Fast Online Diagnosis and Recovery of Reconfigurable Logic Fabrics Using Design Disjunction

Ahmad Alzahrani, *Member, IEEE,* and Ronald F. DeMara, *Senior Member, IEEE*

Abstract—Design disjunction is developed to offer a broad coverage, high resolution, and low overhead approach to online diagnosis and recovery of reconfigurable fabrics. Design disjunction leverages the condensed diagnosability of T logic resources to achieve self-recovery using partial reconfiguration in O(log T) steps. Reconfiguration is guided by the constructive property of f-disjunctness which forms O(log T) resource groups at design-time. Resolution of f simultaneous resource faults is shown to be guaranteed when the resource groups are mutually f-disjunct. This extends run-time fault resilience to a large resource space with certainty for up to f faults using a decision-free resolution process that also provides a high likelihood of identifying the fault's location to a fine granularity. Finally, design disjunction is parameterized to accommodate the low coverage issue of functional testing for which inarticulate tests can otherwise impair fault isolation. Experimental results for MCNC and ISCAS benchmarks on a Xilinx 7-series field programmable gate array (FPGA) demonstrate f-diagnosability at the individual slice level with a minimum average isolation accuracy of 96.4 percent (94.4 percent) for $f = 1$ ($f = 2$). Results have also demonstrated millisecond order recovery with a minimum increase of 83.6 percent in fault coverage compared to N-modular redundancy (NMR) schemes. Recovery is achieved while incurring an average critical path delay impact of only 1.49 percent and energy cost roughly comparable to conventional two-MR approaches.

Index Terms—Reconfigurable logic devices, field programmable gate arrays, autonomous fault handling, fault-tolerant systems, run-time fault diagnosis and recovery, online test, design space exploration

1 Introduction

Continued scaling of transistor feature size has exacerbated reliability concerns such as process variation, aging degradations, latent faults, and temporary failures in integrated circuits (ICs). Consequently, the need for IC fault tolerance has received increasing interest over the last decade. Moreover, the pervasive use of embedded computing systems realized by field-programmable gate arrays (FPGAs) has elevated the importance of FPGA availability requirements corresponding to the proportion of time that their operation can be sustained. A common requirement is to provide high availability (HA) operation defined by 99.999 percent ("five nines") that correlates to five minutes of downtime per year, or greater availability such as 99.999999 percent ("eight nines") that correlates to 316 milliseconds of downtime per year [1]. High availability operation is crucial whenever unavailability could result in potential harm or inconvenience, violation of a service-level agreement, or a loss of revenue, mission, and/or safety.

Traditionally, availability requirements can be achieved through spatial resource redundancy to mask or replace faulty elements. Availability depends on rapid fault recovery to incur minimal downtime via autonomous fault resolution. As opposed to FPGAs, application-specific integrated circuits (ASICs) use fixed redundancy configurations which preclude fine-grained resource remapping. Whereas FPGAs can enable dynamic fine-grained resiliency, a novel online technique is developed using rapid self-organization to attain HA objectives.

Reconfigurable hardware's capacity to self-organize can fulfill anticipated roles in designing future dependable hardware systems [2]. At present, the most widely adopted reconfigurable architectures are SRAM-based FPGAs whose capacity can exceed a million logic cells which can be leveraged to enable resilience. SRAM-based FPGAs are ubiquitous in application-specific embedded systems, high performance computing centers as well as safety-impacting, mission-critical, and commerce-enabling systems. The FPGA devices within these systems can significantly impact the overall system reliability [3]. Fortunately, run-time partial reconfiguration capabilities of contemporary FPGAs can be utilized to maintain degraded-mode operation while enabling rapid recovery from a variety of faults.

Over the last two decades, a significant body of research has focused on realizing FPGA-based systems that are robust to permanent and transient failures. Permanent failures constitute any irreversible damage to the physical resources, whereas transient failures are short-duration events induced by external sources such as charged particles [4]. Particle-induced transient faults, or soft errors, cause single event upsets (SEUs) which can alter SRAM configuration bits and lead to a functional failure. Conventional

• *A. Alzahrani is with the Department of Computer Engineering, Umm Al-Qura University, Makkah 21955, Saudi Arabia. E-mail: azahrani@uqu.edu.sa.*
• *R.F. DeMara is with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816. E-mail: ronald.demara@ucf.edu.*

Manuscript received 2 Mar. 2015; revised 20 Nov. 2015; accepted 24 Nov. 2015. Date of publication 31 Dec. 2015; date of current version 14 Sept. 2016. Recommended for acceptance by P. Eles.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2015.2513762

0018-9340 © 2015 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

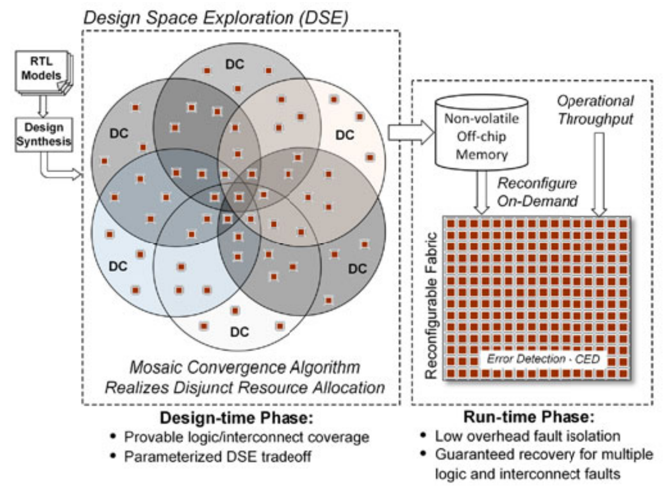

Design Space Exploration (DSE)
RTL Models
Design Synthesis
DC
DC
DC
DC
DC
DC
Mosaic Convergence Algorithm Realizes Disjunct Resource Allocation
Non-volatile Off-chip Memory
Operational Throughput
Reconfigure On-Demand
Reconfigurable Fabric
Error Detection · CED
Design-time Phase:
• Provable logic/interconnect coverage
• Parameterized DSE tradeoff
Run-time Phase:
• Low overhead fault isolation
• Guaranteed recovery for multiple logic and interconnect faults

Fig. 1. Objectives of proposed design disjunction approach.

resilience techniques for soft-errors and single permanent faults are based on fault-masking via majority voting [5], [6]. Voting methods such as N-modular redundancy (NMR) incur N-fold power and area overheads to tolerate temporary and permanent faults in up to $\lfloor (N-1)/2 \rfloor$ modules. Techniques such as re-execution and reconfiguration scrubbing [7], [8] can provide low overhead recovery for temporary failures.

Alternatively, dynamic remapping of a single design implementation at the module or logic-tile level can be employed to deallocate the use of damaged resources [9]. However, the existing techniques for remapping of FPGA resources at runtime can significantly increase the time complexity of recovery, and thus the downtime. The recovery overhead includes run-time remapping entailing on-board execution of FPGA design processes, such as place and route (PAR), which are time consuming. A single implementation of an FPGA-based design can require minutes to hours using a high-end multicore processor [10]. Although execution time for remapping can be substantially decreased using incremental PAR if locations of faulty elements are known, it is still a difficult computational workload for embedded processing cores [11]. Thus, conventional dynamic remapping techniques typically require faulty systems to be taken offline for an undesirable interval of time.

In this paper, a new deterministic design space exploration (DSE) [12], [13] method is used to realize FPGA fault tolerance that achieves the availability and reliability objectives shown in Fig. 1. The design space, and thus the fault-resolution space, need only be explored at design-time by creating a small library of alternative *design configurations* (DCs) with f-disjunct resource usage. DCs are created using the mosaic convergence algorithm developed such that at least one DC in the library evades any occurrence up to d resource faults, where d is lower-bounded by f. The f-disjunction of resources among alternative DCs enables run-time fault localization by a non-adaptive group testing (NGT) technique. This realizes a novel low overhead fault localization/fault isolation capability along with rapid fault recovery from temporary and permanent faults in reconfigurable fabrics while incurring minimal area, power, and perturbation to normal system throughput. We show that the combinatorial properties of f-disjunctness, along with FPGA dynamic partial reconfiguration, enable fault resilience against extensive fault scenarios by reusing a subset of the DCs to ensure continual execution with minimal recovery time.

Overall, the contributions of this work include:

- the first approach to utilize design disjunction for condensed diagnostic analysis of reconfigurable hardware,
- an explicit fine-grained approach to determine the optimal number of DCs at design-time using the property of f-disjunctness for recovery from multiple logic and interconnect failures during the system lifetime,
- an extension of NGT to overcome the low coverage of online functional testing, and
- improvement in crucial metrics including availability, provability of recovery, fault coverage, fault isolation accuracy, and area efficiency.

The remainder of this paper begins with a review of the related work in Section 2. Section 3 provides an introduction to group testing and the property of f-disjunctness along with illustrations. Section 4 discusses design for resource disjunction using the developed mosaic convergence algorithm. Section 5 explains fault isolation and recovery schemes for reconfigurable fabrics using design disjunction. Evaluation results for several case studies are provided and discussed in Section 6. A comparison between the proposed work and modular redundancy schemes is presented in Section 7. Finally, Section 8 presents a brief conclusion.

2 Related Work

For contemporary reconfigurable devices, low-level hardware support for testing can incur a significant area overhead due to uncertainty in the logic and interconnect usage of the target applications. In some cases, the goals of testing have been limited to verifying the collective health of reconfigurable fabrics, whereas in the case of diagnostic testing, locations of faulty elements are also identified. Reconfigurability has been leveraged in various ways to enable online testing strategies which examine correctness throughout the system lifetime.

Table 1 summarizes features of related approaches along with the proposed scheme. Previous online diagnostic test schemes for reconfigurable fabrics [14], [20] provide fine resolution, although they require that the system be halted or become unprotected for extended periods before individual faulty elements can be identified. The ability to rapidly obtain information about faulty resources is a critical factor in realizing efficient self-repair. It facilitates fault evasion whereby faulty resources are avoided, or partially damaged resources are reassigned to other useful functionalities. Online fault localization techniques often consider the structural heterogeneity of contemporary reconfigurable hardware. Testing and fault isolation schemes for structures such as programmable logic, interconnect, and RAM have been developed through the years, based on the nature of each structure. For example, RAM-based testing has been extensively studied and the well-known MARCH algorithms [21] have been proven effective for diagnosis of RAM cells by applying a sequence of tests to each element in succession. Previous online fault isolation and recovery

TABLE 1
Comparison of Design Disjunction with Related Approaches

Approach	Run-time Fault Isolation	Resource Coverage: Resolution	Provable Multiple-fault Coverage	Error-tolerant Fault Isolation	Recovery Latency	Intrinsic Wear Leveling	PAR at Run-time	Advantage
STARs [14]	Yes	Logic: LUT	Yes	No	Exhaustive BIST Overhead	No	Required	Resource Recycling
R3TOS [15]	Yes	Logic: LUT	Yes	Yes	Exhaustive BIST Overhead	No	Required	Robust Control Mechanism
Module Diversity [16]	No	Logic: CLB	No	No	μs$\rightarrow$ ms	Yes	Unnecessary	Effective Aging Mitigation
Hahanov et al. [17]	No	Logic: CLB	Yes	No	Routing Overhead	No	Required	Provable Coverage
AGT [18]	No	Logic: slice	No	No	PAR Overhead	No	Required	Intrinsic Adaptation
Consensus-Based Evaluation [19]	Yes	Logic: slice	No	No	PAR Overhead	No	Required	Outlier Identification
Design Disjunction (approach herein)	*Yes*	*Logic: slice & Interconnect: PIPs*	*Yes*	*Yes*	*μs$\rightarrow$ ms*	*Yes*	*Unnecessary*	*Condensed Diagnosis*

approaches for FPGA logic using dynamic reconfiguration have relied on built-in self-test (BIST) [14], [22]. However, dedicated BIST structures including test pattern generators (TPGs) and output response analyzers (ORAs) are typically not available for FPGA platforms [22]. Modern FPGA architectures are also not entirely scan-ready. Thus, scan chains, TGPs, and ORAs are frequently implemented directly in the fabric using look-up tables (LUTs) and shift registers. As a consequence, BIST-inspired methods can increase FPGA resource requirements by up to 50 percent [23].

The BIST-based roving STARs test scheme in [14] partitions the reconfigurable fabric into tiles, and continuous online testing is carried out by roving a BISTer from one tile to another while the resources not used by the BISTer structure are dynamically reconfigured to maintain availability. Although failures are resolved at a fine resolution, data throughput must be suspended to copy state values prior to each tile movement. Resource recycling is also facilitated; however, fault isolation and recovery depend on the latency of BISTers to rove the device before encountering faulty elements. Another recent BIST-based fault-tolerant FPGA approach is illustrated by the reliable reconfigurable real-time operating system (R3TOS) [15] wherein a hardware microkernel (HWuK) provides a task scheduler, an allocator to manage FPGA resources for tile placement, and a configuration manager which converts commands issued by the scheduler and allocator into FPGA reconfiguration operations. To minimize single-point of failure exposures, HWuK components are realized by an 8-bit PicoBlaze processor occupying six block RAMs (BRAMs) and 500 configurable logic blocks (CLBs) protected with selective triple modular redundancy (TMR) and error-correcting code (ECC) bits whose resources also undergo periodic testing. The impact of BIST latency is masked by the use of hardware replication and voting.

To reduce the high complexity and cost of BIST, application-dependent BIST testing [20] focuses on the subset of resources used to maintain design functionality. Thus, exhaustive test vectors generated by a TPG and response analysis carried out by an ORA can be relaxed without continually engaging a dedicated reconfiguration controller to carry out the test. The work in [20] also demonstrates an effective application-dependent diagnosis for FPGA interconnects. Distinct test configurations are applied to modulate application LUT functionalities and study output patterns to discern which nets are faulty. These application-dependent approaches assume the resources undergoing diagnosis procedures are unavailable during diagnosis. Thus, methods which eliminate these limitations on availability are sought.

Alternative approaches that eliminate BIST area and power overheads, referred to as operational testing techniques, conduct functional tests via input data that are simultaneously used for normal throughput [19]. These techniques attain availability by relying on run-time inputs, computational redundancy, and output comparison to assess the subset of resources currently used by an application. Permanent and temporary fault monitoring for operational testing can be realized using concurrent error detection (CED) techniques based on duplication with comparison (DWC) or parity-based methods [24]. DWC that compares the Hamming distance between the outputs of two spatially redundant modules is compatible with recent multi-objective DSE approaches [25] which utilize a cost function that considers area requirements and resource utilization against overhead of reconfiguration time. In [18], another operational testing method based on adaptive group testing (AGT) for diagnosis of reconfigurable fabrics is described under a single-fault assumption. However, since the creation of test designs are adaptive based on outcomes of successive tests, the AGT method is unsuitable for high availability applications. Similar to iterative logic array (ILA) and array-based testing methods [26], most functional testing techniques are mainly used for testing a group of resources and provide no fault localization at a fine resolution. In this work, benefits of operational testing are explored with design disjunction to locate faulty resources while avoiding BIST overheads.

Other previous design-time approaches for run-time fault recovery have used genetic algorithms (GAs) [27] to evolve a pool of best-fit designs that exhibit resilience to various failures. The evolved designs are used at run-time to maintain system functionality. Although GAs can succeed in finding resilient designs, the number of evolved designs requiring functional evaluation is large, and also being a probabilistic process does not explicitly guarantee convergence. The work in [17] presents an algebraic method for devising an optimal remapping strategy for logic blocks at row and column levels to reduce recovery latency and minimize number of spare rows and columns required to tolerate a large combination of fault locations. Remapping by interchange of device columns and rows is still performed at runtime, which relies on an independent fault diagnosis process to locate faulty cells before identifying which resources to interchange. The consensus-based evaluation(CBE) method described in [19] generates, at design-time, a diverse pool of FPGA designs with alternative device resources. These designs are evaluated against each other using a duplex arrangement. Statistical clustering is used to identify operationally correct designs without the assumption of a golden element. The module diversity approach described in [16] provides yet another method for generating diverse designs at design-time for mitigating aging effects at run-time. The diverse designs can be deployed according to a scheduling policy that results in a steady stress distribution across resources to achieve an extended lifetime. The set of diverse designs also guarantees fault recovery under a single-fault assumption for all possible single CLB faults.

Unfortunately, none of the existing approaches demonstrate provable coverage for multiple faults nor do they allow the use of diverse designs for diagnostic tests to locate faulty resources. In this work, we describe an explicit method for generating the optimal number of DCs that guarantee recovery from multiple faults at fine granularity while providing rapid fault isolation. Broader surveys of recent techniques for fault tolerance, autonomous recovery, and self-healing of FPGA-based systems are presented in [28], [29], and [30], respectively.

3 Group Testing for Diagnosis of Reconfigurable Architectures

If a test is used to identify f defectives among T elements, where f is unknown, then a straightforward, albeit suboptimal, procedure is to evaluate each element individually. Assuming all tests are reliable, then the testing time complexity becomes $O(T)$. This cost can be considerably reduced by dividing the T elements into g subsets, or groups. The collective results after testing each group can be interpreted to identify the f defectives. The challenge is to sample the minimum number of groups sufficient to find the defectives. This is the basic idea behind group testing first introduced by Dorfman [31] for screening a large number of blood samples by pooling them together to reduce testing cost. Group testing has been adapted to diverse applications such as testing for manufacturing defects, DNA library screening, coding theory, software testing, and BIST-based diagnosis in digital systems [32]. Based upon how test groups are sampled, most group testing techniques can be classified into adaptive or non-adaptive categories.

3.1 Adaptive Group Testing

When using adaptive group testing, complete knowledge of how groups are sampled before testing begins is not specified. The groups are constructed iteratively based on each successive test outcome during the testing procedure. As testing progresses, the iterative sampling of groups narrows down the suspect set of faulty resources until defectives are identified. The binary search (BS) method described in [33] presents one of the simplest AGT algorithms. At the initial stage of BS, the set of scan cells to be tested, X, are considered suspect. The set X is partitioned into two groups, each of which is collectively tested using two scan chains. The BS technique is applied recursively to any erroneous group until faulty cells are singled out. A modified implementation of this algorithm was first proposed for functional testing of FPGAs in [18] under a single-fault assumption. Each test group is a set of resources which implement a functionally equivalent design. Initially, all resources in the reconfigurable container are deemed suspect. The test starts by dividing suspect resources among different functionally equivalent FPGA designs. The suspect set is narrowed down to those that implement a fault-affected design. The modified suspect set is iteratively divided and utilized by a new generation of test designs. The algorithm terminates when only a single cell remains in the suspect set, thus identifying the defective resource. The operational complexity of this algorithm depends on the maximum number of test designs allowed in every test generation. Thus, an overriding concern with AGT is the downtime needed to generate new test designs by repeatedly invoking the design flow which is infeasible on deployed real-time embedded systems.

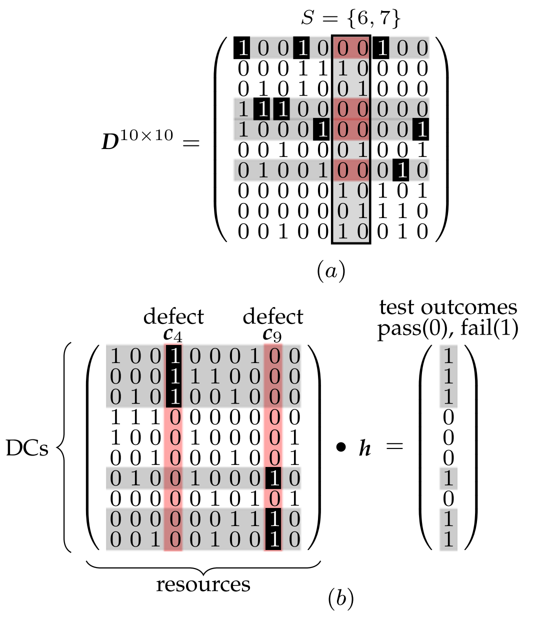
3.2 Non-Adaptive Group Testing

In the case of non-adaptive group testing, the sampling procedure for all groups is known apriori to the execution of tests. An intuitive way to model and describe the problem of fault isolation in FPGAs using this class of group testing techniques is through matrix algebra. The following notations are used throughout the paper:

- Design matrix $\boldsymbol{D}^{g\times T}$ is a binary matrix indicating the subset of resources used by each of g DCs. Rows in this matrix correspond to DCs whereas columns correspond to resources. An entry $k_{i,j}$ of $\boldsymbol{D}$ matrix is one if resource j is utilized by DC$_i$, and zero otherwise.
- Health vector $\boldsymbol{h}^{T\times 1}$ is a binary vector of length T representing the health of the T resources, i.e., an entry h_j is one if resource j is defective and zero if resource j is healthy.
- Outcome vector $\boldsymbol{o}^{g\times 1}$ is a binary vector of length g containing the error detection outcomes of all g DCs, i.e., an entry o_i is one if an erroneous outcome is detected while DC_i is deployed and zero if DC_i sustains correct operation.
- Set $\psi(\boldsymbol{v})$ is the subset of elements in binary vector v whose entries are one.
- $\omega(\boldsymbol{v})$ is the weight of binary vector v, i.e., number of elements whose entries are one.
- $\boldsymbol{\Gamma}_r^n$ is the set of all r-combinations of n elements.

The Outcome Vector, $\boldsymbol{o}^{g\times 1}$, can be given as follows:

$$\boldsymbol{o}^{g\times 1} = \boldsymbol{D}^{g\times T} \cdot \boldsymbol{h}^{T\times 1}. \quad (1)$$



$S = \{6, 7\}$
$\boldsymbol{D}^{10\times 10} =$
(a)
defect $\boldsymbol{c}_4$
defect $\boldsymbol{c}_9$
test outcomes pass(0), fail(1)
DCs
$\bullet\ \boldsymbol{h} =$
resources
(b)

Fig. 2. (a) Example of two-disjunct design matrix. (b) Conventional diagnosis decoder.

The objective is to recover the health vector $\boldsymbol{h}$ given that both the design matrix and the outcome vector are known. The health vector can be efficiently recovered if the design matrix obeys the f-disjunctness property and no more than f resources are defective [34]. The f-disjunctness property constrains how alternative groups are overlapped such that f-diagnosability still holds. It provides an efficient strategy to distribute each possible subset of resources of size up to f among a unique subset of DCs. Therefore, defective resources can be identified by finding the common resources among faulty DCs. The matrix $D^{g\times T}$ is considered f-disjunct if and only if for any possible combination of columns, S, of size f, every column not in S has at least δ row elements whose entries are one and all entries of the columns S are zero [35]. This can be expressed as:

$$\forall S \in \Gamma_f^T, \sum_{i=1}^{g}\left(D_{i,j} = 1 \ \wedge \bigcup_{k\in S} D_{i,k} = 0 \right) \geqslant \delta, \tag{2}$$

where $1 \leqslant j \leqslant T$ and $j \notin S$.

The parameter δ represents the number of rows that satisfy the left side of inequality in Eq. (2). We refer to this parameter as the *disjunction factor*. The minimum value of δ necessary to ensure f-disjunctness is 1 in which all possible combinations of up to f faulty resources can be identified provided that all tests are reliable, i.e. each faulty DC will generate a detectable erroneous outcome. Fig. 2a shows a two-disjunct matrix and a one subset of columns, S, of size 2 that meets the condition given by Eq. (2) for $\delta = 1$.

The decoding procedure to infer the sparse health vector assuming reliable testing is illustrated through a binary comparison between each column vector, $\boldsymbol{c}$, of the $\boldsymbol{D}$ matrix and the outcome vector $\boldsymbol{o}$. If the subset of elements of $\boldsymbol{c}_k$ having value equal to one is fully contained within the subset of elements of the outcome vector $\boldsymbol{o}$ having value equal to one, then the resource k must be faulty. Thus, the health vector can be obtained as follows:

$$\boldsymbol{h} = \{h_k \,|\, h_k = \begin{cases} 1 & if \ \ \psi(\boldsymbol{c}_k) \subseteq \psi(\boldsymbol{o}) \\ 0 & otherwise \end{cases}, \ 1 \leq k \leq T\}. \tag{3}$$

Fig. 2b illustrates how the same two-disjunct matrix is used to single out the two defective resources, four and nine, using the described decoding method. In this example, the sparse health vector is given as:

$$\boldsymbol{h} = (\, 0\,0\,0\,1\,0\,0\,0\,0\,1\,0 \,)^T. \tag{4}$$

Although the binary decoder is efficient, there are two main challenges to properly exploit this technique for fault isolation of reconfigurable hardware. The first challenge is the well-known limitation of low coverage from functional testing which can introduce a sampling noise to the binary decoding method leading to misdiagnosis. Hence, a suspiciousness ranking metric that classifies resources according to their existence rate in failed DCs is developed instead of binary decoding methods. Additionally, f-disjunctness for $\delta > 1$ along with the proposed ranking metric are shown to be effective for surmounting the low coverage issue of functional testing as explained in Section 5.2. Since all DCs implement the same application functionality while utilizing a disjunct set of T resources, each DC requires the same resource count. The second challenge is to construct a constrained f-disjunct design matrix for any given T and with rows of equal weight dictated by the application size, R. Available techniques used to construct f-disjunct matrices stipulate a set of conditions on matrix size and the row weights which preclude the flexibility needed to meet design and resource count constraints of operational testing of reconfigurable fabrics. In this work, a new combinatorial search algorithm is described to achieve f-disjunctness for any given design parameters T, R, and δ. In Sections 4 and 5, solutions to these two challenges are discussed with results demonstrating feasibility and advantages of the proposed approach.

4 Design for Disjunction on Reconfigurable Architectures

Design disjunction realizes a set of f-disjunct DCs, each of which implements the same application functionality, and then employs them to locate and evade defective resources during system lifetime while maintaining optimal availability. These DCs are produced prior to the test procedure; therefore, only partial reconfiguration overhead of existing DCs is incurred during fault diagnosis and recovery. Fault tolerance is achieved by run-time reconfiguration to load one of the bitfiles from the subset of DCs which does not utilize defective resources. The constructive property of f-disjunctness is shown to be effective for extracting highly fault-resilient DCs against logic and interconnect failures.

In this work, FPGA-based fault scenarios are considered for evaluation of design disjunction since FPGAs are the prominent form of contemporary reconfigurable hardware. Modern FPGAs have multiple levels of logic cell granularity. For instance, basic logic elements such as LUTs and flip-flops of Xilinx FPGAs are organized into logic slices which are considered the most primitive programmable logic blocks. As such, design disjunction is examined at the slice level. Thus, the columns of the design matrix $\boldsymbol{D}$ correspond to slices while rows represent DCs. We also focus on logic fault localization. However, the proposed work can be combined with other application-dependent interconnect testing such as [20] for fault isolation at the level of interconnect points.

Assuming an application is synthesized to a minimum of R slices, then the weight, i.e. the number of non-zero elements, of every row of the design matrix must equal R. The problem of constructing f-disjunct matrices has been increasingly studied within coding theory literature [34]. For the interest of this work, we empirically evaluate the lower bound on DC count required to reach f-disjunction using the developed mosaic convergence algorithm. Let the notation *(T,R,f)*-disjunct matrix denote an f-disjunct design matrix whose rows have exactly R non-zero entries out of T. Algorithm 1 shows the pseudocode for the proposed mosaic convergence approach for constructing such a matrix. Starting with an initial row that has R non-zero entries (lines 4-7), each added row represents the best-found row vector that maximizes the accumulative *disjunction ratio* (lines 36-49). The disjunction ratio is defined as follows:

Definition 4.1. *Disjunction ratio (DR) is the proportion of $\boldsymbol{\Gamma}_f^T$ elements that satisfy the condition stated in Eq. (2).*

The binary coverage matrix $\boldsymbol{\lambda}$ (line 9) tracks whether each combination $S \in \boldsymbol{\Gamma}_f^T$ has satisfied the condition in Eq. (2). Every added row is initially a T-dimensional row vector $\boldsymbol{v}$ of weight equals T (line 12). The combinatorial search for optimal v, requires two nested sequential loops (lines 17-31) which examine each non-zero element in v and pick the element which, if flipped to zero, yields the largest increment to the disjunction ratio DR. This latter step is repeated until the weight of the vector $\boldsymbol{v}$ is reduced to R. Once an optimal row vector is found, the coverage matrix $\boldsymbol{\lambda}$ is updated to include the incremental coverage of each row (lines 36-49). The row-by-row construction of design matrix $\boldsymbol{D}$ terminates once the DR value reaches its maximum value of 1 (line 11).

The complexity of the binary search for each new row is largely determined by T and the cardinality of set $\Re \subseteq \boldsymbol{\Gamma}_f^T$ that have not yet satisfied the condition expressed in Eq. (2). The cardinality of $\Re$ decreases exponentially as number of rows in the D matrix increase. For search of the first few rows, the search space for optimal v is still large, which rapidly decreases as more rows are added to the D matrix. To decrease the execution time of the algorithm, one option is to limit the combinatorial search to a randomly selected subset of $\Re$. This will increase the speed of the construction algorithm at the expense of obtaining a suboptimal v in each row iteration. The effect of this suboptimality appears in the final solution as an increase in g, or number of required DCs to achieve f-disjunctness. In this work, we utilized exhaustive combinatorial search to capture the lower bound on number of DCs needed to achieve the discussed FT objectives, although search can be relaxed in practice. The constructed design matrix is then used to define the set of placement constraints supplied to the design tools to implement disjunct DCs.

Algorithm 1. Mosaic Convergence Algorithm for Constructing (T,R,f)-Disjunct Design Matrix

```
   Procedure construct $(T,R,f)$-disjunct matrix
   Input: $T$: Total Number of Resources
          $R$: Required Resources to Implement Application
          $f$: Number of Defects
          $\delta$: Disjunction Factor
   Output: Design Matrix, $\boldsymbol{D}^{g\times T}$.
1  $\phi := \binom{T}{f} = \frac{T!}{f!(T-f)!}$
2  $\varepsilon := \phi \times (T-f)$ // binary check count
3  $DR := 0$
4  Generate a random row vector $\boldsymbol{v}$, s.t.: $length(\boldsymbol{v}) = T$
   and $\omega(\boldsymbol{v}) = R$
5  $g := 1$ // point to the first row of $\mathbf{D}$
6  $\boldsymbol{D}_g := \boldsymbol{v}$ // insert $\boldsymbol{v}$ as the first row of the design
   matrix
7  $g := g + 1$
8  $\boldsymbol{C} := \boldsymbol{\Gamma}_f^T$ // set of all f-combinations out of T
9  $\lambda^{\phi\times T} := [\delta]^{\phi\times T}$ //initialize binary coverage
   matrix entries to $\delta$
10 $DR\_func(\boldsymbol{v})$ // call function $DR\_func$ to update $DR$
   after inserting the row vector $\boldsymbol{v}$
11 while ($DR \neq 1$) do
12   $v := [1]^{1\times T}$ // start with a row vector
       $\boldsymbol{v}$ s.t. $length(\boldsymbol{v}) = \omega(\boldsymbol{v}) = T$
13   $S\_max := \boldsymbol{C}_{z\_max}$
14   for each $k \in S\_max$ do
15     $v_k := 0$
16   while ($\omega(\boldsymbol{v}) \neq R$) do
17     $max := 0$
18     for $i := 1$ to $T$ for do
19       if ($v_i \neq 0$) then
20         $\boldsymbol{t} := \boldsymbol{v}$
21         $\boldsymbol{y} := \lambda_{z\_max}$
22         $t_i := 0$
23         $count := 0$
24         for each $S \in \boldsymbol{C}$ s.t. $i \in S$ do
25           for $j := 1$ to $T$ do
26             if ($t_j = 1 \ \wedge \ y_j \neq 0$) then
27               $y_j := y_j - 1$
28               $count := count + 1$
29         if ($count \ > \ max$) then
30           $top\_entry\_index := i$
31           $max := count$
32     $v_{top\_entry\_index} := 0$
33   $\boldsymbol{D}_g := \boldsymbol{v}$
34   $g := g + 1$
35   $DR\_func(\boldsymbol{v})$
   // update DR after inserting a new row
36 Function DR_func(a)
37   $count := 0$
38   $max := 0$
39   for $z := 1$ to $\phi$ do
40     $S := \boldsymbol{C}_z$
41     if ($\forall \ k \in S, \ \boldsymbol{a}_k = 0$) then
42       for $j := 1$ to $T$ s.t. $j \notin S$ do
43         if ($\boldsymbol{\lambda}_{z,j} \neq 0 \ \wedge \ \boldsymbol{a}_j = 1$) then
44           $\boldsymbol{\lambda}_{z,j} := \boldsymbol{\lambda}_{z,j} - 1$
45           $count := count + 1$
46     if ($\omega(\boldsymbol{\lambda}_z) \ > \ max$) then
47       $z\_max := z$
48       $max := \omega(\boldsymbol{\lambda}_z)$
49   $DR := DR + \frac{count}{\varepsilon\times\delta}$
```

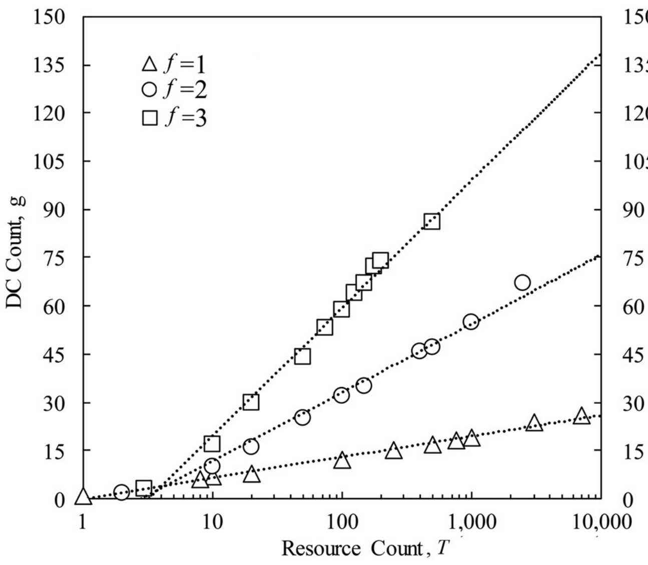


Fig. 3. Required number of DCs versus resource count for typical values of f ($\delta = 1$).

The mosaic convergence algorithm was implemented on an Intel quad-core processor based PC design station. The number of DCs g required to reach f-disjunctness with respect to T and f is obtained for $\delta = 1$. Fig. 3 shows collected g values for $f = 1$, 2, and 3. The logarithmic trend lines indicate that g grows linearly as resource count increases exponentially. The advantageous logarithmic dependence of g on resource count T obtained by the mosaic convergence procedure is consistent with results from other probabilistic methods for constructing unconstrained disjunct matrices [36], [37]. Fig. 3 also shows the non-linear increase in g for increasing f. The small number of disjunct DCs signifies the advantage of design disjunction to lower testing cost and recovery overhead.

5 Design Disjunction for Fault Tolerance

5.1 Fault Diagnosis Using Design Disjunction

The binary decoder described in Section 3.2 provides only binary diagnostic data which can lead to incorrect fault diagnosis in the presence of inarticulate tests. Instead, a ranking scheme that assesses resources according to their existence rate in failed DCs can reveal a more accurate estimate of the failure state of the resources. For each resource, the proportion of failed DCs that utilize the resource is computed and compared with other resources. This ratio is referred to as *fault sensing ratio* (FSR) and can be expressed as follows:

$$FSR_i = \frac{\left| \bigcup_{k=1}^{g} D_{k,i} \mid \; D_{k,i} = 1 \; \wedge \; o_k = 1 \right|}{\omega(\boldsymbol{c}_i)} \;, \; 1 \leq i \leq T, \quad (5)$$

where $\boldsymbol{c}_i$ is the ith column vector of the design matrix D.

A resource with a large FSR has a high likelihood of being faulty. To illustrate how FSR is obtained, the health vector $\boldsymbol{h}$ given by the example described in Section 3.2 can be rewritten using FSR for each cell, as follows, in which faulty resources get the highest FSR values.

$$\boldsymbol{h} = (\; 0.\bar{3} \quad 0.\bar{6} \quad 0.\bar{3} \quad 1 \quad 0.\bar{6} \quad 0.\bar{6} \quad 0.\bar{6} \quad 0.\bar{6} \quad 1 \quad 0 \;)^T$$

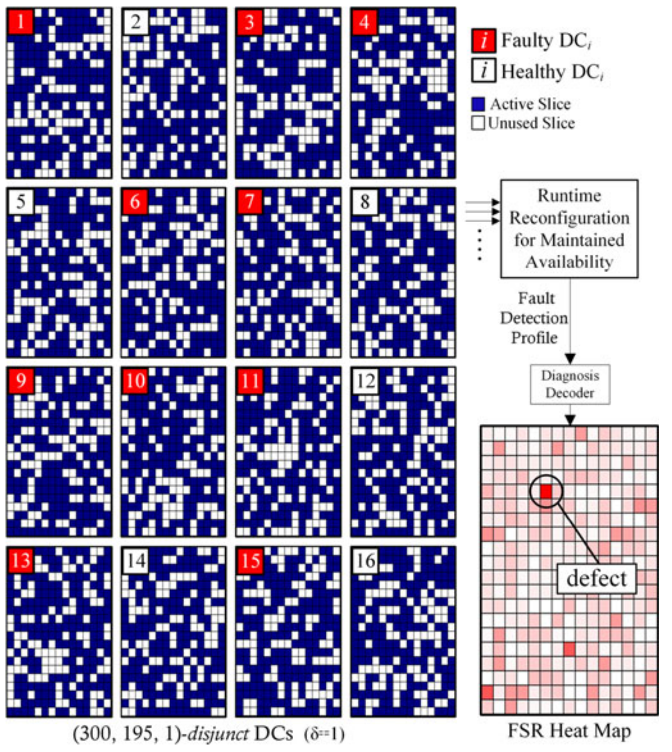


Fig. 4. Fault diagnosis using the FSR metric.

Similarly, the cumulative sum of FSR, denoted as $CFSR$, for all resources used by each DC yields a failure ranking metric for DCs. The $CFSR$ is used to determine the best operational DC if fault isolation at the design configuration level is sought.

We first focus on the case of ideal test coverage in which all fault-affected DCs manifest at least one erroneous functional output. Fig. 4 illustrates an example of a single fault isolation case on a reconfigurable partition of size $20 \times 15 = 300$ slices for an application mapped to 195 slices. Using the mosaic convergence procedure in Algorithm 1, 16 DCs (indexed 1-16) are found sufficient to achieve one-disjunctness for $\delta = 1$ in this example. The resource grouping defined by a $(300, 195, 1)$-disjunct design matrix is shown by the dark blue cells for each DC. Based on fault detection outcomes after evaluating all the 16 DCs, the FSR value for each slice is computed. The highest observed FSR reveals the location of faulty slice as depicted by the FSR heat map.

To examine the quality of fault isolation using the proposed ranking method, the terms *isolation accuracy* and *fault coverage* are defined as follows:

Definition 5.1. *Isolation accuracy is the number of non-faulty resources that have lower FSR values than all defectives, divided by the total number of resources.*

For instance, given a pool of 1,000 resources having two defects, an isolation accuracy of 95 percent indicates that $\lfloor 998 \times 95\% \rfloor = 948$ of non-faulty resources score lower FSR values than the two defects.

Definition 5.2. *Fault coverage is the proportion of all combinations of faulty resources of size up to f that attain a specified isolation accuracy.*

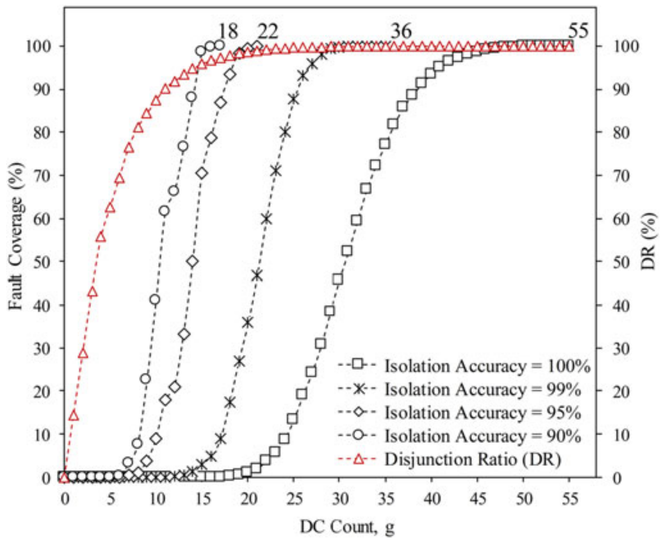


Fig. 5. Isolation accuracy versus g ($T = 1{,}000$, $f = 2$, $\delta = 1$).

Fig. 5 shows the required number of DCs, g, to reach various isolation accuracies and their fault coverage values. The results also demonstrate how Algorithm 1 progresses towards the termination criteria, i.e. $DR = 100$ percent, as g increases. The resource count T chosen for this analysis equals $1{,}000$ and disjunction parameters are $f = 2$ and $\delta = 1$. In this case, 55 DCs are sufficient to identify all $\binom{1{,}000}{2} + \binom{1{,}000}{1}$ possible fault locations with 100 percent isolation accuracy. The value of g can be considerably reduced while maintaining a high isolation accuracy. A reduction of 36.4 percent (61.8 percent) in g results in a slight decrease in isolation accuracy of 1 percent (5 percent). This tradeoff between isolation accuracy and number of required tests can be conducted based on system reliability goals, e.g., the extent sufficient to achieve fast self-repair. It is important to note again that these simulation results are collected under the conditions of reliable tests. It is expected that g is increased to tolerate inarticulate tests while maintaining equivalent isolation accuracy as demonstrated in Section 5.2.

5.2 Inarticulate Operational Testing

In the preceding analysis, we have assumed that a test outcome generated by a fault detection scheme embedded within each DC is reflective of the actual health state of used resources. However, this assumption for functional testing of digital designs cannot be guaranteed for various reasons. These include low test coverage due to node's controllability and observability constraints, common mode failures, or stuck-at 0 fault conditions in the fault detection logic. Error-resilient NGT was previously investigated through probabilistic and theoretical analysis with direct numerical simulations [37], [38]. In Section 3.2, a discussion was provided for the classical requirement to obtain f-disjunction which states that δ must be greater than or equal 1. As δ increases beyond 1, the effect of inarticulate tests on the decoding procedure can be masked. In the context of operational testing of reconfigurable hardware, increasing the disjunction factor δ results in an increased number of alternative DCs. Since resources are sensitized in a diverse way as the device is reconfigured to different DCs, diversity among DCs enables a better collective diagnostic coverage

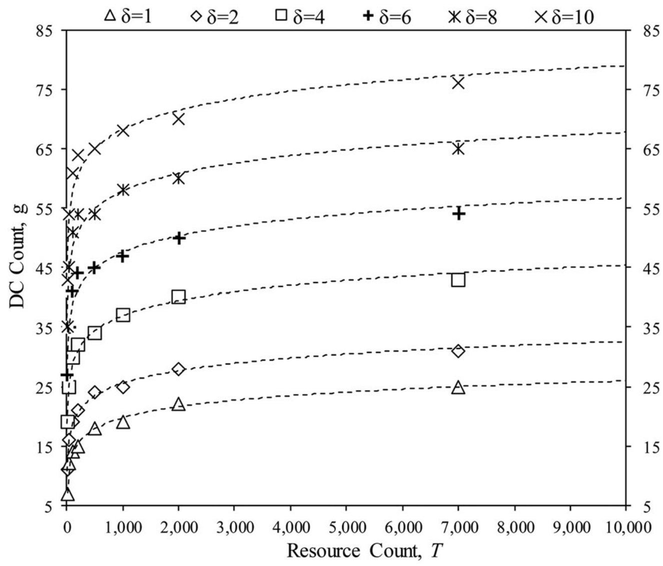


Fig. 6. DC count for increasing δ ($f = 1$).

to attenuate the chance of false test outcomes during individual tests.

In this work, we study how such an extension affects fault diagnosis using the proposed ranking scheme. The described combinatorial construction method given by the mosaic convergence procedure in Algorithm 1 is also used to realize design disjunction for $\delta > 1$. Fig. 6 shows the number of DCs for one-disjunctness and selected δ values. It is evident that design disjunction for $\delta > 1$ is achieved at modest linear increase in DC count g. For instance, the case of $7{,}000$ resources indicates that δ can be increased by an order of magnitude from $\delta = 1$ to $\delta = 10$ while only roughly tripling the number of DCs required. In Section 6, we evaluate the effect of increasing δ on fault diagnosis for various case studies in which we compare the isolation accuracy under the low coverage of operational testing.

5.3 Fault Recovery Using Design Disjunction

The combinatorial characteristics of f-disjunct design matrices add another advantage for design disjunction. The definition expressed in Eq. (2) implies that any f-disjunct set of DCs should guarantee that for any possible accumulation of f faulty resources there exists at least one DC whose resource set does not include a defective. This implication should not be considered as the upper bound on the number of recoverable defectives. Since hardware utilization ratio R/T can increase or decrease the sparsity of design matrix, it is possible to guarantee fault evasion for larger than f defectives. The normal probability $p_{dc\_nf}(d)$ that up to d defective resources are not used by a DC is given as:

$$p_{dc\_nf}(d) = \prod_{k=1}^{d}\left(1 - \frac{R}{T-k-1}\right),\ d \geqslant 1. \tag{6}$$

Thus, *recovery coverage* (RC), defined by the probability of recovery for g DCs, can be computed for any accumulated fault count d as:

$$RC(d) = 1 - \left[1 - p_{dc\_nf}(d)\right]^{g},\ d \geqslant 1. \tag{7}$$

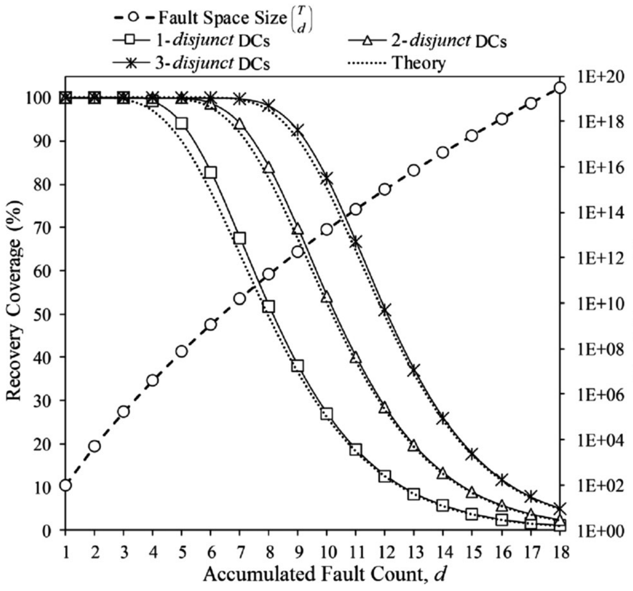


Fig. 7. Recovery coverage of disjunct DCs ($T = 100$, $R = 30$, $\delta = 1$).

In order to examine the recovery behavior of the proposed method, three sets of f-disjunct designs for $f = 1$, 2, and 3 were tested against all possible set of fault locations Γ_d^T for varying accumulated fault count d. Fig. 7 compares simulation results against our model given by Eq. (7). Recovery coverage on the left vertical axis also indicates the proportion of Γ_d^T combinations of defective(s) that were successfully evaded by at least one DC. All three disjunct sets exhibit high fault resilience for fault count d larger than f.

A target recovery rate can be met by choosing the appropriate hardware utilization as indicated in Eq. (6). For practical considerations, the optimal number of DCs for recovery during the system lifetime can be generated at design-time and stored in an off-chip flash memory.The data in the external flash memory can be protected using hardware redundancy or error correction schemes in addition to functional verification by CED which is resident on the FPGA.

5.4 Incidental Disjunction for Interconnect Fault Tolerance

Contemporary reconfigurable devices utilize hundreds of thousands of routing points. For instance, Xilinx 7-series FPGAs fabricated in a $28\,\mathrm{nm}$ process allow over $3{,}500$ programmable interconnect points (PIPs) to be defined in each switch tile of the device. This presents a significant challenge for run-time interconnect testing and diagnosis. Specialized functional testing for interconnects based on output pattern analysis as in [20] and [39] has been shown to be effective for diagnosis at the net level of a target design. However, a net in a design can utilize a considerable number of PIPs spanning multiple switch tiles that can prolong the self-repair process. Since allocation of interconnect resources is precipitated by mapping and placement of logic resources [40], a design disjunction in the logic fabric has been demonstrated to also confer significant incidental disjunction in interconnect resources. This property effectively extends fault recovery to routing fabrics as demonstrated in Section 6.2.

6 Evaluation

6.1 Evaluation Setup

The proposed work is initially evaluated on a set of MCNC and ISCAS benchmarks through hardware simulations to show its applicability to a variety of applications. A modularized AES128 encryption core is selected as a realistic target application for the hardware prototype. The actual hardware demonstration is performed on the commercial Xilinx KC705 FPGA evaluation board. The KC705 board features: 28 nm-based Kintex-7 FPGA, 1 GB DDR3 memory, 128MB linear flash memory, and a joint test action group (JTAG) interface. For hardware simulation, a software-based CED scheme is utilized to detect failures during simulation. Parity-based and DWC error detection methods are adopted in the hardware prototype. For all case studies, Xilinx 7-series FPGAs using Xilinx design toolsets are used to generate disjunct DCs.

The design flow for the evaluation framework is depicted in Fig. 8. The flow starts from a conventional design in a hardware description language using Xilinx's ISE synthesis tool. The synthesized netlists for target application are imported to Xilinx's PlanAhead to generate the physical implementation of all disjunct DCs. To enable partial reconfiguration support in the PlanAhead tool, a reconfigurable partition (RP) must be floorplanned such that it contains T resources necessary to realize the disjunct DCs. The RP is interfaced with the static region (SR) outside the RP through proxy LUTs. All disjunct DCs must use the same proxy logic for the target application's input and output ports which is possible by locking all port sets with the `LOC` constraint. Each DC is defined as a distinct reconfigurable module

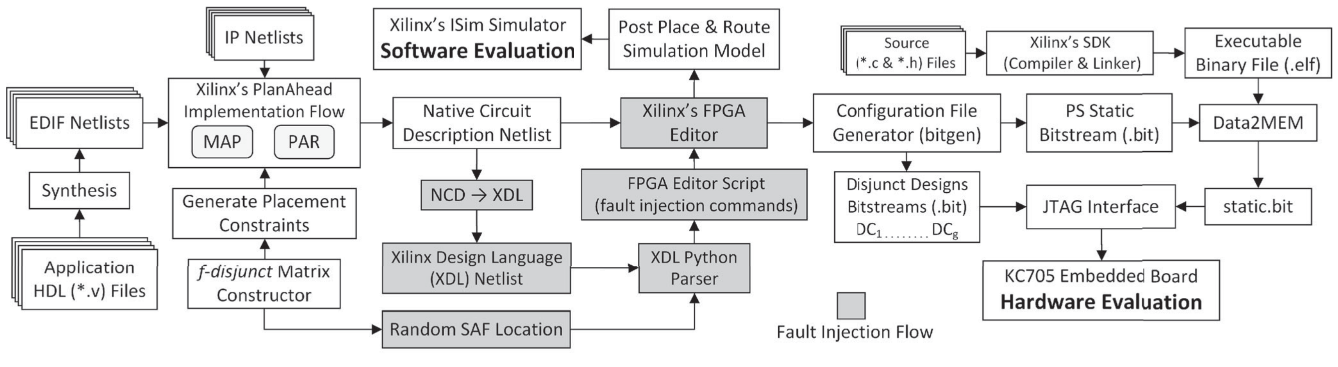

IP Netlists
Xilinx's ISim Simulator
Software Evaluation
Post Place & Route Simulation Model
Source (*.c & *.h) Files
Xilinx's SDK (Compiler & Linker)
Executable Binary File (.elf)
EDIF Netlists
Xilinx's PlanAhead Implementation Flow
MAP
PAR
Native Circuit Description Netlist
Xilinx's FPGA Editor
Configuration File Generator (bitgen)
PS Static Bitstream (.bit)
Data2MEM
Synthesis
Generate Placement Constraints
NCD → XDL
FPGA Editor Script (fault injection commands)
Disjunct Designs Bitstreams (.bit) DC1........DCg
JTAG Interface
static.bit
Application HDL (*.v) Files
f-disjunct Matrix Constructor
Xilinx Design Language (XDL) Netlist
XDL Python Parser
KC705 Embedded Board
Hardware Evaluation
Random SAF Location
Fault Injection Flow

Fig. 8. Framework of demonstration system.

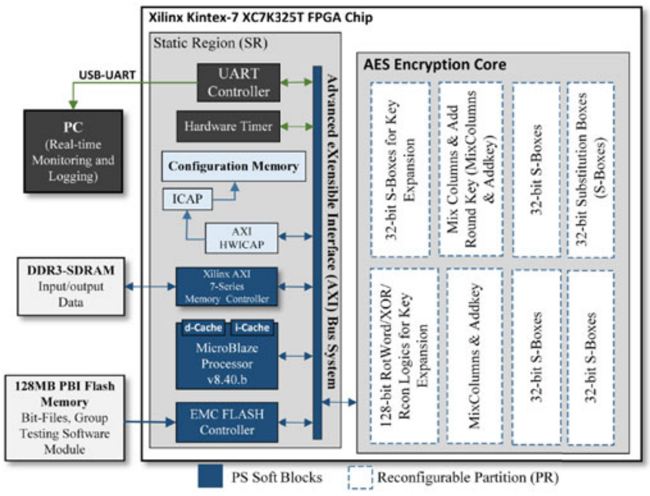

Xilinx Kintex-7 XC7K325T FPGA Chip
Static Region (SR)
USB-UART
UART Controller
Hardware Timer
Configuration Memory
ICAP
AXI HWICAP
PC (Real-time Monitoring and Logging)
DDR3-SDRAM Input/output Data
Xilinx AXI 7-Series Memory Controller
d-Cache
i-Cache
MicroBlaze Processor v8.40.b
128MB PBI Flash Memory Bit-Files, Group Testing Software Module
EMC FLASH Controller
Advanced eXtensible Interface (AXI) Bus System
AES Encryption Core
32-bit S-Boxes for Key Expansion
Mix Columns & Add Round Key (MixColumns & Addkey)
32-bit S-Boxes
32-bit Substitution Boxes (S-Boxes)
128-bit RotWord/XOR/ Rcon Logics for Key Expansion
MixColumns & Addkey
32-bit S-Boxes
32-bit S-Boxes
PS Soft Blocks
Reconfigurable Partition (PR)

Fig. 9. Block diagram of hardware demonstration system.

(RM) inside the RP. Resource allocation for each RM is dictated by the design matrix constructed for the target application according to the design parameters discussed in Section 4. Resource allocation for each DC is added to the design flow by defining the placement `AREA_GROUP` and `CONFIG_PROHIBIT` constraints in the user constraints file (UCF) for each RM. The PlanAhead tool then generates Xilinx's native circuit description (NCD) netlist for each RM.

The stuck-at fault (SAF) model is adopted for fault injection in this evaluation. Fault injection is incorporated into the flow using Xilinx's FPGA Editor which can inject SAF into NCD netlists at any randomly chosen location. Resource information for generating appropriate fault injection commands for the FPGA Editor tool are extracted from Xilinx design language (XDL) netlists. For hardware simulation of each benchmark, a post PAR simulation model is generated from each NCD netlist before Xilinx's ISim simulator is invoked to verify functionality of each DC. To drive each simulation case, a subset of random inputs generated from a uniform distribution are used to mimic run-time operational inputs. It is worth noting that operational testing using concurrent error detection schemes employs a functional fault model (FFM) which encompasses SAF and a wide range of failure modes that can alter application functionality.

The considered AES encryption core for the hardware prototype is comprised of non-linear substitution boxes, a key expansion and addition units, and other logic blocks for shifting and mixing columns of the state matrix where input words are arranged. The AES core is decomposed into eight modules each of which has its own embedded error detection domain. Fig. 9 shows a block diagram for the hardware demonstration system on the KC705 FPGA board. Error detection schemes for the AES modules are derived mostly from [41]. An embedded MicroBlaze processor orchestrates execution flow of fault recovery and diagnosis, and constitutes a golden element in this prototype. Partial reconfiguration (PR) using the internal configuration access port (ICAP) is utilized for partial reconfiguration to minimize reconfiguration overhead. Xilinx provides the AXI_HWICAP IP core and a set of basic library functions supplied with the Xilinx's software development kit (SDK) that are used to control partial reconfiguration via the ICAP at the system level. The advanced extensible interface (AXI) bus system is used to interface the processor with the ICAP, memory interfaces, RPs, and other IPs used in the prototype.

Design disjunction is evaluated on the hardware platform using high-resolution image data which reside in the external DDR3 during the recovery process. A hardware timer is attached to the developed system bus to accurately capture system throughput and processing time of fault diagnosis flow. Xilinx's IPs which form the processing system (PS) including the MicroBlaze core, memory and communication interfaces, and ICAP reconfiguration logic, reside in the SR of the device. Partial reconfiguration is integrated in this prototype by defining a distinct RP for each AES module. Disjunct RMs are then defined and added for each RP. The design flow of the hardware prototype is extended from the implementation steps of experimental simulation. The static bitfile for the SR and partial bitfiles for each RP are obtained from the NCD netlists using the Xilinx's BitGen tool. The software module running on the embedded processor developed for the prototype using the Xilinx's SDK is combined with the static bitfile using Xilinx's Data2MEM tool before programming the FPGA board through its JTAG interface. Partial bitfiles for all RPs are stored in the off-FPGA flash memory chip before the evaluation begins. When partial reconfiguration is required, the embedded MicroBlaze processor moves each partial bitstream in the flash memory to the DDR3 memory before being written by the ICAP.

The evaluation process including resource allocation for design disjunction, fault injection, and simulation, is carried out by a Python-based software module that automates design and simulation tasks by invoking all required Xilinx tools through external system commands. The Python module also parses post PAR design files to extract delays and build a slice-level netlist using a net connectivity graph with associated functionality and routing resource information. This netlist is used to examine the recovery rate in relation to logic resources and PIPs.

6.2 Design Parameters and Results

For each MCNC and ISCAS benchmark, two f-disjunct sets of DCs are generated for $f = 1$ and $f = 2$. Table 2 lists the isolation accuracy results averaged over $1{,}000$ experimental runs on all benchmarks for $f = 1$ and $f = 2$. Results include the 95 percent confidence interval (CI) and the area requirements indicated by parameters R and T. In this evaluation, T values are selected such that the area overhead $T/R \approx 2$ and $T/R \approx 3$ for $f = 1$ and $f = 2$, respectively, to demonstrate adaptation to various design parameters. The execution time of the mosaic convergence algorithm, denoted by τ_{mc}, to generate the (T,R,f)-disjunct design matrix for each benchmark is also included. For this evaluation, design disjunction for each benchmark is realized using $\delta = 1$ to observe the effect of inarticulate operational testing on fault isolation. As discussed in Section 4, the execution time of the mosaic convergence algorithm depends largely on T and size of Γ_f^T. The average execution time of the algorithm for the application set examined in this evaluation is 89.8 ms (61.1 s) for $f = 1$ ($f = 2$). Table 2 also shows that the

TABLE 2
Isolation Accuracy Results ($\delta = 1$)

		$f = 1$						$f = 2$					
					Isolation Accuracy (%)						Isolation Accuracy (%)		
Benchmark	R	T	g	τ_{mc}	μ	95% CI		T	g	τ_{mc}	μ	95% CI	
Circuit				(ms)		lower	upper			(s)		lower	upper
alu4	73	144	15	41	96.86	96.07	97.65	198	41	12.74	95.78	93.89	97.67
c880	16	30	10	7	95.80	93.85	97.75	45	25	0.057	95.56	93.54	97.57
misex3	103	198	15	98	91.73	89.28	94.18	286	44	51.7	88.16	84.34	91.99
exp5	22	40	11	9	97.17	96.28	98.07	66	29	0.161	93.42	90.19	96.64
vda	43	84	14	13	98.32	97.15	99.50	119	35	1.97	97.13	95.12	99.15
c6288	139	256	15	211	99.14	98.53	99.75	390	48	174.7	97.01	94.69	99.33
seq	132	252	15	205	91.71	89.69	93.74	385	47	170.3	89.90	86.49	93.32
apex4	70	136	14	31	98.56	97.75	99.37	204	41	14.7	97.40	95.87	98.94
des	146	275	16	262	97.31	96.26	98.35	391	48	179.8	92.67	89.55	95.79
c3540	58	112	14	21	97.66	96.31	99.01	162	38	5.97	96.67	95.16	98.19
average	–	–	–	89.8	96.43	95.11	97.74	–	–	61.12	94.37	91.88	96.86

average isolation accuracy over all benchmarks for $f = 1$ ($f = 2$) is 96.4 percent (94.4 percent). Although the obtained isolation accuracy results are still promising, it is evident that design disjunction for $\delta > 1$ is needed to overcome the impact of low test coverage. Test coverage also depends on the quality of input test patterns, a higher isolation accuracy can be achieved if specialized high-coverage test patterns generated by conventional ATPG tools at design-time are used at run-time.

Design disjunction for $\delta > 1$ is also evaluated to demonstrate feasibility to reach optimal fault isolation under inarticulate testing. Table 3 shows how design disjunction for a moderate increase in disjunction factor δ results in a greater than 99 percent isolation accuracy for all selected benchmarks. The three selected benchmarks include the misex3 benchmark which gives the worst combined isolation accuracy for $f = 1$ and $f = 2$ using $\delta = 1$. Nevertheless, isolation accuracy exceeding > 99 percent given by the upper 95 percent CI is reached using $\delta = 5$. A diminishing return in improving isolation accuracy is also observed as δ increases. Thus, the range $1 \leqslant \delta \leqslant 11$ can be chosen for an optimal tradeoff between isolation accuracy and g. A linear dependency of g on δ is also observed that is consistent with the analysis provided in Section 5.

Fig. 10 reports fault recovery results for the exhaustive fault coverage evaluation on logic and PIPs for $f = 1$ and $\delta = 1$. The design parameters for these benchmarks are similar to those listed in Table 2. It is evident that design disjunction allows the ratio of shared PIPs among DCs to be much lower than that of logic resources. This is attributed to the PAR mechanism in the FPGA tool and its reaction to the diverse logic realizations. Also, it translates into an increase in the likelihood of finding at least one DC that avoids all faulty resources as confirmed here for logic slices and PIPs.

To observe the impact of design disjunction on application performance, the timing slacks along critical paths of all DCs are compared to the total slack of baseline design for each benchmark. The baseline design is the conventional physical implementation of an application inside its dedicated RP without resource constraints. For typical implementation, PAR algorithms search for the best placement and routing to meet timing constraints. Total slack s is given by post PAR timing reports as follows:

$$s = t_{target} - t_{total} = t_{target} - [t_{cp} - t_{cps} + t_{cu}], \qquad (8)$$

where t_{target} is target clock period, t_{total} is total delay, t_{cp} is critical path delay, t_{cps} is clock path skew, and t_{cu} is clock uncertainty. t_{target} is set such that the total slack of baseline design is 2 ns. Figure 11 shows s and t_{cp} data for each benchmark. The average increase in t_{cp} compared to the baseline design is 1.49 percent and the average decrease in the ratio of the total slack to the total delay is only 1.78 percent. It is also observed that the top-performing DC can be slightly

TABLE 3
Isolation Accuracy versus δ for Selected Benchmarks ($f = 1$)

	misex3					c3540					alu4				
			Isolation Accuracy (%)					Isolation Accuracy (%)					Isolation Accuracy (%)		
δ	g	τ_{mc}	μ	95% CI		g	τ_{mc}	μ	95% CI		g	τ_{mc}	μ	95% CI	
		(ms)		lower	upper		(ms)		lower	upper		(ms)		lower	upper
1	15	98	91.7	89.3	94.2	14	21	97.7	96.3	99.0	15	41	96.9	96.1	97.7
3	25	146	96.4	94.7	98.0	23	43	99.7	99.5	99.9	26	75	99.7	98.4	99.5
5	36	201	97.7	96.0	99.4	33	59	99.8	99.7	100.0	34	101	99.7	99.5	99.9
7	46	281	98.8	97.6	100.0	42	79	99.9	99.8	100.0	44	142	99.8	99.7	100.0
9	55	339	98.9	98.0	99.7	51	123	100.0	99.9	100.0	53	179	100.0	100.0	100.0
11	65	426	99.3	98.5	100.0	–	–	–	–	–	–	–	–	–	–

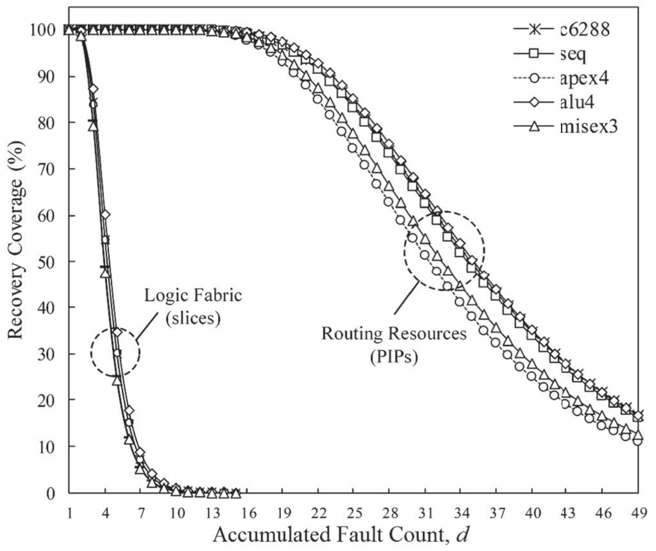


Fig. 10. Fault recovery coverage ($f = 1$, $\delta = 1$).

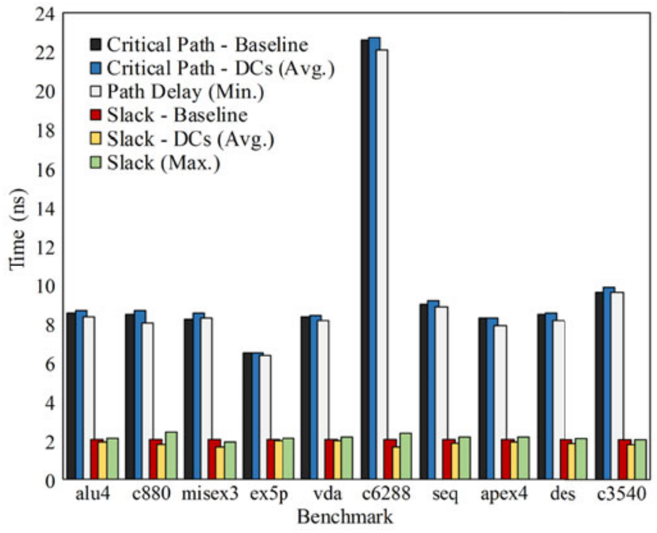


Fig. 11. Effect of design disjunction on system performance.

faster than the baseline design due to the stochastic nature of placement and routing algorithms which does not guarantee convergence to the optimal solution, or due to random variation of the timing of logic resources [42].

Table 4 lists design parameters, execution time to realize the design matrix, error detection method, and size of partial bitstream for each distinct AES module shown in Fig. 9. A failure in any module triggers the embedded processor to execute diagnosis and recovery service routines. Initially, transient and permanent failures are undistinguished. Thus, articulating inputs are re-issued to ascertain if reconfiguration scrubbing can resolve possible SEUs. If discrepancies persist, then DCs of the respective RP are configured to the FPGA through the ICAP. Reconfiguration occurs while using application throughput to stimulate test sequences and maintain availability. The evaluation window for this prototype is set to $1{,}000$ blocks which can be adapted to maintain a desired throughput rate. If the fault detection signal is asserted at any time within the evaluation window, the fault isolation flow will continue by loading a subsequent DC. The feedback from the fault detection logic is captured by the processor where diagnostic data are decoded to identify faulty resources and the optimal resilient DC based on the ranking scheme described in section 5.1.

Figs. 12a and 12b show the outlier behavior for FSR and $CFSR$ ranking metrics, respectively, for 15 test cases. For the sake of comparison, FSR and $CFSR$ values for each test case are normalized from 1 to 10. Each test case is conducted by first selecting an AES module at random and then injecting a SAF at a randomly chosen LUT input. Fig. 12a depicts the top 50 resources in ascending order of FSR for each of the 15 test cases. The defective resources indicated by the red dots rank the highest in FSR with a considerable difference to their next lower ranking resources. The normalized $CFSR$ values for DCs for the 15 test cases depicted in Fig. 12b show that faulty DCs accumulate higher $CFSR$ values. Thus, the DC ranking the lowest $CFSR$ for each test case is selected as the optimal fault-resilient candidate DC for recovery.

Fig. 12c shows the encryption time of the AES core during fault-handling routine for a selected test case. The test procedure is triggered after injecting a SAF at a randomly chosen LUT input in one of the 32-bit s-boxes. At the beginning, DC_{14} is deployed during fault occurrence. The fault recovery procedure reconfigures the device with the partial bitfile of DC_{14} to rule out SEUs. Since discrepancies persist, diagnosis flow continues by testing the remaining 23 DCs. Execution time is given per 100 plaintext blocks. The encryption core throughput is mainly impacted by the partial reconfiguration overhead $t_{pr} = 4.58$ ms and the latency of post-testing decoding phase $t_d = 6.14$ ms. The entire diagnosis flow completes in a millisecond-order time. Fault recovery is achieved after the second test using DC_2 which can be kept in service to maintain availability during time-critical events. The fault diagnosis flow can continue as shown until all DCs are evaluated so that the locations of damaged resources and DC for recovery are determined. Since design disjunction is realized using $\delta = 3$ for the hardware prototype, the inarticulate tests of DC_{12} and DC_{19} have no impact on the trends given by FSR and $CFSR$. The obtained optimal resilient DC in this test case is DC_6 which is deployed to guarantee sustained recovery.

TABLE 4
Design Parameters for AES Modules

Module	R	T	δ	g	$\tau_{mc}(ms)$	Bitstream Size	Detection Scheme
32-Bit s-boxes	60	119	3	24	41		Parity-based [41]
Mix Columns & Add Round Key	55	111	3	24	39	57.9 KB	
128-bit Rotate/Rcon Logics for Key Expansion	52	102	3	23	32		DWC

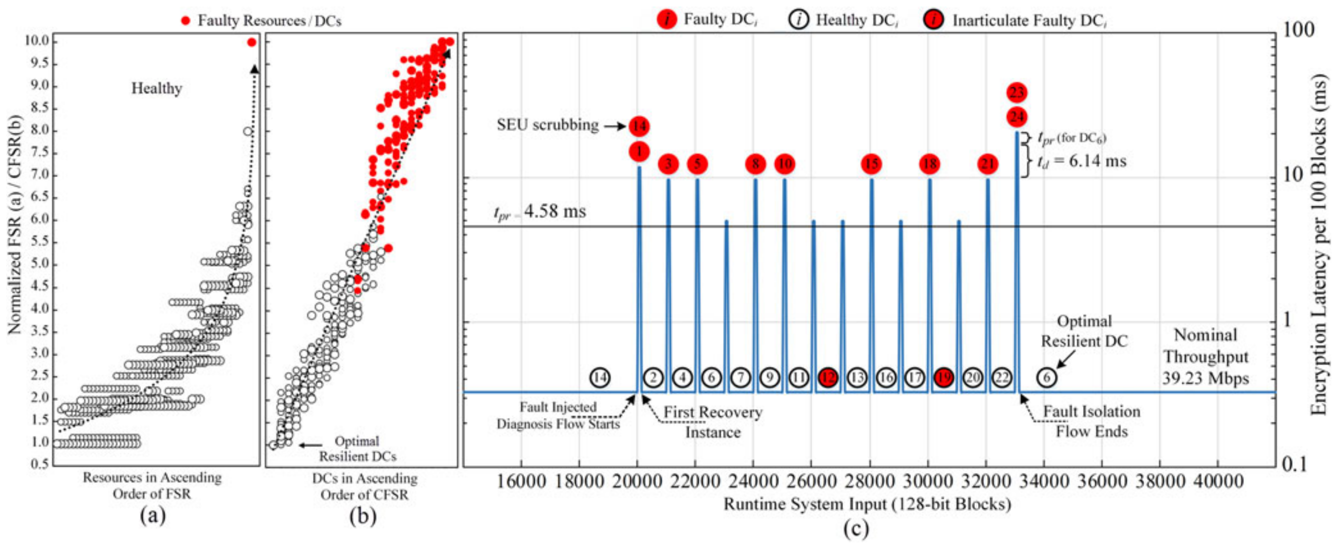


Fig. 12. Execution of isolation phase on an AES module.

7 Comparison of Design Disjunction and Modular Redundancy

Modular redundancy using an NMR method is the most common form of hardware redundancy to tolerate failures. NMR methods can be realized using commercially-available and academic design tools such as Xilinx TMR (XTMR) and BYU-LANL TMR (BL-TMR), respectively. NMR employs N replicas and majority voting which masks failed modules by selecting a majority output. The area and power overheads of this scheme are approximately $(N-1)$-fold including overheads incurred by voting logic. A single failure in a module can render that module unusable which compromises failure recoverability besides pre-determining resource use. *Failure recoverability*, denoted by FR, is defined as the cumulative sum of recovery coverage for all possible combinations of fault locations. This definition can be expressed for a given fault count d as:

$$FR = \sum_{d=1}^{T} RC(d). \tag{9}$$

Let A_m be the minimum resource count required to implement a single module and m_f be the number of failed modules, then recovery coverage for NMR scheme denoted by RC_{NMR} is computed as follows:

$$RC_{NMR}(d) = \frac{|\{x \in \Gamma_d^T \; s.t. \; m_f \leqslant \lfloor \frac{N-1}{2} \rfloor\}|}{|\Gamma_d^T|}. \tag{10}$$

For NMR systems where $N = 3$ and $N = 5$, RC_{NMR} can be given as $3 \cdot |\Gamma_d^{A_m}|/|\Gamma_d^T|$ and $[10 \cdot |\Gamma_d^{2A_m}| - 15 \cdot |\Gamma_d^{A_m}|]/|\Gamma_d^T|$, respectively. Fig. 13a compares the FR of the proposed work with that of NMR. The area overhead of design disjunction in this comparison includes the overhead of CED based on DWC. Both redundancy methods achieve a linear increase in failure recoverability as more redundant resources are added; however, design disjunction offers a higher linear increase. Designing for a higher disjunction factor δ increases g which proportionately results in a higher RC as given by Eq. (7) and thus improves FR.

As depicted in Fig. 13a, due to the provision of fine-grained resource allocation and relocation by design disjunction, a higher FR compared to NMR schemes can be obtained for the same area overhead. For instance, with a similar area overhead to TMR, design disjunction achieves 83.6 percent (143.3 percent) increase in FR over TMR for $\delta = 1$ ($\delta = 7$). Similarly, design disjunction can provide a comparable FR to that of TMR using a considerably lower area overhead. Fig. 13b reflects the area efficiency of the proposed work compared to modular redundancy. Area efficiency is quantified by the ratio of FR to the total resource count T. Similar to modular redundancy methods, a diminishing return on FR occurs as more hardware resources are considered. The resultant area advantage from using design disjunction is more prominent for larger area overhead. For the lowest design setting, i.e., $f = 1$ and $\delta = 1$, design disjunction still enables a higher FR per area than any NMR setup included in this analysis. It is also worth noting that

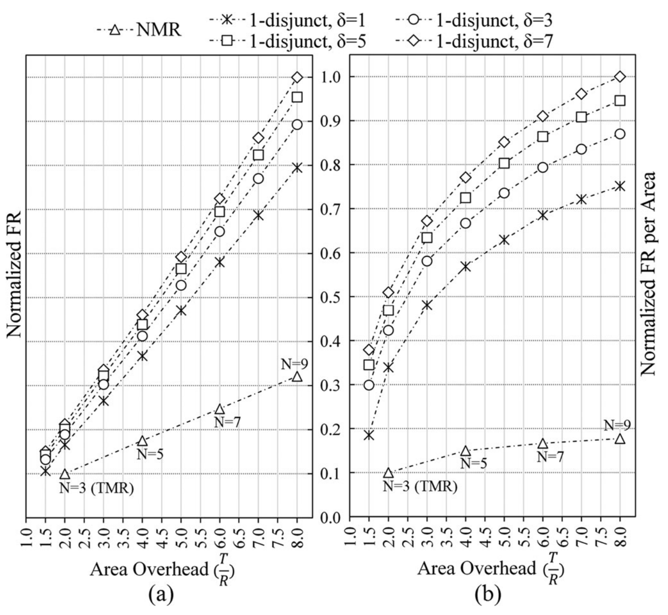


Fig. 13. Area efficiency of design disjunction.

the area advantage of design disjunction can be further enhanced by using parity-based error detection instead of DWC.

The proposed approach can be applied at the reconfigurable logic block level with a broadened range of design parameters to meet area and power constraints while maintaining both adequate fault isolation and recovery. The area overhead imposed by design disjunction is roughly limited to T/R, where R includes the resources required to deploy a CED scheme. Other components such as the embedded processor and memory controller are often present in embedded reconfigurable systems, and thus do not incur an additional area cost. The reliability of these components falls within the scope of embedded system reliability and can be protected by appropriate techniques [43]. The reconfiguration structure is not limited to ICAP. For instance, Xilinx has recently introduced processor configuration access port (PCAP) interface [44] for ARM-based systems to write configuration bits. Design disjunction is realized without loss of generality by the regularity and reconfigurability features of the FPGA device used. Since these features are ubiquitous in contemporary reconfigurable devices, the proposed approach can be highly compatible with many FPGA families from different vendors and other classes of reconfigurable ICs, such as complex programmable logic devices (CPLDs).

8 Conclusion

Design disjunction offers a mathematically-rooted, parameterized, multi-fault isolation and recovery technique for reconfigurable hardware fabrics. Combinatorial construction methods for disjunction and failure ranking schemes for fault diagnosis are developed using operational testing techniques. Experimental results for a set of benchmarks on a Xilinx 7-series FPGA have demonstrated f-diagnosability at the individual slice level with a minimum average isolation accuracy of 96.4 percent (94.4 percent) for $f = 1$ ($f = 2$). An algebraic-based extension was also developed to tolerate inarticulate tests and increase isolation accuracy to any level deemed adequate for successful recovery and repair. Based on these favorable properties and low costs, design disjunction is worthy of consideration for autonomous resiliency in reconfigurable systems demanding high availability.

Acknowledgments

This research was funded by the Ministry of Education of Saudi Arabia under scholarship grant no. 64923.

References

[1] E. Marcus and H. Stern, *Blueprints for High Availability*. New York, NY, USA: Wiley, 2003.

[2] J. Henkel, L. Bauer, J. Becker, O. Bringmann, U. Brinkschulte, S. Chakraborty, M. Engel, R. Ernst, H. Hartig, L. Hedrich, et al., “Design and architectures for dependable embedded systems,” in *Proc. IEEE 9th Int. Conf. Hardware/Softw. Codes. Syst. Synthesis*, Taipei, Taiwan, Oct. 2011, pp. 69–78.

[3] P. S. Ostler, M. P. Caffrey, D. S. Gibelyou, P. S. Graham, K. S. Morgan, B. H. Pratt, H. M. Quinn, and M. J. Wirthlin, “SRAM FPGA reliability analysis for harsh radiation environments,” *IEEE Trans. Nucl. Sci.*, vol. 56, no. 6, pp. 3519–3526, Dec. 2009.

[4] C. Constantinescu, “Trends and challenges in VLSI circuit reliability,” *IEEE Micro*, vol. 23, no. 4, pp. 14–19, Jul./Aug. 2003.

[5] C. Bolchini, A. Miele, and C. Sandionigi, “A novel design methodology for implementing reliability-aware systems on SRAM-based FPGAs,” *IEEE Trans. Comput.*, vol. 60, no. 12, pp. 1744–1758, Dec. 2011.

[6] C. Carmichael, “Triple module redundancy design techniques for virtex FPGAs,” Xilinx, San Jose, CA, USA, Application Note XAPP197(v1.0.1), Jul. 2001.

[7] C. Carmichael and C. W. Tseng, “Correcting single-event upsets in Virtex-4 FPGA configuration memory,” Xilinx, San Jose, CA, USA, Application Note XAPP1088(v1.0), Oct. 2009.

[8] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb, “FPGA partial reconfiguration via configuration scrubbing,” in *Proc. IEEE Int. Conf. Field Programmable Logic Appl.*, Prague, Czech Republic, Aug./Sep. 2009, pp. 99–104.

[9] S. Mitra, W.-J. Huang, N. R. Saxena, S.-Y. Yu, and E. J. McCluskey, “Reconfigurable architecture for autonomous self-repair,” *IEEE Des. Test. Comput.*, vol. 21, no. 3, pp. 228–240, Jun. 2004.

[10] Xilinx, “Vivado design suite,” White Paper WP416 (v1.1), Jun. 2012.

[11] W. Zha, “Facilitating FPGA reconfiguration through low-level manipulation,” Ph.D. dissertation, Virginia Polytechnic Inst. State Univ., Blacksburg, VA, USA, Feb.. 2014.

[12] C. Bolchini and A. Miele, “Design space exploration for the design of reliable SRAM-based FPGA systems,” in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Boston, MA, USA, Oct. 2008, pp. 332–340.

[13] S. Chakraverty, A. Agarwal, A. Agarwal, A. Kumar, and A. Sikri, “Design space exploration for high availability drFPGA based embedded systems,” in *Proc. 1st Int. Conf. Adv. Mach. Learn. Technol. Appl.*, Cairo, Egypt, Dec. 2012, pp. 234–243.

[14] M. Abramovici, C. Strond, C. Hamilton, S. Wijesuriya, and V. Verma, “Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications,” in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, USA, Sep. 1999, pp. 973–982.

[15] X. Iturbe, K. Benkrid, C. Hong, A. Ebrahim, R. Torrego, I. Martinez, T. Arslan, and J. Perez, “R3TOS: A novel reliable reconfigurable real-time operating system for highly adaptive, efficient, and dependable computing on FPGAs,” *IEEE Trans. Comput.*, vol. 62, no. 8, pp. 1542–1556, Aug. 2013.

[16] H. Zhang, L. Bauer, M. A. Kochte, E. Schneider, C. Braun, M. E. Imhof, H.-J. Wunderlich, and J. Henkel, “Module diversification: Fault tolerance and aging mitigation for runtime reconfigurable architectures,” in *Proc. IEEE Int. Test Conf.*, Anaheim, CA, USA, Sep. 2013, pp. 1–10.

[17] V. Hahanov, S. Galagan, V. Olchovoy, and A. Priymak, “Algebralogical repair method for FPGA logic blocks,” in *Proc. IEEE East-West Des. Test Symp.*, St. Petersburg, Russia, Sep. 2010, pp. 482–487.

[18] C. A. Sharma, A. Sarvi, A. Alzahrani, and R. F. DeMara, “Self-healing reconfigurable logic using autonomous group testing,” *Microprocess. Microsyst.*, vol. 37, no. 2, pp. 174–184, Mar. 2013.

[19] K. Zhang, R. F. DeMara, and C. A. Sharma, “Consensus-based evaluation for fault isolation and on-line evolutionary regeneration,” in *Proc. 6th Int. Conf. Evolvable Syst.: From Biol. Hardware*, 2005, pp. 12–24.

[20] M. B. Tahoori, “High resolution application specific fault diagnosis of FPGAs,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 10, pp. 1775–1786, Oct. 2011.

[21] A. J. Van De Goor, “Using march tests to test SRAMs,” *IEEE Des. Test. Comput.*, vol. 10, no. 1, pp. 8–14, Mar. 1993.

[22] L. Bauer, C. Braun, M. Imhof, M. Kochte, E. Schneider, H. Zhang, J. Henkel, and H.-J. Wunderlich, “Test strategies for reliable runtime reconfigurable architectures,” *IEEE Trans. Comput.*, vol. 62, no. 8, pp. 1494–1507, Aug. 2013.

[23] M. Renovell, P. Faure, J. M. Portal, J. Figueras, and Y. Zorian, “IS-FPGA: A new symmetric FPGA architecture with implicit scan,” in *Proc. IEEE Int. Test Conf.*, Baltimore, MD, USA, Oct./Nov. 2001, pp. 924–931.

[24] S. Mitra and E. McCluskey, “Which concurrent error detection scheme to choose?” in *Proc. IEEE Int. Test Conf.*, Atlantic City, NJ, USA, Oct. 2000, pp. 985–994.

[25] C. Bolchini, A. Miele, and C. Sandionigi, “Autonomous fault-tolerant systems onto SRAM-based FPGA platforms,” *J. Electron. Test.*, vol. 29, no. 6, pp. 779–793, Nov. 2013.

[26] A. Doumar and H. Ito, “Detecting, diagnosing, and tolerating faults in SRAM-based field programmable gate arrays: A survey,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 11, no. 3, pp. 386–405, Jun. 2003.

[27] D. Keymeulen, R. Zebulum, Y. Jin, and A. Stoica, "Fault-tolerant evolvable hardware using field-programmable transistor arrays," *IEEE Trans. Rel.*, vol. 49, no. 3, pp. 305–316, Sep. 2000.
[28] E. A. Stott, N. P. Sedcole, and P. Y. K. Cheung, "Fault tolerance and reliability in field-programmable gate arrays," *IET Comput. Digital Techn.*, vol. 4, no. 3, pp. 196–210, May 2010.
[29] M. G. Parris, C. A. Sharma, and R. F. DeMara, "Progress in autonomous fault recovery of field-programmable gate arrays," *ACM Comput. Surveys*, vol. 43, no. 4, p. 31, Oct. 2011.
[30] A. Seffrin and A. Biedermann, "Cellular-array implementations of bio-inspired self-healing systems: State of the art and future perspectives," in *Design Methodologies for Secure Embedded Systems*. Berlin, Germany: Springer, 2011, vol. 78, pp. 151–170.
[31] R. Dorfman, "The detection of defective members of large populations," *Ann. Math. Statist.*, vol. 14, no. 4, pp. 436–440, Dec. 1943.
[32] A. B. Kahng and S. Reda, "New and improved BIST diagnosis methods from combinatorial group testing theory," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 3, pp. 533–543, Mar. 2006.
[33] J. Ghosh-Dastidar, and N. A. Touba, "A rapid and scalable diagnosis scheme for BIST environments with a large number of scan chains," in *Proc. IEEE 18th VLSI Test Symp.*, Montreal, PQ, Canada, Apr./May 2000, pp. 79–85.
[34] M. Cheraghchi, "Coding-theoretic methods for sparse recovery," in *Proc. IEEE 49th Annu. Allerton Conf. Commun., Control Comput.*, Monticello, IL, USA, Sep. 2011, pp. 909–916.
[35] A. J. Macula, "A simple construction of d-disjunct matrices with certain constant weights," *Discr. Math.*, vol. 162, nos. 1–3, pp. 311–312, Dec. 1996.
[36] C. L. Chan, S. Jaggi, V. Saligrama, and S. Agnihotri, "Non-adaptive group testing: Explicit bounds and novel algorithms," in *Proc. IEEE Int. Symp. Inform. Theory*, Jul. 2012, pp. 1837–1841.
[37] M. Cheraghchi, A. Hormati, A. Karbasi, and M. Vetterli, "Group testing with probabilistic tests: Theory, design and application," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 7057–7067, Oct. 2011.
[38] E. Knill, W. J. Bruno, and D. C. Torney, "Non-adaptive group testing in the presence of errors," *Discr. Appl. Math.*, vol. 88, no. 1, pp. 261–290, Nov. 1998.
[39] T. Kumar and F. Lombardi, "A novel heuristic method for application-dependent testing of a SRAM-based FPGA interconnect," *IEEE Trans. Comput.*, vol. 62, no. 1, pp. 163–172, Jan. 2013.
[40] A. Alzahrani and R. F. DeMara, "Hypergraph-cover diversity for maximally-resilient reconfigurable systems," in *Proc. IEEE 12th Int. Conf. Embedded Softw. Syst.*, New York, NY, USA, Aug 2015, pp. 1086–1092.
[41] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the advanced encryption standard," *IEEE Trans. Comput.*, vol. 59, no. 5, pp. 608–622, May 2010.
[42] A. Alzahrani and R. F. DeMara, "Process variation immunity of alternative 16nm HK/MG-based FPGA logic blocks," in *Proc. IEEE 58th Int. Midwest Symp. Circuits Syst.*, Fort Collins, CO, USA, Aug 2015, pp. 1–4.
[43] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 2nd ed. Berlin, Germany: Springer, Apr. 2011.
[44] C. Kohn, "Partial reconfiguration of a hardware accelerator on Zynq-7000 all programmable SoC devices," Xilinx, San Jose, CA, USA, Application Note XAPP1088(v1.0), Jan. 2013.

Ahmad Alzahrani received the BS degree in electrical engineering from the Umm Al-Qura University, in 2002. He received the MS degree in computer engineering from the University of Arkansas, Fayetteville, in 2009, and the PhD degree in computer engineering from the University of Central Florida, in 2015. His research interests include computer architecture, fault tolerance, and adaptive reconfigurable computing. He is a member of the IEEE.

Ronald F. DeMara received the PhD degree in computer engineering from the University of Southern California, in 1992. Since 1993, he has been a full-time faculty member at the University of Central Florida where he is a professor and Computer Engineering program coordinator. His research interests are in computer architecture with emphasis on Evolvable and Resilient Hardware, on which he has published approximately 175 articles. He has served on the Editorial Boards of *IEEE Transactions on VLSI Systems*, *ACM Transactions on Embedded Systems*, *Journal of Circuits, Systems, and Computers*, the journal *Microprocessors and Microsystems*, various conference program committees, and is currently an associate editor of *IEEE Transactions on Computers*. He received the Joseph M. Bidenbach Outstanding Engineering Educator Award from the IEEE, in 2008. He is a senior member of the IEEE

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.