permanently damaged resources and hence manifest discrepant outputs, receive lower rank are thus less preferred for instantiation on the FPGA. Results indicate accurate identification of fault-free configurations in a pool of pregenerated bitstreams with a low number of reconfigurations and input evaluations. For MCNC benchmark circuits, the observed reduction in input evaluations is up to 75% when comparing the DRFI technique to unguided evaluation. The DRFI diagnosis method is seen to isolate all 14 healthy configurations from a pool of 100 pregenerated configurations, and thereby offering a 100% isolation accuracy provided the fault-free configurations exist in the design pool. When a complete recovery is not feasible, graceful degradation may be realized which is demonstrated by the PSNR improvement of images processed in a video encoder case study.

1. Introduction

The self-reconfiguration capability of FPGAs has been identified as a useful feature for realizing designs which are resilient to local permanent faults as well as mitigating transistor aging degradations [1]. Recovery from local permanent damage in FPGA-based designs can be realized by reconfigurations to utilize fault-free logic resources at runtime. Given some faulty resources in a particular region on an FPGA chip, the lost functionality can be refurbished by utilizing a pristine area of the chip. Conversely, if a circuit realized by a particular bitstream manifests an observable fault, then an alternate bitstream utilizing only fault-free resources can be downloaded into a reconfigurable region.

A *Concurrent Error Detection (CED)* scheme [2] is a well-established low-latency spatial-redundancy approach to fault detection. Such circuits are instantiated with a single replicated module to realize a *Duplex Modular Redundancy (DMR)* arrangement. When a discrepancy is observed in the output, it reveals the faulty nature of at least one of the instances of the duplex arrangement. If autonomous recovery capability is desired, then after *fault detection*, an efficient *fault recovery* technique is sought which is the subject of this paper.

A previous technique which uses a diverse pool of FPGA configurations for recovery from local permanent damage in online FPGAs is the *Consensus-Based Evaluation (CBE)* [3] method. CBE generates a diverse population of circuit configurations utilizing alternative device resources. When discrepancies are detected, the configurations in the pool are evaluated using online inputs in a duplex arrangement. Each configuration has an associated *Discrepancy Value (DV)* metric which is increased based upon the discrepancy history of that configuration. This evaluation process increases the DV of configuration pairs which exhibit discrepant behavior to identical inputs in a given *Evaluation Period*. Thus, those configurations utilizing faulty resources accumulate a higher DV than those which utilize fault-free resources. CBE used

statistical clustering of DV values to identify outliers. A consensus is formed and the fitness of an individual configuration relative to the consensus value differentiates the faulty configuration in order to initiate repair. In the current work, our primary goal is to rapidly identify the operationally correct circuits out of a population of permanent fault-affected circuits using a search-driven ranking scheme. Secondary goals are recovery from for multiple permanent faults, while realizing graceful degradation.

There is a body of research dealing with the problem of identifying faulty components by employing system diagnosis theory. A pioneering work in diagnosis theory is by Preparata et al. [4] in which the problem of identifying faulty nodes in a digital system is formulated as a connection assignment procedure. Various components of a digital system are represented by nodes in a graph described by a *connection matrix*. A given edge in the graph connects two nodes, one being the node *under test* and the other being *testing* node. The diagnosibility of digital systems containing faulty modules has been studied by various researchers [5, 6]. In the proposed scheme, the configuration bitstreams for reconfigurable hardware fabric are conceptualized as nodes of the graph which represents digital circuits undergoing diagnosis.

The novel diagnosis technique proposed herein offers several benefits. It avoids the explicit step of faulty resource isolation and does not take the device offline to enable exhaustive or pseudo-exhaustive testing of all possible sets of inputs. It does not require testing the device resources individually which may contribute to unobservable faults. Instead, the actual throughput inputs are utilized for the evaluation of an online FPGA device. Various design configurations are evaluated in pairs. The pairs are selected from a pool of designs which are functionally identical and yet utilize alternate hardware resources. Ideally, some circuit throughput is maintained during the fault handling process, and thereby providing the potential of partially correct output during recovery for some applications such as signal processing tasks. These techniques are developed and evaluated herein to mitigate local permanent damage by functionally evaluating the Circuit Under Test (CUT). A unique hardware configuration of a CUT corresponds to a set of logic and interconnect resources allocated to realize each CUT in the proposed approach.

The remainder of the paper is organized as follows. In Section 2, related work on fault handling in live FPGA devices is presented. Section 3 describes the design flow of generating a diverse pool of configurations. Section 4 defines the online fault-handling process in terms of two subphases consisting of fault detection and fault diagnosis. Section 5 describes the DRFI fault-handling flow in detail. Here, an analogy between online ranking of a pool of configurations according to their runtime fitness is provided in the context of online search techniques. Section 6 includes the fault recovery results for two classes of case studies: MCNC benchmark circuits and DCT cores. Section 7 compares the proposed scheme with a traditional TMR technique as well as describing the overhead involved in DRFI over a uniplex design. Finally, Section 8 concludes the paper while also identifying useful future directions.

2. Related Work

In order to mitigate local permanent damage, *Fault-handling (FH)* systems typically employ a sequence of handling phases including *Fault Detection*, *Fault Diagnosis*, and *Fault Recovery*. By employing these phases at runtime, a fault-resilient system can continue its operation in the presence of failures, sometimes in a degraded mode with partially restored functionality [7] if the full restoration of functionality is infeasible. The systems reliability and availability are measured in terms of *service continuity* and *operational availability* in presence of adverse events, respectively [8]. In this paper, an increase in reliability is sought by employing reconfigurable hardware in the fault-handling flow, whereas increased availability is sought by instantiating the preferred configurations in the throughput datapath.

Redundancy-based fault detection methods are widely used in fault-handling systems although incur costs of area and power overhead. In the *Comparison Diagnosis Model* [4, 9], a module pair is evaluated subjected to the inputs to check for any discrepancy. For example, a CED arrangement employs either two replicas of a design or a diverse duplex design to reduce common mode faults [2]. Its advantage is a very low fault detection latency. A *Triple Modular Redundancy (TMR)* system [10–12] utilizes three instances of a throughput module. The outputs of these three instances are passed through a majority voter circuit, which in turn provides the main output of the system. In this way, in addition to 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 arrangement incurs an increased area and power overhead to accommodate the replicated datapaths. It will be shown that these overheads can be significantly reduced by employing reconfiguration.

The *Fault Diagnosis* phase consists of identifying properly functioning computational resources in some larger set of *Suspect* resources. Traditionally, in many fault tolerant digital circuits, the resources 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 and 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 DRFI runtime reconfiguration approach combines the benefits of redundancy with only twice the computational requirement while significantly maintaining the throughput in presence of hardware failures.

While reconfiguration and redundancy are fundamental components of a fault recovery process, both the choice of reconfiguration scheduling policy and the relative fitness of computational modules affect the availability during the recovery phase and quality of recovery, after fault handling. Here, it is possible to maintain reasonable levels of availability by instantiating preferred configurations during the fault-handling phase and by promoting the relatively higher-ranked configurations after fault recovery.

Reliability of FPGA-based designs [13] can be achieved in various ways. TMR is popular in FPGA-based reliable designs for protection against permanent as well as transient

faults. For instance, a vendor's tool *XTMR* is available to triplicate the user logic in Xilinx devices [10]. The TMR's fault recovery capability is limited to the faults which impact one module. This limitation of TMR can be overcome using *self-repair* [14, 15] approaches to increase sustainability, such as refurbishing the failed instance using *jiggling* [16] technique of faults mitigation. Other *active recovery* techniques incorporate control schemes which realize intelligent actions to cope with permanent failures. Keymeulen et al. [17] proposed an *evolutionary* approach to circumvent the faults in reconfigurable digital and analog circuits. They proposed genetic algorithms to evolve the population of fault tolerant circuits by applying genetic operators such as mutation and crossover over the circuit's bitstream representation. Other self-repair techniques by Garvie and Thompson [16] are based upon direct bitstream manipulation by evolutionary algorithms to recover from faults. Many evolvable hardware techniques have been presented in the literature that rely on intricate details of the FPGA device structure and routing. Their recovery times may be extensive as the design tools must be invoked at runtime to generate new alternatives or in some cases nonconvergent based on the stochastic nature of genetic algorithm based search. In addition, a fitness evaluation function must be defined in advance to select the best individuals in a population, which may in turn necessitate complete knowledge of the input-output truth table for fault-free behavior. DRFI avoids both of these complications. Altogether, DRFI is able to utilize actual inputs, instead of exhaustive or pseudo-exhaustive test vectors, on any commercial off-the-shelf FPGA with partial reconfiguration capability without runtime invocation of the design tools.

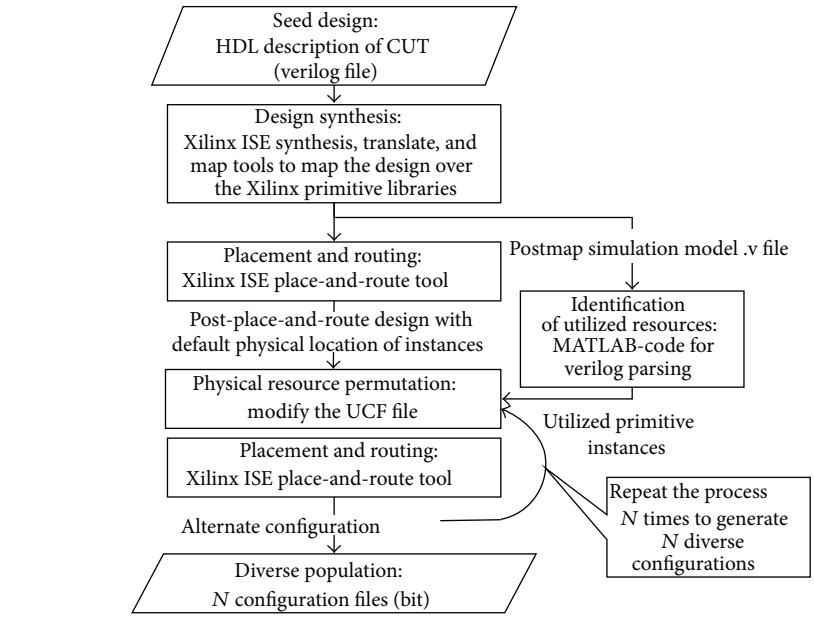
One approach to reducing overheads associated with TMR is to employ the Comparison Diagnosis Model with a pair of modules in an adaptable CED arrangement subjected to the same inputs. For example, the *Competitive Runtime Reconfiguration (CRR)* [18] scheme uses an initial population of functionally identical (same input output behavior), yet physically distinct (alternative design or place-and-route realization), FPGA configurations which are produced at design time. At 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 in subsequent operations.

Other runtime testing methods, such as online *Built-in Self-Test (BIST)* techniques [19], offer the advantages of a roving test, which checks a subset of the chip's resources while retaining the remaining nontested 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 correctness of the datapath functions [20]. In [21], a pair of blocks configured with identical operating modes are subjected to resource-oriented test patterns. This *Self-Testing AReas (STARs)* approach maintains a relatively small area of the reconfigurable fabric offline which is under test, while the remainder of the reconfigurable fabric is online and continues its operation. STARs compare the output of each Programmable Logic Block (PLB) to that of an identically configured PLB. This utilizes the property that a discrepancy between the outputs alerts the PLB to be suspected as outlined by Dutt et al.'s *Roving Tester (ROTE)* technique [20]. Gericota et al.'s active replication technique [22] concurrently creates replicas of *Configurable Logic Blocks (CLBs)*. In the STARs approach, each block under test is successively evaluated in multiple reconfiguration modes, and when a block is completely tested, 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 halt 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. The heterogeneous nature of FPGA resources, for example, LUTs, FFs, BRAMS, multipliers, DSP Blocks, and processor cores, can make it challenging to achieve a generic testing methodology based on a roving approach. Moreover, the scalability of resource-based testing techniques with the significant growth of on-chip resources is also a concern. Therefore, functional testing can offer an appealing alternative to resource-based testing and thus it is embraced herein. In this paper, we concentrate on a live FPGA testing scenario. Nonetheless, backend testing with the proposed technique is also possible, although BIST schemes are generally preferred in such situations due to their fine-grained resolution which is beneficial for backend testing.

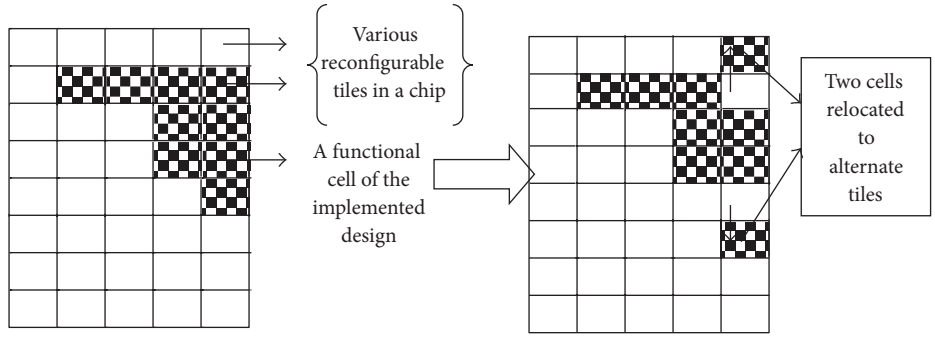
3. Generating a Diverse Pool of Configurations by Design Relocation

A diverse population of configurations which randomly employ different resources within the FPGA fabric is relatively straightforward to generate at design time [23]. For this purpose, the seed design, which is a post-placed-and-routed circuit, is relocated to alternate areas in a chip. Modifying the User Constraints File (`.ucf`) can constrain the place-and-route tool to generate alternate configurations. Each distinct `.ucf` file in Xilinx ISE environment corresponds to a diverse physical configuration, and thus the generated configuration bitstream (`.bit`) file is unique. The process is described in detail below.

As shown in Figure 1(a), the circuit is specified using Verilog HDL and mapped to a Xilinx FPGA chip by the vendor-provided synthesis tool. The location of the design components mapped over corresponding logic resources is determined by the synthesis and implementation toolset itself. The Xilinx *Integrated Software Environment (ISE)* placement tool automatically places the design components considering the area and timing optimizations. The post-place-and-route simulation model is considered as a seed design here. The chip area is divided into various *Reconfigurable Tiles (RT)* where each tile may contain multiple *Partial Reconfiguration Regions (PRRs)* [24]. The distinction between RT and PRR allows changing the granularity of fault handling during runtime. The design is partitioned into basic *Logic Cells*. The initial locations of all Logic Cells are obtained from the seed



(a) Relocation algorithm mapped over the tool flow



(b) A sample relocated design

Figure 1: Generating various configurations by design relocation.

design. For assigning alternate *Reconfigurable Tiles* to the logic cells, the *User Constraints File (UCF)* file [25] is modified to relocate the circuit components and then the circuit is reinstantiated. This results in an alternate configuration utilizing unused tiles of the chip as shown in Figure 1(b). In this manner, a diverse set of configurations is generated which utilize alternate logic resources in the chip. In the current design tool suite from Xilinx, the term PRR has been renamed to *Reconfigurable Partition* and the requirement to insert bus macros has been voided by proxy-LUTs [26, 27]. Our design relocation flow supports current versions of Xilinx design suites ISE, for example, ISE 14.7, and Vivado.

4. Online Fault Handling by Dynamic Reconfiguration

4.1. Fault Detection. A pair of configuration bitstreams is randomly selected to instantiate a CED arrangement in the FPGA device. Only those configuration pairs can be instantiated which utilize mutually exclusive device resources. Mutual exclusion can be ensured by virtually dividing the chip into two distinct regions, one for each CED instance. An instantiated pair provides the desired DMR which can be used for error detection as illustrated in Figure 2. In the following discussion, the configurations in a DMR arrangement are referred to as the active CUT and RS corresponding to Reconfigurable Slack, respectively.

A discrepancy between the outputs of the two instances in DMR arrangement reveals the faulty nature of at least one of those configurations. Afterwards when a discrepancy in the outputs occurs, a *fault detection* condition is asserted and the proposed fault-handling methodology is initiated. The problem of identifying healthy configurations out of suspected configurations is then formulated as a system-level diagnosis problem.

4.2. System-Level Diagnosis of Hardware Configurations. The fault(s) occurring in an FPGA chip may impact multiple circuit implementations in the configuration pool. Thus, after fault detection, the health of all of the configurations is

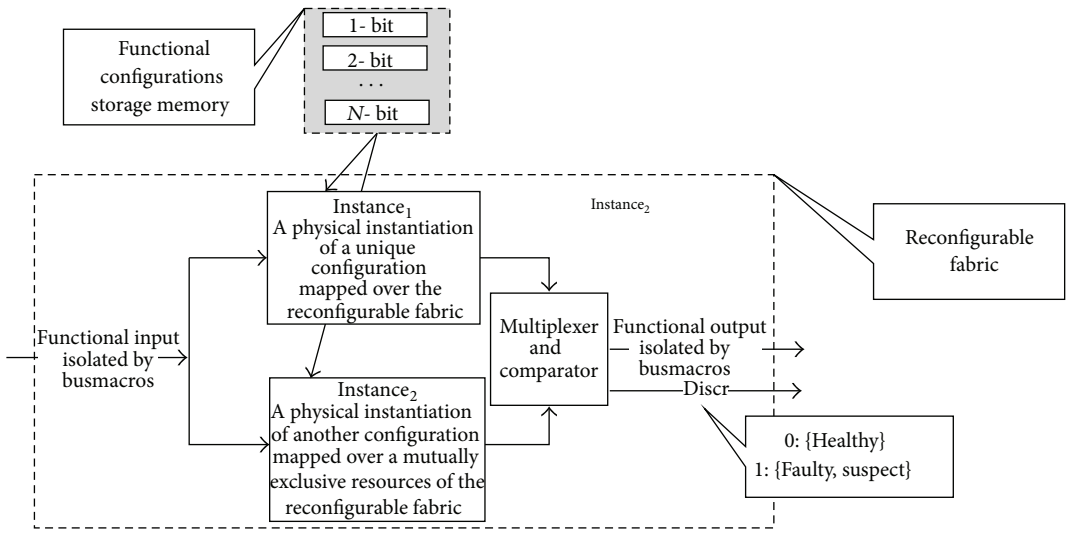


Figure 2: A CED arrangement of a functional element.

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

Figure 3 outlines the scope, diagnosis approaches, and metrics used below. The diagnosability formulation for identifying faulty nodes is developed herein using a syndrome function. The three diagnosis algorithms of *Exhaustive Evaluation approach*, *the State Transitions approach*, and *the DRFI approach employing PageRank technique* developed are described in Sections 4.2.1, 4.2.2, and 5, respectively. Section 6 reports experimental results for MCNC benchmark circuits and H.263 video encoder's DCT hardware core.

The same diagnosis formulation applies to each of the three algorithms developed and is described first here. Given an undirected graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ of vertex set $\mathbf{V}$ and edges set $\mathbf{E}$, the diagnosis objective is to identify faulty nodes. The nodes of $\mathbf{G}$ correspond to configurations to be compared in a CED arrangement. The diagnosis process is described in terms of CED comparisons to identify discrepancies. However, the formulation is not restricted to a pair-wise comparison. Instead, the fault diagnosis process can utilize N-Modular Redundancy (NMR) in accordance with availability of resources. NMR is a generalization of TMR where $N \geq 2$ modules provide $N - 1$ redundant instances, which has found applicability in adaptive fault handling [28, 29].

An element (u, v) in the edge set $\mathbf{E}$ indicates the feasibility that the output from corresponding configurations 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 $\widehat{\Phi}$. We define the *Connectivity Matrix* $\mathbf{C}$ to show the comparison performed between two nodes in $\mathbf{G}$. Thus, an entry $c_{ij} = 1$ denotes that a comparison between node i and node j is performed, where each node depicts a distinct configuration:

$$\mathbf{C} = \begin{bmatrix} 0 & c_{12} & \cdots & c_{1N} \\ c_{21} & 0 & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & \cdots & 0 \end{bmatrix}. \tag{1}$$

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:

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

Entries where $\psi_{ij} = 1$ indicate that the output from nodes i and j is discrepant for the same input, and $\psi_{ij} = 0$ indicates their agreement. Meanwhile $\psi_{ij} = x$ 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. It is worthwhile to highlight that we consider a pair to be healthy if and only if no discrepancy had occurred during the lifetime of comparison evaluation. In other words, a value $\psi_{ij} = 1$ renders all future comparisons between node i and node j to retain its constant value 1.

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

Approach	Emphasis	Challenges
Baseline arrangement: TMR	Fault masking	Area, power, and resiliency
Algorithm 1: exhaustive evaluation	Quality of recovery	Diagnosis latency
Algorithm 2: SFH state transitions	Reduced diagnosis latency by employing evaluation history	Quality of recovery due to false negatives
Algorithm 3: DRFI	Availability by maintaining throughput during recovery	Diagnosis latency and recovery quality tradeoff

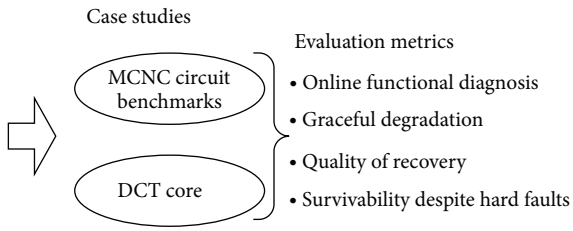


Figure 3: Online fault diagnosis strategies evaluated herein to availability, throughput degradation, and latency tradeoffs.

Require: N
Ensure: $\widehat{\Phi}$
(1) *Starting from the first configuration*
Initialize CUT = 1
(2) **while** CUT < N **do**
(3) *Choose the next configuration from the configurations pool as slack*
Designate the CUT as active configuration, and RS = CUT + 1 as slack.
Then, the pair under test is given by: V_{CED} = {CUT, RS}
(4) *Exhaustive evaluation method involves applying all the points in the input space of circuit*
Perform concurrent comparison to the same inputs for various edges of the graph represented by connectivity matrix **C**
(5) *Evaluate all the possible configurations pair to complete the syndrome matrix*
Update the Syndrome Matrix **Ψ** based upon comparisons outcome of all possible inputs
(6) CUT = CUT + 1
(7) **end while**
(8) *This method involves exploration of the entire input-space and thus exposes all the operationally manifested faults*
Given **Ψ**, isolate the faulty nodes:
$\widehat{\phi}_i \leftarrow 0$ and $\widehat{\phi}_j \leftarrow 0$, if $c_{ij} = 1$, and $\psi_{ij} = 0$
$\widehat{\phi}_i \leftarrow 1$ if $\widehat{\phi}_j = 0$, $c_{ij} = 1$, and $\psi_{ij} = 1$

Algorithm 1: Exhaustive evaluation of configurations for fault diagnosis.

condition for healthiness, with the estimated fitness vector being updated accordingly.

Condition. $\psi(i, j) = 0$ for any $1 \le i \le N$ and $1 \le j \le N$, where $i \ne j$ and $c_{ij} = 1$.

Update. $\widehat{\phi}_i = 0$.

4.2.1. Exhaustive Evaluation. To address the identification of faults without loss of generality, it is assumed that no correctness information is known priori for the functionality of the bitstreams in the configuration pool. In particular, there are no configurations that are known to be fault free or resilient to faults. For purposes of operation, the DRFI implementation and the device reconfiguration mechanism are assumed to be reliable which can be encouraged by minimizing device area relative to that of the throughput fabric. Such components have been referred to in the literature as *golden elements* [16]. Golden elements are subcircuits which are assumed to remain fault free throughout the duration of the mission. As there are no golden configurations needed to examine the fitness of other *Suspect* configurations, one straightforward approach is to exhaustively evaluate all the configuration pairs in a mutual checking paradigm as given by Algorithm 1. Given a configuration pair instantiated in a CED manner, a complete agreement in their output throughout all the possible inputs would affirm their healthy status. In practice, only nonexhaustive input evaluation is feasible which motivates the proposed DRFI technique described later.

Assume that an I-input circuit is instantiated in duplex manner. The cardinality of the input set is 2^I and all distinct combinations of input samples need to be applied to evaluate the behavior of the circuit to the entire input space. If the number of defective configurations is not known priori, an upper bound on diagnosis time in terms of number of input evaluations can be derived as given in the following: the number of ways in which k objects can be chosen out of a set of N objects is given by Binomial Coefficient [30]. To realize

Require: N
Ensure: $\widehat{\Phi}$
(1) Initialize CUT = 1
(2) **while** ($V_h == \phi$) and (CUT < N) **do**
(3) *The configurations under test in SFH method are selected in a round-robin manner*
Designate the CUT as active configuration, and RS as slack. Then, the pair under test is given by: V_{CED} = {CUT, RS}
(4) Perform concurrent comparison to the same inputs for various edges of the graph represented by connectivity matrix **C**
(5) *The comparison outcome in SFH method is binary. A 1 corresponds to discrepant behavior while a 0 corresponds to healthy configurations*
Update the Syndrome Matrix **Ψ** based upon comparisons outcome during an evaluation interval
(6) Increment the reload number $n_r = n_r + 1$
(7) Given **Ψ**, isolate the faulty nodes:
$\widehat{\phi}_i \leftarrow 0$ and $\widehat{\phi}_j \leftarrow 0$, if $c_{ij} = 1$, and $\psi_{ij} = 0$
$\widehat{\phi}_i \leftarrow 1$ if $\widehat{\phi}_j = 0$, $c_{ij} = 1$, and $\psi_{ij} = 1$
(8) **end while**
(9) *A healthy configuration can check the health of other configuration when instantiated concurrently for discrepancy check.*
Use a healthy identified configuration to test all other configurations

Algorithm 2: SFH state transition algorithm for functional diagnosis of configurations.

a CED pair, two configurations are selected out of the pool. Thus,

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

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

Figure 4 shows the upper bound on number of reconfigurations required to identify faulty configurations by duplex evaluations for various configuration pool sizes.

4.2.2. The SFH Fitness States Transitions Diagram Method. As a competing approach to the DRFI technique, we consider a state transition diagram method based upon the *Suspect*, *Faulty*, and *Healthy (SFH)* fitness transitions of the configuration bitstreams. An individual configuration undergoes different transitions in its fitness state throughout the life of a circuit. After fault detection, the fitness state of every configuration is *Suspect*. If two configurations show complete agreement in a given *Evaluation Period*, E, both are declared as *Presumed Healthy*. However, if a *Suspect* configuration exhibits discrepancy with a *healthy* one, it is marked as *Faulty*. The state transition diagram is illustrated in Figure 5. The objective of the state transitions flow is to identify *healthy* configurations in a pool of *Suspect* configurations which, in turn, helps to identify *faulty* items. The problem is similar to the counterfeit coin identification problem [31] with a restriction that only two coins can be tested at a time.

The SFH method is evaluated using Monte-Carlo simulation of the configurations' behavior. Let u_i represent configuration labels for all $1 \le i \le N$ for a configurations pool V of size N. The number of healthy configurations is $N_h = N - N_f$, and we identify them using discrepancy information. For this purpose, a configuration pair {CUT, RS} is randomly chosen to be instantiated on the chip, where {CUT, RS} $\in V$. The CUT and RS correspond to active and slack configurations of a CED pair, respectively. Once a discrepancy is detected, the fitness state of all of the configurations is suspected; that is,

$$u_i \cdot \text{FS} = \text{Suspect}; \quad \forall 1 \le i \le N. \tag{3}$$

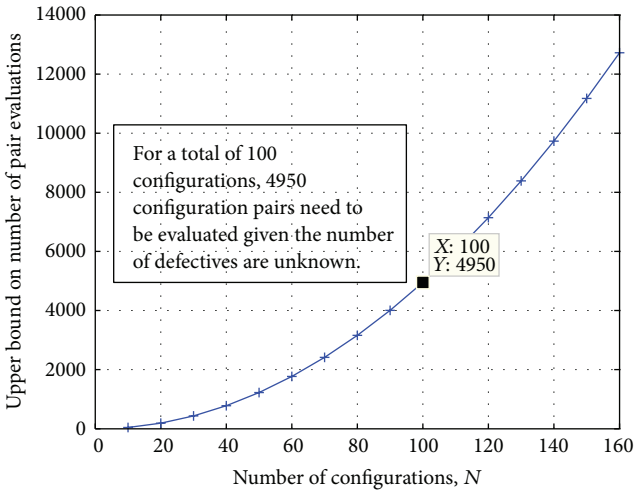
As the knowledge about fitness of those configurations is not available initially, the estimated number of *healthy* configurations is $\widehat{N}_h = 0$. Afterwards, another pair of configurations is randomly selected for instantiation while incrementing a variable *Reload Number*, n_r, as listed in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

$$v_{\text{CUT,RS}} \cdot \text{FS} = \text{Healthy}, \qquad \widehat{N}_h = \widehat{N}_h + 2. \tag{4}$$

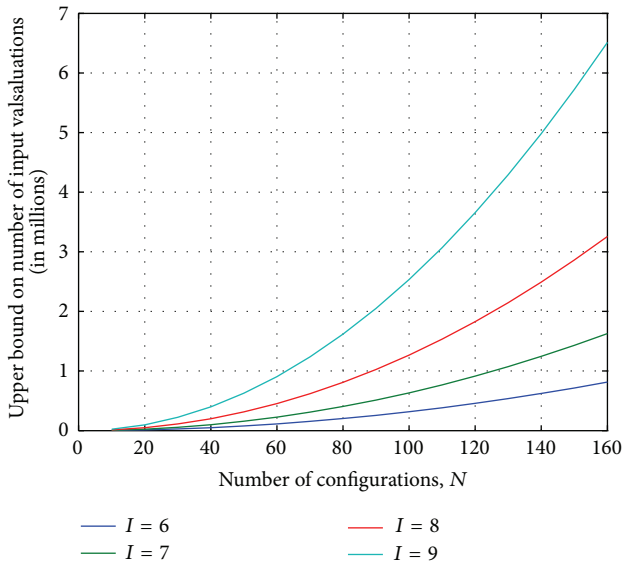
To reduce the number of configurations reloads, as it costs reconfiguration latency which, in turn, would affect the throughput, a *without-replacement* policy is also evaluated. In this strategy, an identified healthy pair is never reinstantiated during the diagnosis process under SFH approach.

Figure 6 shows history of knowledge about various configurations while they are instantiated for evaluation. The y-axis of the plot shows percentage of total number of *Presumed Healthy* configurations correctly identified using the discrepancy information, that is, $\widehat{N}_h/(N - N_f)$. We can see from the plot that the required number of configuration reloads increases with increased number of defective configurations, N_f. Moreover, the *without-replacement* policy provides better results than the *with-replacement* policy given no additional faults occur upon initiation of the fault diagnosis phase. In the following, we provide some probability analysis of the problem. The diagnosis problem of configuration bitstreams can be formulated as given below.

Given a collection containing a mix of defective and nondefective items, what is the probability that two items



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



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

FIGURE 4: Evaluation Cost of Fault diagnosis by exhaustive evaluation of configuration pairs.

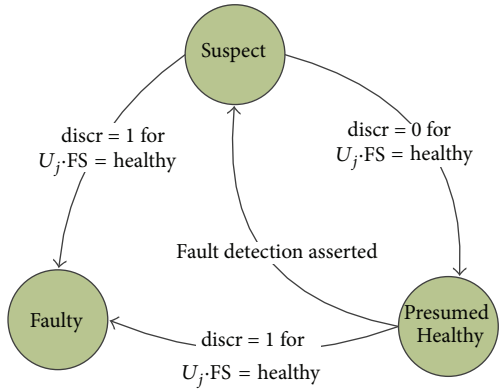
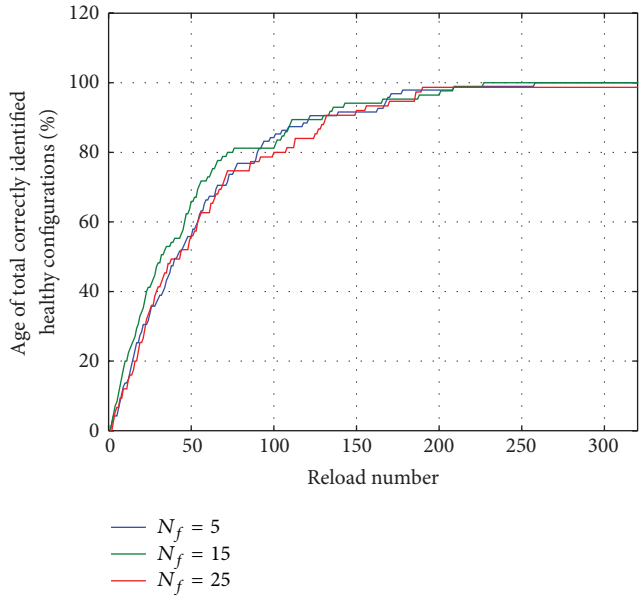
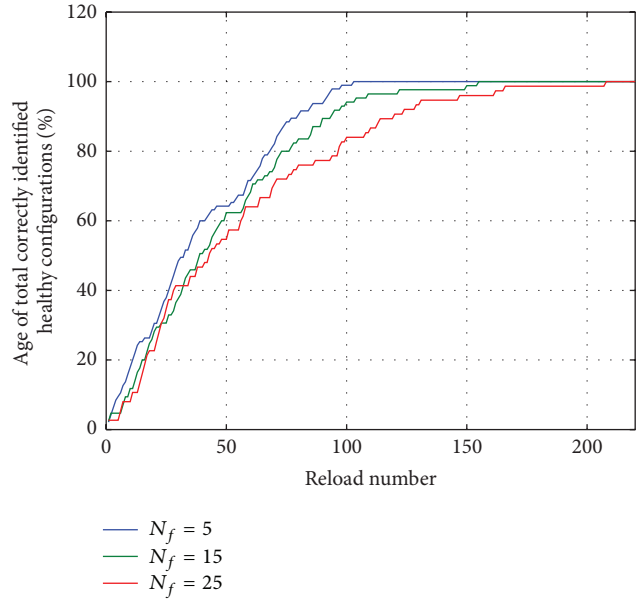


FIGURE 5: Fitness states of a design configuration during a circuit's lifetime.



(a) With replacement



(b) Without replacement

FIGURE 6: Identifying healthy configurations in a *Suspect* pool.

selected at random are nondefective [30]? To analyze this problem, first suppose $p(h)$ is the probability of selecting a single nondefective (healthy) item and $p(f)$ is the probability of selecting a defective (faulty) item. Using the notation introduced in the previous section, $p(h) = N_h/N$ and $p(f) = N_f/N$. Thus, the probability of selecting a nondefective pair is given by $p(hh) = p(h) * p(h \mid h)$ where $p(h \mid h)$ is the probability that the second item is nondefective given the first item was nondefective. The experiment of instantiating and evaluating a configuration pair is a Bernoulli trial whose outcome is either *success* when a healthy pair is selected or a *failure* when at least one of the configurations is faulty. The probability of k successes in the outcome of n Bernoulli trials

with replacement strategy is given by binomial probability law [30]:

$$p(k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad (5)$$

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

As each trial consists of picking a pair of items instead of a single item, the *probability mass function (pmf)* [30] becomes

$$\text{pmf}(k) = \binom{n}{k} p(hh)^k (1 - p(hh))^{n-k}. \quad (6)$$

The *cumulative distribution function (cdf)* of a random variable X provides the probability that the event will be found with value less than or equal to k [30]; that is,

$$\text{cdf}(k) = P[X \leq k]. \quad (7)$$

The probability of finding nondefective items in a batch of 100 items with various numbers of trials is shown in Figure 7. Out of one hundred items, 5 items are assumed to be defective. The pmf and cdf depend on p, k, and n. For example, the pmf for $n = 100$ trials shows that the probability of choosing exact $k = 91$ nondefective pairs is only 0.1331. In addition, the cdf plot shows that probability of success of selecting $k \leq 91$ healthy pairs in $n = 100$ trials is 0.6549.

To relate the probability analysis with the results from the Monte-Carlo simulation in Figure 6, assume that we are interested in finding the probability of success greater than k given n trials. This measure relates to probability that each healthy configuration is selected at least once paired with a healthy other configuration, in a certain number of loadings of configuration bitstreams of pairs. The cdf in Figure 7 shows that probability of successes $k \leq 95$ is approximately 0.1 given 110 trials, implying that probability of successes, $k > 95$, is 0.9; that is, $p(k > 95) = (1.0 - 0.1)$. Thus, we can expect that 90% of the trials would be successful in terms of selecting $k > 95$ nondefective pairs in $n = 110$ trials. It is evident from Figure 6 that roughly 90% of nondefective items are isolated in 110 iterations under SFH transitions method of input evaluations of various pairs.

It is essential to note our assumption here that if two configurations are loaded for a given evaluation period, they will exhibit discrepancy at least once if at least one of them is faulty. This assumption may not be true in many cases as we discuss in the next section. This is acceptable since SFH is just providing a baseline for comparison to the proposed DRFI approach.

5. The DRFI Approach

The DRFI technique of fault diagnosis using a functional testing paradigm fully exploits the dynamic reconfiguration capability of contemporary FPGAs. This technique utilizes the information about difference in output values of the duplex arrangement, in addition to discrepancy information. The diagnosis process begins with constructing a Circuit Similarity Graph and then applying the PageRank algorithm to compute the rank score of each node in the graph. The top

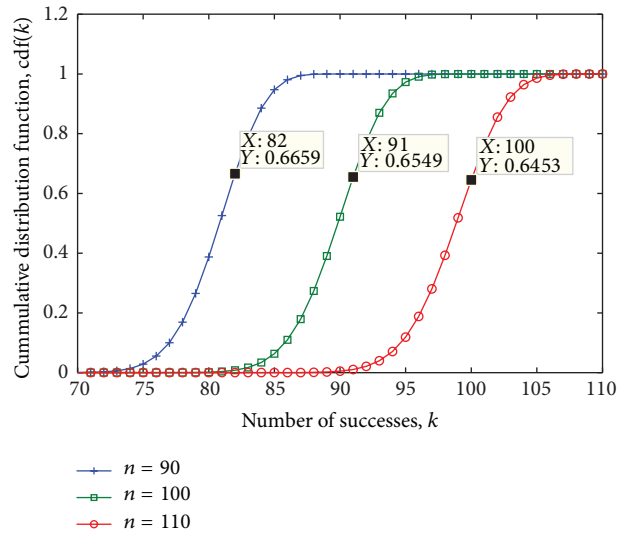


Figure 7: Probability of success for various trials with replacement.

μ configurations, having a score greater than the average score of a pool, are assumed to be fault free and hence can be used by the system. However, if no healthy configuration exists, then the pool is sorted in ascending order according to the scores and higher score configurations are preferred.

An online method to prioritize alternative configurations for instantiation is needed. The PageRank algorithm, which originated to rank the pages in the World Wide Web, has also been effective on non-web-based applications and has been found to be fast, scalable, and robust [32]. In the following, some background of the PageRank algorithm is provided to give some intuition of the analogy to the ranking of bitstreams.

Page et al. [33] developed the PageRank algorithm to rank the webpages on World Wide Web according to their importance. This algorithm is successfully being employed by the Google search engine. The basic motivation is that the webpages which are more important should be given higher PageRank value. The importance of a webpage is based on a factor determined by the number of references made to it by other pages, and hence it is determined recursively [33].

We make an analogy with the problem at hand, that is, bringing out some fault-free configurations from a pool of configurations. The configurations which use faulty resources lack consistent behavior with other configurations. However, the hardware realizations which utilize pristine resources would exhibit consistent behavior, when evaluated in duplex manner with the other realizations. The configurations showing consistent behavior are marked as important and mined from the pool using the PageRank algorithm. Table 1 identifies the analogies among various applications in which PageRank algorithm is being deployed. In Web Search applications, there are multiple content terms used on a webpage that determine its search relevance. Similarly, there are multiple resources in the FPGA fabric that determine its testing relevance.

Therefore, in this problem, the PageRank algorithm is utilized to analyze the relationship among different hardware

Table 1: Analogy of FPGA testing characteristics with ranking.

Context	Ranked element	Ranking metric	Dynamic coverage
Web search	Webpage content	Page views	Yes
Bioinformatics	Metabolic and protein interaction network databases	Biochemical reactions	No
FPGA failure testing	*FPGA configuration*	*Output discrepancy distance*	*Yes*

realizations. It assigns a score to each configuration depending on its relative significance in the pool of designs. The higher the score of a configuration is, the more consistent its behavior will be in the population of designs. After a fault is detected, a Circuit Similarity Graph (CSG) of configurations is constructed. In the CSG, an edge between two nodes represents the similarity between two circuits in terms of their output. If a realized circuit's output is consistently matched with the other circuits, it is considered more important than others and a higher rank is assigned to it. The fault handling flow is elaborated below.

Algorithm 3 lists various steps in the DRFI technique of functional diagnosis to rank hardware configurations in a reconfigurable, fault-resilient hardware platform. Fitness and throughput heuristics can be customized by considering the throughput quality during diagnosis phase and fault detection latency tradeoffs.

Building the Circuit Similarity Graph. The CSG is a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{W})$, where $\mathbf{V}$ is the vertex set, $\mathbf{E}$ is the set of edges, and $\mathbf{W}$ is the weight adjacency matrix associated with the graph. This is similar representation used for image features in a feature similarity graph of [34]. Each entry of $\mathbf{W}$ represents the degree of match between the corresponding circuits in terms of their output.

For constructing the weight adjacency matrix $\mathbf{W}$, each entry in the corresponding pair of the configurations forming a CED arrangement is evaluated during an evaluation period. While a binary assignment to a comparison-based diagnosis outcome is sufficient to relate the configurations under test to their relevant pool {*Healthy*, *Faulty*, and *Suspect*}, a quantification of their discrepant behavior further provides a relative ranking within each pool. Thus, a configuration whose output is relatively more incorrect in terms of its number of discrepant output bits becomes ranked low and is thus less preferred. Here, a Hamming distance measure can quantify number of discrepant bits in the pair-under-test evaluation. This distance measure is utilized to assign relative weights to the comparison outcome which quantify the partial functionality of configuration bitstreams. As discussed later, such a ranking scheme is beneficial in utilizing the partially functional circuits in signal processing applications where inexact behavior may be acceptable to some extent in terms of signal quality. However, the Euclidean distance between the outputs x_i and x_j represents the dissimilarity of the two circuits for online inputs. In general, the Euclidean distance d between two points $\mathbf{x}$ and $\mathbf{y}$ in n-dimensional space is defined as [35]

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \tag{8}$$

where n refers to the number of inputs in an evaluation interval. Then, the distance is normalized so that the measure becomes the matching score of the two circuits. For this purpose, Gaussian kernel is used to compute the matrix entries w_{ij} that represent the pair-wise similarity of corresponding indices [36]:

$$w_{ij}(x_i, x_j) = e^{-\|x_i - x_j\|^2 / 2\sigma^2}, \tag{9}$$

where σ^2 represents the variance of the Gaussian kernel [37]. Thus, the weight adjacency matrix can be computed using the output variation between two *Suspect* configurations operating a CED manner.

A pair having consistently matched outputs with each other for a whole range of inputs will get higher score as compared to the configurations differing in their outputs. In addition, the configurations completing agreeing during evaluation window are rewarded by subtracting a *Reward Score* from their associated *DV*. Consequently, a sparse matrix $\mathbf{W}$ is obtained for the configurations pool by randomly selecting different configuration pairs.

The size of the CSG seems a significant concern of the proposed method. It grows rapidly as the number of configurations increases. The size of CSG is directly impacted by the number of configurations created at design time and is determined by the extent to which fault tolerance is desired. A large number of configurations at design time imply requirement of large storage memory and increased fault-handling latency due to evaluation time. However, an improved fault coverage due to increased diversity of resource usage can be provided by a large number of configurations.

Ranking via PageRank. Given the CSG, we are interested in assigning score to each node where each node represents a particular circuit configuration. The idea is to give more score to the circuit whose output is consistently matched with the other circuits. Faults injected at random locations affect the different circuit configurations in different ways, and hence the circuits behave inconsistently to the inputs when evaluated in pairs with the other circuits. The CSG may be thought of as a graph similar to that of all linked webpages. The webpage which gets many votes or gets vote from high ranked pages receives higher rank. Therefore, to rank the pages according to their importance we apply the PageRank algorithm, which is demonstrated in Section 6. For web, the rank vector is computed for n webpages by observing the hyperlinks coming to and leaving from the webpages. For n circuit configurations, the rank vector P_r is $1 \times n$ vector where each value of P_r represents the PageRank score of the corresponding configuration.

Require: N
Ensure: $\widehat{\Phi}$
(1) **while** (1) **do**
(2) *DRFI is integrated fault detection, isolation, and recovery technique thereby keeping the system online during these phases.* Select CUT by using the fitness and throughput heuristic
(3) *Select two configurations from the configurations pool* Designate the CUT as active configuration, and RS as slack. Then, the pair under test is given by: $V_{\text{CED}} = \{\text{CUT, RS}\}$
(4) Perform concurrent comparison to the same inputs for various edges of the graph represented by connectivity matrix **C** given in (1)
(5) *DRFI employs error information between the outputs of two distinct reconfigurations as compared to* Algorithms 1 and 2 *which solely rely on a binary decision on the presumed fitness of configurations under test*
Compute the Syndrome matrix employing the distance information between two reconfigurations using (8)
(6) Update the weight adjacency matrix **W** based upon comparisons outcome during an evaluation interval using (9)
(7) Build the Circuit Similarity Graph from **W** and **C**
(8) Use PageRank Algorithm to rank the configuration pool according to the fitness assessment.
(9) **end while**

Algorithm 3: DRFI algorithm of ranking the functional configurations.

The PageRank of a page A is computed by [38]

$$\text{PR}(A) = (1-d) + d\left(\frac{\text{PR}(T_1)}{c(T_1)} + \cdots + \frac{\text{PR}(T_n)}{c(T_n)}\right), \quad (10)$$

where PR(A) = PageRank of a page A, $T_1 \cdots T_n$ = the pages which refer to page A, $c(A)$ = number of links going out of page A, and d = Damping Factor, empirically set to 0.85.

The PageRank is a probability distribution over all the linked webpages, and a random reference occurs to a webpage with a probability given by its PageRank value PR(A) [38].

Considering a random surfer model defined for the original PageRank algorithm [33], the PageRank can be conceptualized as a distribution based on a Markov chain. In a graph represented by the adjacency matrix, the surfer travels along the directed path with some transition probabilities. If at a given instant k the surfer is located at node i, then the node traversed at the next step $k+1$ can be any neighbor j of i. Thus, nodes can be considered to constitute n states of a Markov chain with a transition matrix P. An analysis of a random walk in a PageRank Markov chain is provided in [39]. It has been shown [40] that if the distribution of probabilities at a node i at instant k is given by $p^{(k)}$, then the probability to encounter node j at next step is given by

$$p^{(k+1)} = P^T p^{(k)}. \quad (11)$$

The PageRank vector is a stationary point of the above transformation as follows:

$$p = Ap; \qquad A = P^T. \quad (12)$$

The PageRank can be computed for a graph represented by an adjacency matrix by using various methods such as Arnoldi iteration, Gauss-Seidel iterations, power iterations, linear system formulations, and approximate formulations [41]. We used a linear system formulation of PageRank in this work.

An example of the configuration ranking process after evaluating multiple reconfigurations is shown in Figure 8. A pool of six design-time generated configurations are represented by the nodes in the graph. After multiple reconfigurations of the CED pair, the resultant connectivity matrix is given by

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \quad (13)$$

The Syndrome Matrix employing the comparison-output information after computing the distances via (8) is given below. The X values represent unknown distances between nodes which have not yet been compared by a discrepancy check:

$$\mathbf{\Psi} = \begin{bmatrix} X & X & -1.20 & X & X & X \\ X & X & -0.22 & X & X & X \\ -1.20 & -0.22 & X & -0.92 & X & X \\ X & X & -0.92 & X & -2.30 & -2.30 \\ X & X & X & -2.30 & X & X \\ X & X & X & -2.30 & X & X \end{bmatrix}. \quad (14)$$

The similarity of these configurations in terms of their output is given by the weight adjacency matrix, which represents the weights of edges between nodes. Figure 8(b) shows the weight adjacency matrix for $\sigma^2 = 0.5$. The PageRank value of the nodes is given below the node labels. As it is evident that configuration `label 3`'s similarity measure is higher among other configurations, therefore, it is ranked higher by the algorithm and thus preferred for fault recovery as described in the following section.

6. Fault Recovery Results

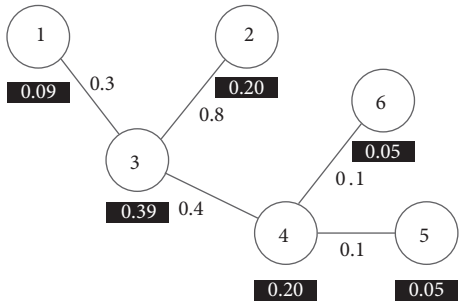
The fault model used in the experimental work to evaluate the proposed fault-handling scheme is a *Stuck-At (SA)* model

```
Before Fault Injection:

wire DONE_OBUF_4985;

  X_SFF #(
    .INIT (1'b0))
  DONE_3 (
    .CLK(CLK_BUFGP),
    .I(\count[6]_DONE_Select_14_o),
    .SRST(RST_START_OR_1_o),
    .O(DONE_OBUF_4985),
    .CE(VCC),
    .SET(GND),
    .RST(GND),
    .SSET(GND)
  );
  X_LUT2 #(
    .INIT (4'h8))
  Mmux_DOUT1111 (
    .ADR0(DONE_OBUF_4985),
    .ADR1(aes_dout[84]),
    .O(DOUT_84_OBUF_5035)
  );

After Fault Injection:

wire DONE_OBUF_4985;
wire DONE_OBUF_4985_tmp;
assign DONE_OBUF_4985_tmp=1; //Stuck-At '1' fault
  X_SFF #(
    .INIT (1'b0))
  DONE_3 (
    .CLK(CLK_BUFGP),
    .I(\count[6]_DONE_Select_14_o),
    .SRST(RST_START_OR_1_o),
    .O(DONE_OBUF_4985),
    .CE(VCC),
    .SET(GND),
    .RST(GND),
    .SSET(GND)
  );
  X_LUT2 #(
    .INIT (4'h8))
  Mmux_DOUT1111 (
    .ADR0(DONE_OBUF_4985_tmp),
    .ADR1(aes_dout[84]),
    .O(DOUT_84_OBUF_5035)
  );
```

Algorithm 4: An example of fault injection into the simulation model of the circuit.

in which such fault can occur at any of the LUT inputs used by a configuration. The SA model reasonably models the *permanent* effect of aging-degradation and radiation hazards on an FPGA device in a space environment. In addition, the DRFI technique deals with the faults at a higher level, that is, by functional evaluation of the overall circuit, and therefore, it should be capable of handling a wide variety of fault models. SA faults are injected in the simulation model of circuit generated by the Xilinx tool 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 Algorithm 4.

6.1. Experiment 1: MCNC Benchmark Circuits. To evaluate the DRFI technique, MCNC [42] benchmark circuit `misex` is analyzed in detail first, and then recovery of other circuits in the MCNC benchmark is assessed. The benchmark circuits are implemented targeting a Virtex-4 device. We used the MATLAB implementation of the PageRank algorithm by Gleich et al. [39, 41]. The ISim simulator output is an interface to a MATLAB script which issues commands to the ISE.



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

1.0	0.0	0.3	0.0	0.0	0.0
0.0	1.0	0.8	0.0	0.0	0.0
0.3	0.8	1.0	0.4	0.0	0.0
0.0	0.0	0.4	1.0	0.1	0.1
0.0	0.0	0.0	0.1	1.0	0.0
0.0	0.0	0.0	0.1	0.0	1.0

The similarity measure between C_6 and C_4

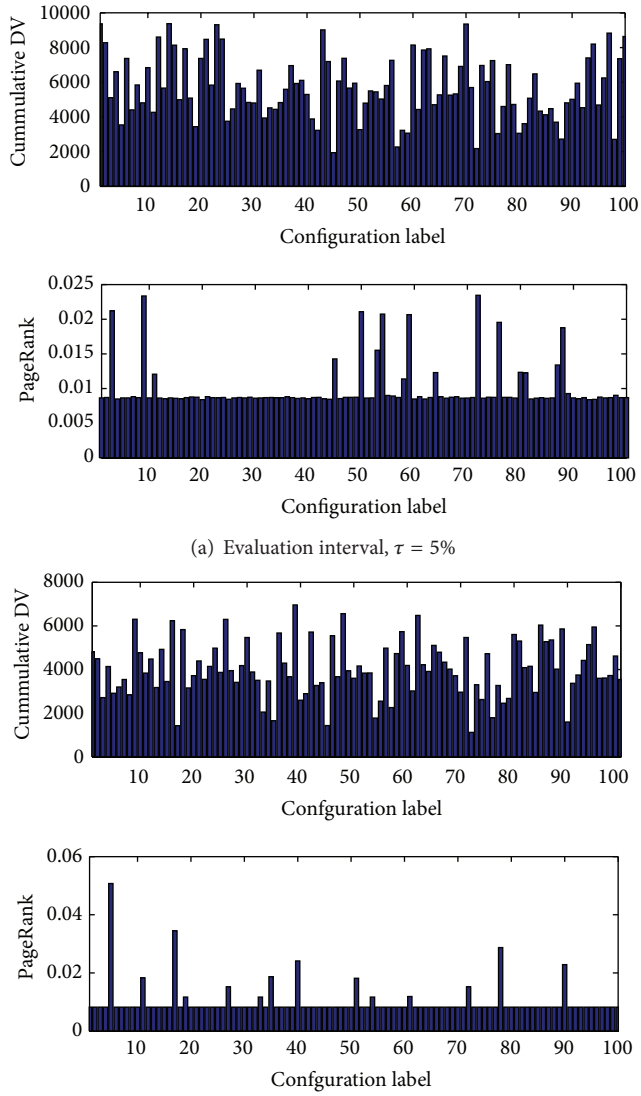
(b) Weight adjacency matrix, **W**

FIGURE 8: An example of configurations ranking.

The selection of the number of functional configurations is lower bounded by the amount of diversity required to mitigate faults while upper bounded by the tractability to handle the CSG. In this experiment, a total of 100 diverse configurations for the `misex` circuit are generated at design time. Then faults were randomly injected into the post-place-and-route simulation model affecting 86 circuit configurations and thereby leaving only 14 designs fully functional. The healthy configuration labels are listed here (`3,5,11,19,25,45,51,54,55,57,72,76,77,90`). The CSG is built by evaluating a pair of circuits to a subset of random inputs. The cardinality of the subset affects the fault-diagnosis quality. The evaluation interval τ is shown in Figure 9 as the percentage of the number of inputs applied to CUT from the overall input space. A sliding window of size 20 with an overlap of 10 is selected to evaluate the circuits in subpools. Instead of evaluating all exhaustive pairs with all exhaustive sets of inputs, the similarity matrix is built using a smaller set and thereby resulting in a sparse CSG. After computing the PageRank for the resulting graph, the results are shown in Figure 9 in which the PageRank value of each circuit implementation identified by its configuration label is plotted. The Cumulative Discrepancy Value (CDV) is defined as

$$\text{CDV} = \sum_{i=1}^{\tau} \text{DV}_i, \tag{15}$$

where τ denotes the evaluation interval as the number of inputs applied. CDV is used to build the CSG,



(a) Evaluation interval, τ = 5%

(b) Evaluation interval, τ = 10%

FIGURE 9: CDV and PR of various configurations (a) and (b).

and then PageRank is computed given the CSG of the reconfigurable design. As seen in Figure 9(a) for Cumulative DV, the CDV of various configurations cannot assist much in differentiating the healthy configurations from faulty configurations. However, the corresponding PageRank values clearly distinguish the healthy and faulty groups in the plot for PageRank shown in Figure 9(a) which has 14 peaks corresponding to each of the 14 unaffected configurations having configuration numbers, (`3,5,11,19,25,45,51,54,55,57,72,76,77,90`). In addition, Figure 9(a) shows that a sufficiently long evaluation interval should be chosen to confidently isolate healthy configurations in the configuration pool. Figure 9 shows that all the healthy configurations are identified without any false positives in the plot of PageRank results.

Analyzing the circuits with relatively high score in Figure 9, we observe that they utilize fault-free resources. It is, however, worth noting here that as few as two configurations are needed at any given time for the circuit to

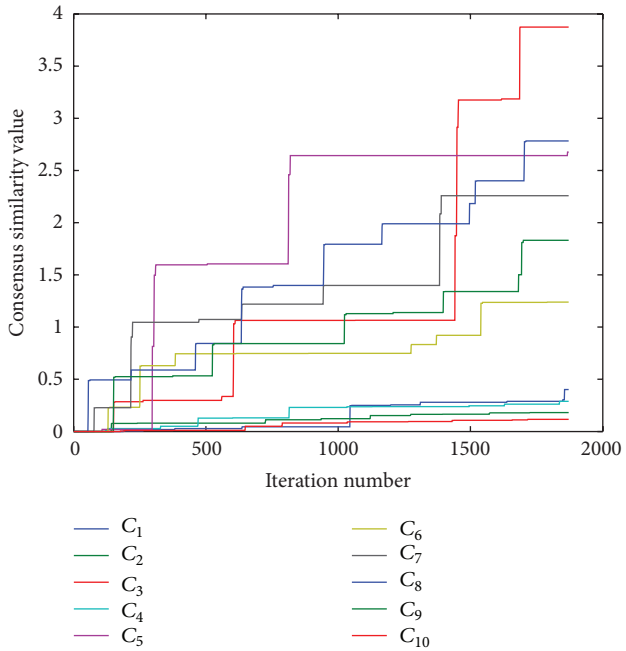


FIGURE 10: The discrepancy history of various configurations of the circuit.

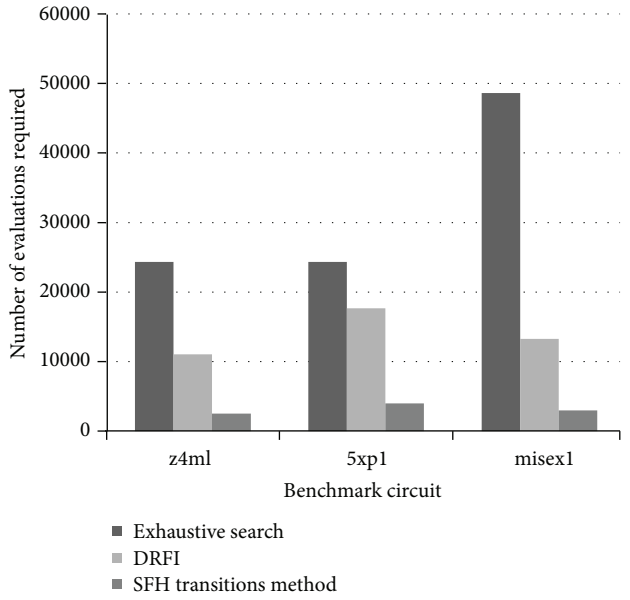


FIGURE 11: A comparison of fault-diagnosis methods for various MCNC benchmarks.

produce validated output, although we have identified much more successfully. The cumulative *Consensus Similarity Value* history of first 10 configurations is plotted in Figure 10. It is evident that the CSV of C_3 and C_5 increases with time as they both utilize fault-free resources.

As compared to the exhaustive testing method which involves evaluating all possible pairs with all possible set of inputs, DRFI achieved considerable improvement, as evident by Figure 11 which shows the results for other benchmarks. The observed reduction in input evaluations is up to 75% when using this approach for `misex1` benchmark circuit. For example, for the `misex1` benchmark results shown in Figure 11, the number of evaluations is reduced from 48640 in an exhaustive approach to 13260 when using the DRFI technique for a configuration pool size of 20 bitstreams.

The operation of the circuit in a duplex manner is simulated in Figure 12. The DV between outputs of the duplex circuit in each evaluation window is shown. During the *normal operation* period of the circuit, when no fault is present, the discrepancy value is zero. After one or more faults occur, the difference in output increases. During the *repair process*, different pairs of configurations are loaded and evaluated using random inputs. After a sparse similarity matrix is built, the PageRank algorithm is executed. Once the configurations are ranked and identified correctly, the normal operation of the circuit is recovered. It should be noted that Figure 12 is only for the illustration of the system's operation and not scaled by actual time, which depends on the time complexity of the controller and the CUT as identified later.

The method presented in this paper does not require explicit fault isolation phase, while the configurations are only generated at design time thereby not necessitating any synthesis or implementation tools at runtime. Fault handling is accomplished by promoting the hardware configurations which utilize fault-free resources [43]. DRFI is a system-level fault-diagnosis technique by which healthy configurations are identified in a configuration pool, while the instantiation of two healthy configurations in a duplex manner completes the fault-recovery process. The area overhead of DRFI technique over a baseline design is a replica of the original circuit plus the reconfiguration controller. Thus, it turns out to be rather comparable to CED as the reconfiguration controller hardware is already provided in Xilinx devices facilitating dynamic reconfiguration. The software for reconfiguration can be executed on the on-chip processor, soft core, or LUT realization.

6.2. Experiment 2: DCT Core. A DCT core was selected due to its popularity in deep space, earth satellites, unmanned vehicles, and other applications utilizing signal processing where human intervention may not be feasible. Here due to signal processing applications' inherent tolerance for noise and thus faults, it may not be necessary to triplicate modules in a TMR manner. Instead, we demonstrate how to exploit the quantifiable characteristics such as SNR and relative priority of the DCT coefficients to realize resilience, thereby reducing area resources, and energy while increasing sustainability for multiple faults.

The DRFI scheme is validated using the H.263 video encoder's 1-dimensional DCT block implemented on FPGA fabric using Xilinx ISE and PlanAhead for partial reconfiguration flow. There are 8 *Processing Elements (PEs)* computing the DCT coefficients [44, 45] of a row of pixels in 8×8 macroblock. Each PE's function is to compute one coefficient of the DCT function. For example, PE_0 computes the DC-coefficient, PE_1 computes the AC_0 coefficient, and so on. The 2D DCT is computed by using the 1-D DCT twice.

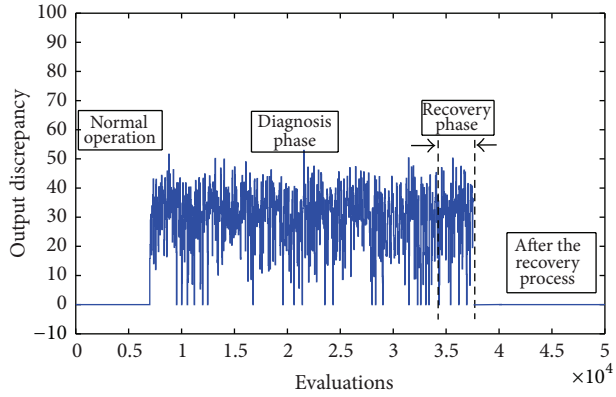


Figure 12: An operational example of a circuit on a $\times 10^4$ evaluations scale.

In a prototype to evaluate DRFI approach, we used the platform developed in [46] in which the video encoder application is run on the on-chip processor. All the subblocks 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 2D 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 100 MHz core operation can provide maximum throughput 100 M-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 760 K pixels per second.

Table 2 lists the resource utilization generated by the Xilinx tools for the DCT core interfaced with the on-chip PowerPC processor [47] which illustrates a significant reduction in reconfigurable resources when embedded multipliers are used.

Figure 13 shows qualitative results of the fault identification scheme. A frame in the video encoder's frame memory is shown in Figure 13(a). A total of 10 alternate configurations are generated at design time which utilize various PRRs in the chip. The DCT core is instantiated in CED mode to provide error detection capability. Figure 13(b) shows an intraframe after a fault injected in the PE_0 of the DCT core. The system is recovered by instantiating a pregenerated configuration which uses fault-free resources, as given in Figure 13(c).

In many signal processing applications, the underlying algorithms are inherently robust and a complete recovery is not necessary. While the latency of fault diagnosis may be excessive to achieve diminishing returns, sometimes it may barely improve the overall health metric of the system. On the other hand, a partial recovery can be quick and sufficient for certain applications. In a broad sense, provision of resilience in reconfigurable architectures for signal processing can take advantage of a shift from a conventional accurate computing model towards an approximate computing model [48–51]. This significance-driven model provides support for operational performance which is compatible with the concepts of signal quality and noise.

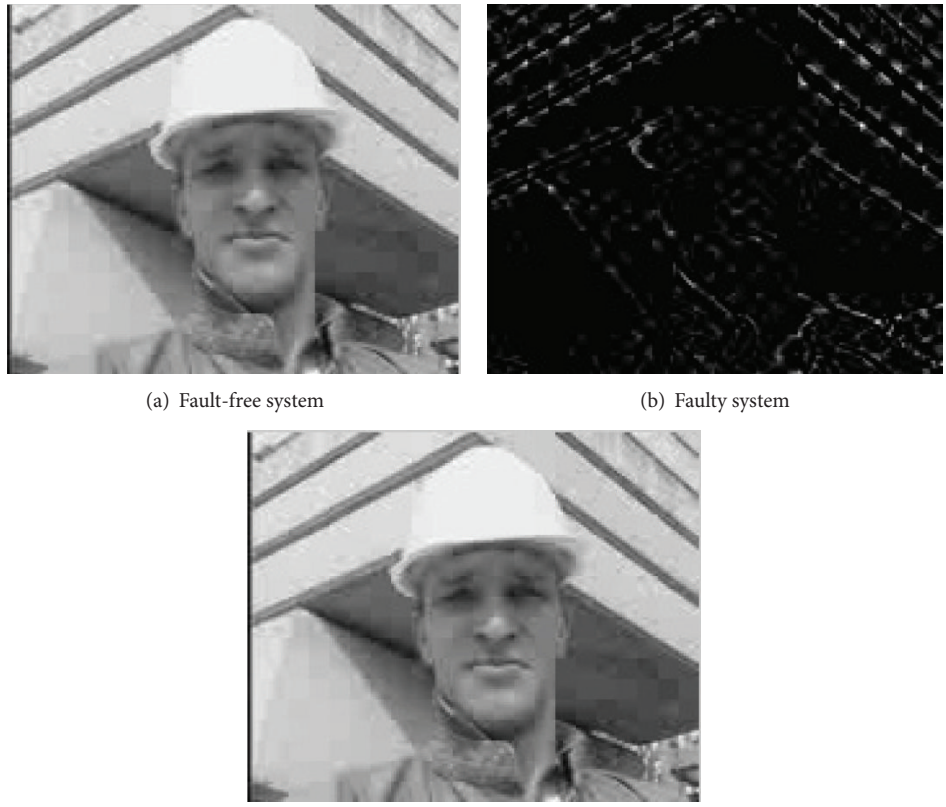
If all of the configurations become faulty, then in the current scheme a complete recovery is not possible. However, we are interested in at least those configurations whose behavior exhibits more correct outputs than the others for the relevant online input subspace. In cases where no individual configuration in the design pool is operational due to the faults affecting all the configurations, it is preferable to assign higher scores to those circuits which are relatively better. Figure 14 shows results of a simulation in which all the pregenerated configurations are affected by faults. The discrepancy check is made on the DC and AC_0-coefficients output values which contain most of the information about the image content. Figure 14(a) shows a case in which faults are injected in PE1. As can be seen, the image in Figure 14(c) is visually better than that in Figure 14(a), the former is an output of a configuration which utilizes a fault-free PE_1. Although the recovered system utilizes a faulty PE_3, a graceful degradation is made by the proposed recovery solution. Thus, the image quality in the frame buffer reflects the benefit of such partial recovery.

7. Comparisons and Tradeoffs

Figure 15 illustrates an operational comparison of two fault-handing techniques to mitigate local permanent damage, namely, TMR and DRFI, respectively. While the benefit of TMR is instantaneous fault-masking capability, its fault-handling capacity is limited to failure of only a single instance in the triplicated arrangement. On the other hand, DRFI's diagnosis and recovery phase involves multiple reconfigurations, and thus the fault-handling latency is significantly longer than that of TMR. On the positive side, DRFI can sustain multiple failures in the design. After fault detection, the diagnosis and repair mechanism are triggered which selects configurations from the pool and healthy resources are promoted to higher preference which results in improved

Table 2: Resource utilization summary of the DCT core.

Logic resource	Used by the static region	Used by a PRR	Used by a PRR employing multipliers
Number of slices	4516	169	62
Number of slice flip flops	5961	79	44
Number of 4 input LUTs	6155	312	100
Number of FIFO16/RAMB16s	45	0	0
Number of DCM_ADVs	1	0	0
Number of DSP48 blocks	0	0	1



(a) Fault-free system (b) Faulty system

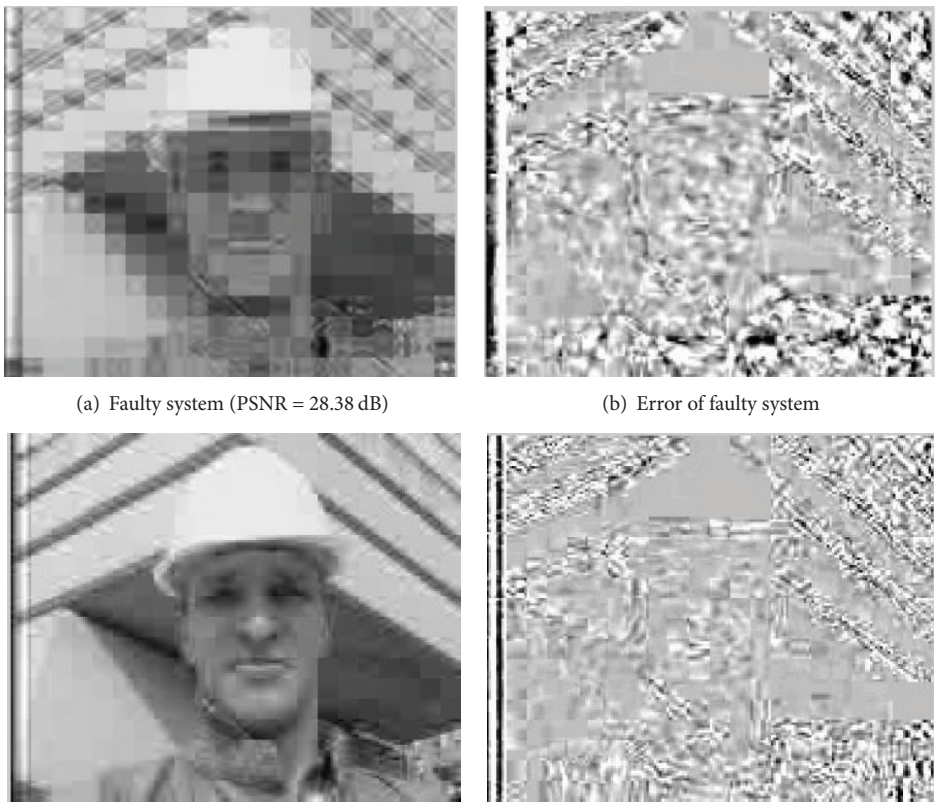
(c) Recovered system

Figure 13: An image in the frame memory of video encoder.

throughput quality. Our results show that the recovery latency is 1.2 sec for the video encoder case study. If the mission cannot tolerate recovery delay of this interval then TMR is preferable for the first fault; however, multiple faults impacting distinct TMR modules lead to an indeterminate result.

Figure 15 illustrates the operational differences of TMR and DRFI techniques to mitigate local permanent damage. Although DRFI incurs a recovery latency of approximately 1 second, it can sustain recovery after second, third, or even subsequent faults while TMR is only able to recover from a single fault per module over the entire mission. If the mission cannot tolerate a recovery delay of this duration, then TMR is preferable for the first fault and yet DRFI may be preferred for handling multiple faults and for provision of graceful degradation. However, an important insight is that TMR and DRFI need not be mutually exclusive. If device area is not at a premium, then each configuration could be realized in a TMR arrangement for low-latency initial fault recovery, along with DRFI being applied to a pool of TMR configurations at the next higher layer. Thus, DRFI need not necessarily be exclusive of TMR but orthogonal to it if sustainability and graceful degradation are also sought.

In order to assess the resource overhead of TMR and DRFI over a uniplex scheme, we implemented DCT core on Virtex-4 FPGAs in two different arrangements. The TMR arrangement involves a triplicated design while the DRFI arrangement involves duplicated design along with the configuration memory required to store multiple bitstreams in order to mitigate hardware faults. While a simplex arrangement incurs only a 33% of the area of TMR, the DRFI technique incurs approximately 67% of the area of TMR



(a) Faulty system (PSNR = 28.38 dB)

(b) Error of faulty system

(c) Recovered system (PSNR = 30.75 dB)

(d) Error of recovered system

Figure 14: Partial recovery results of the scheme.

when considering the logic resource count. In addition, DRFI contains provision of reconfiguration and thereby utilizes a reconfiguration controller and peripherals which are already resident on the chip.

For DRFI, configuration ranking is invoked to leverage the priority inherent in the computation to mitigate performance-impacting phenomena such as Extrinsic Fault Sources, Aging-induced Degradations, or manufacturing Process Variations. Such reconfigurations are only initiated periodically, for example, when adverse events such as discrepancies occur. Fault recovery is performed by fetching alternative partial Configuration Bitstreams 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 reconfigure the alternative bitstreams. 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 reconfiguration flows needed to realize a runtime adaptive approach to fault-handling architectures. However, it is worth mentioning here that the reconfiguration related components are not on the critical throughput datapath and can be triggered only when needed. That is, only after a fault is detected in the CED pair. Their reliability only impacts the recovery capability but not the correctness of the throughput datapath itself as would a Stuck-At fault in the voter output in a TMR design. Although based on equiprobable fault distribution dictated by relative area of the voter and module datapaths, it is a relatively remote possibility.

Table 3 lists the configuration bitstream sizes for various PEs in DCT core which can be used to assess the configuration memory size requirement. The listed factors are involved in the reconfiguration flow and hence add to the overhead of the diagnostic provision in DRFI approach.

In addition to the above mentioned components required by the DRFI approach, *bus-macros* and unoccupied resources also add to the overhead of the DRFI adaptive reconfiguration scheme. A distributed realization of the PRRs needed to create diverse configurations pool may result in consuming an increased count of the above resources. In order to minimize the bus-macros as well as simplifying the reconfiguration scheme, we have partitioned the chip into an exclusive set of resources, right and left halves. Thus, an MCNC benchmark circuit is fit into one half of the device while the other half contains its replica. This results in the mapping of many unoccupied resources into a defined PRR. However, given that the capacity of contemporary FPGAs keeps increasing, area overhead is not a significant concern especially when survivability and adaptability of the mission are primary concerns.

In a Virtex-4 device, the minimum PRR height that can be defined is 16 CLBs [26] while the maximum height can span an entire column in the chip. To effectively utilize

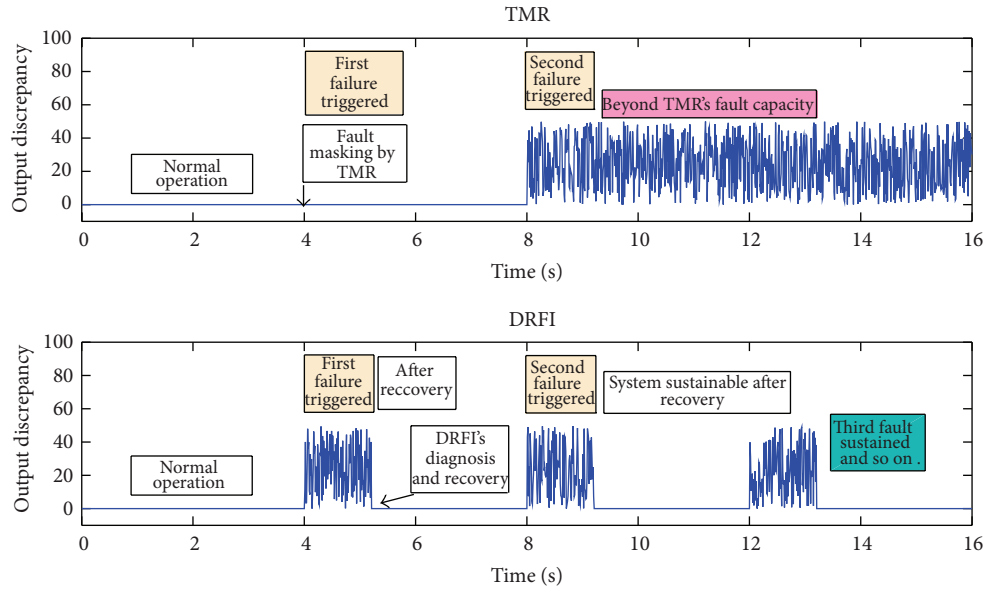


Figure 15: A comparison of DRFI with TMR in terms of system's capability to sustain multiple failures.

Table 3: Configuration bitstreams storage requirement for DCT core.

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

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. While the PE design we have considered in the DCT core consumes fewer resources than the capacity of a PRR, we choose the same as the minimum PRR size constrained by the vendor's tool and FPGA device under consideration. To reduce the memory size required in order to store configuration bistreams, a compression technique significantly reducing the storage space requirements of alternative partial bitstreams is presented in [52] and can be employed in future work.

While partial reconfiguration is not a requirement for the DRFI scheme and the scheme is applicable to static designs as well, Partial Reconfiguration helps apportion large designs into independent testing domains. This would facilitate throughput of nonaffected regions of a device while recovery occurs in the fault-affected areas. This can hide the latency of recovery in designs where the FPGA is performing decoupled independent operations. Examples would include a single FPGA device with a DCT core and an independent encryption core. If the encryption core is involved in transmission of information unrelated to the DCT core, its operation is unaffected during reconfiguration utilized for DRFI-based recovery. Moreover, even with a faulty design, partial functionality of the DCT core at runtime for a signal processing application during the fault-handling phase may provide graceful degradation and thus maintain some system functionality, although with a degraded quality of throughout. Demonstration of partial online functionality to help maintain signal quality to some extent during the fault-handling phase is demonstrated in [46] for non-distance-ranked approaches.

8. Conclusion

An approach for fault handling in FPGA-based systems is presented. In this method, a pool of hardware configurations for a reconfigurable platform is generated at design time utilizing a distinct set of hardware resources. Once faults affect circuit realizations, the PageRank algorithm is used to identify the most functional realizations. The experiments indicate that the approach is effective at identifying the correct configuration in a fraction of the comparisons needed by

unguided search and thereby offering considerably improved throughput. In addition, *graceful degradation* is realized by prioritizing the bitstreams in situations where all configurations are impacted by a fault. It may be noted that the method concentrates on local permanent damage rather than soft errors which can be effectively mitigated by *Scrubbing* [53–55]. Scrubbing can be effective for SEUs/soft errors in the configuration memory; however, it cannot accommodate local permanent damage due to Stuck-At faults, Electromigration, and Bias Temperature Instability (BTI) aging effects which require an alternate configuration to avoid the faulty resources. For future work, an interesting direction can be analyzing the effect of varying the granularity of diagnosis by using the PR model developed in [56]. Alternative ranking algorithms popular in web search in addition to PageRank, such as HITS [57] and SALSA [58], would explore interesting tradeoffs such as completeness of recovery and the recovery time.

Notation

G(**V**,**E**): An undirected graph, where **V** is the set of all nodes and **E** is the set of edges
C: Connectivity matrix
c_{ij}: An element of **C** corresponding to an output comparison of node i and node j
Ψ: Syndrome Matrix
Φ: Fitness State Vector
V_h: Set of healthy nodes
V_{CED}: Set for Concurrent Error Detection (CED) checking
N: Total number of nodes
N_f: Number of faulty nodes
N_h: Number of healthy nodes
n_r: Reload number or testing arrangement instance
τ: Evaluation interval.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] E. Stott and P. Y. K. Cheung, "Improving FPGA reliability with wear-levelling," in *Proceedings of the 21st International Conference on Field Programmable Logic and Applications (FPL '11)*, pp. 323–328, September 2011.

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

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

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

[5] J. D. Russell and C. R. Kime, "System fault diagnosis: closure and diagnosability with repair," *IEEE Transactions on Computers*, vol. C-24, no. 11, pp. 1078–1089, 1975.

[6] A. D. Friedman and L. Simoncini, "System-level fault diagnosis," *Computer*, vol. 13, no. 3, pp. 47–53, 1980.

[7] G. W. Greenwood, "On the practicality of using intrinsic reconfiguration for fault recovery," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, pp. 398–405, 2005.

[8] J. C. Laprie, "Dependable computing and fault tolerance : concepts and terminology," in *Proceedings of the 25th International Symposium on Fault-Tolerant Computing-Highlights from Twenty-Five Years*, June 1995.

[9] M. Malek, "A comparison connection assignment for diagnosis of multiprocessor systems," in *Proceedings of the 7th annual symposium on Computer Architecture (ISCA '80)*, pp. 31–36, ACM, New York, NY, USA, 1980.

[10] C. Carmichael, "Triple module redundancy design techniques for virtex FPGAs," Xilinx Application Note, Virtex Series XAPP197 (v1. 0. 1), 2006.

[11] F. Lima Kastensmidt, L. Sterpone, L. Carro, and M. Sonza Reorda, "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs," in *Proceedings of the Design, Automation and Test in Europe (DATE '05)*, pp. 1290–1295, March 2005.

[12] B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin, "Improving FPGA design robustness with partial TMR," in *Proceedings of the 44th Annual IEEE International Reliability Physics Symposium (IRPS '06)*, pp. 226–232, March 2006.

[13] J. Sloan, D. Kesler, R. Kumar, and A. Rahimi, "A numerical optimization-based methodology for application robustification: transforming applications for error tolerance," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '10)*, pp. 161–170, July 2010.

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

[15] E. Stott, P. Sedcole, and P. Y. K. Cheung, "Fault tolerant methods for reliability in FPGAs," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '08)*, pp. 415–420, September 2008.

[16] M. Garvie and A. Thompson, "Scrubbing away transients and Jiggling around the permanent: Long survival of FPGA systems through evolutionary self-repair," in *Proceedings of the 10th IEEE International On-Line Testing Symposium (IOLTS '04)*, pp. 155–160, July 2004.

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

[18] R. F. Demara, K. Zhang, and C. A. Sharma, "Autonomic fault-handling and refurbishment using throughput-driven assessment," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1588–1599, 2011.

[19] J. M. Emmert, C. E. Stroud, and M. Abramovici, "Online fault tolerance for FPGA logic blocks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 2, pp. 216–226, 2007.

[20] S. Dutt, V. Verma, and V. Suthar, "Built-in-self-test of FPGAs with provable diagnosabilities and high diagnostic coverage with application to online testing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 2, pp. 309–326, 2008.

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

[22] M. G. Gericota, G. R. Alves, M. L. Silva, and J. M. Ferreira, "Reliability and availability in reconfigurable computing: a basis for a common solution," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 11, pp. 1545–1558, 2008.

[23] S. Mitra, W.-J. Huang, N. R. Saxena, S.-Y. Yu, and E. J. McCluskey, "Reconfigurable architecture for autonomous self-repair," *IEEE Design and Test of Computers*, vol. 21, no. 3, pp. 228–240, 2004.

[24] Xilinx, *Planahead 10. 1 User Guide*, 2008.

[25] Xilinx, *Virtex-4 FPGA Configuration User Guide (Ug071)*, 2009, http://www.xilinx.com/support/documentation/user_guides/ug071.pdf.

[26] Xilinx, *Partial Reconfiguration User Guide*, UG702 (v14. 3), 2012.

[27] Xilinx, *Planahead User Guide*, UG632 (v14. 3), 2012.

[28] E. P. Kim and N. R. Shanbhag, "Soft N-modular redundancy," *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 323–336, 2012.

[29] I. Koren and S. Y. H. Su, "Reliability analysis of n-modular redundancy systems with intermittent and permanent faults," *IEEE Transactions on Computers*, vol. C-28, no. 7, pp. 514–520, 1979.

[30] A. Leon-Garcia, *Probability, Statistics, and Random Processes for Electrical Engineering*, Pearson/Prentice Hall, 2008.

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

[32] G. Iván and V. Grolmusz, "When the web meets the cell: using personalized PageRank for analyzing protein interaction networks," *Bioinformatics*, vol. 27, no. 3, Article ID btq680, pp. 405–407, 2011.

[33] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: bringing order to the web," Technical Report 1999-66, Stanford InfoLab, 1999.

[34] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos in the Wild," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops '09)*, pp. 1996–2003, June 2009.

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

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

[37] U. von Luxburg, "A tutorial on spectral clustering," Technical Report TR-149, Max Planck Institute for Biological Cybernetics, 2007.

[38] S. Brin, "The anatomy of a large-scale hypertextual Web search engine 1," *Computer Networks*, vol. 30, no. 1–7, pp. 107–117, 1998.

[39] D. Gleich, P. Glynn, G. H. Golub, and C. Greif, "Three results on the PageRank vector: eigenstructure, sensitivity and the derivative," in *Proceedings of the Dagstuhl conference in Web retrieval and numerical linear algebra algorithms*, A. Frommer, M. Mahoney, and D. Szyld, Eds., Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Dagstuhl, Germany, 2007 2007.

[40] P. Berkhin, "A survey on PageRank computing," *Internet Mathematics*, vol. 2, no. 1, pp. 73–120, 2005.

[41] D. Gleich, "pagerank at mathworks.com," 2006, https://www.cs.purdue.edu/homes/dgleich/.

[42] S. Yang, "Logic synthesis and optimization benchmarks version 3," Tech. Rep., Microelectronics Center of North Carolina, 1991.

[43] N. Imran and R. F. DeMara, "A fault-handling methodology by promoting hardware configurations via pagerank," in *Proceedings of the ReSpace/MAPLD Conference*, Albuquerque, New Mexico, 2011.

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

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

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

[47] Xilinx, *Embedded System Tools Reference Manual*, 2008, http://www.xilinx.com/support/documentation/sw_manuals/edk10_est_rm.pdf.

[48] K. V. Palem, L. N. B. Chakrapani, Z. M. Kedem, A. Lingamneni, and K. K. Muntimadugu, "Sustaining Moore's law in embedded computing through probabilistic and approximate design: retrospects and prospects," in *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '09)*, pp. 1–10, ACM, New York, NY, USA, October 2009.

[49] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar, "Scalable effort hardware design: exploiting algorithmic resilience for energy efficiency," in *Proceedings of the 47th Design Automation Conference (DAC '10)*, pp. 555–560, June 2010.

[50] D. Mohapatra, G. Karakonstantis, and K. Roy, "Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '09)*, pp. 195–200, ACM, New York, NY, USA, August 2009.

[51] G. Karakonstantis, N. Banerjee, and K. Roy, "Process-variation resilient and voltage-scalable dct architecture for robust low-power computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 10, pp. 1461–1470, 2010.

[52] A. Vavousis, A. Apostolakis, and M. Psarakis, "A fault tolerant approach for FPGA embedded processors based on runtime partial reconfiguration," *Journal of Electronic Testing*, vol. 29, no. 6, pp. 805–823, 2013.

[53] P. S. Ostler, M. P. Caffrey, D. S. Gibelyou et al., "SRAM FPGA reliability analysis for harsh radiation environments," *IEEE Transactions on Nuclear Science*, vol. 56, no. 6, pp. 3519–3526, 2009.

[54] N. Imran, R. Ashraf, and R. F. DeMara, "On-demand fault scrubbing using adaptive modular redundancy," in *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA '13)*, pp. 22–25, Las Vegas, Nev, USA, July 2013.

[55] J. Heiner, N. Collins, and M. Wirthlin, "Fault tolerant ICAP controller for high-reliable internal scrubbing," in *Proceedings of the IEEE Aerospace Conference (AC '28)*, March 2008.

[56] K. Papadimitriou, A. Dollas, and S. Hauck, "Performance of partial reconfiguration in FPGA systems: a survey and a cost

model," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 4, pp. 36:1–36:24, 2011.

[57] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[58] R. Lempel and S. Moran, "SALSA: the stochastic approach for link-structure analysis," *ACM Transactions on Information Systems*, vol. 19, no. 2, pp. 131–160, 2002.