2015

Design Disjunction for Resilient Reconfigurable Hardware

Ahmad Alzahrani
University of Central Florida

DESIGN DISJUNCTION FOR RESILIENT RECONFIGURABLE HARDWARE

by

AHMAD A ALZAHRANI
B.S. Umm Al-Qura University, Saudi Arabia, 2002
M.Sc. University of Arkansas, Fayetteville, 2009

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

Fall Term
2015

Major Professor: Ronald F. DeMara

ABSTRACT

Contemporary reconfigurable hardware devices have the capability to achieve high performance, power efficiency, and adaptability required to meet a wide range of design goals. With scaling challenges facing current complementary metal oxide semiconductor (CMOS), new concepts and methodologies supporting efficient adaptation to handle reliability issues are becoming increasingly prominent. Reconfigurable hardware and their ability to realize self-organization features are expected to play a key role in designing future dependable hardware architectures. However, the exponential increase in density and complexity of current commercial *SRAM-based field-programmable gate arrays (FPGAs)* has escalated the overhead associated with dynamic runtime design adaptation. Traditionally, static modular redundancy techniques are considered to surmount this limitation; however, they can incur substantial overheads in both area and power requirements. To achieve a better trade-off among performance, area, power, and reliability, this research presents design-time approaches that enable fine selection of redundancy level based on target reliability goals and autonomous adaptation to runtime demands. To achieve this goal, three studies were conducted:

First, a graph and set theoretic approach, named Hypergraph-Cover Diversity (HCD), is introduced as a preemptive design technique to shift the dominant costs of resiliency to design-time. In particular, union-free hypergraphs are exploited to partition the reconfigurable resources pool into highly separable subsets of resources, each of which can be utilized by the same synthesized application netlist. The diverse implementations provide reconfiguration-based resilience throughout the system lifetime while avoiding the significant overheads associated with runtime placement and routing phases. Evaluation on a Motion-JPEG image compression core using a Xilinx 7-series-based FPGA hardware platform has demonstrated the potential of the proposed FT method to achieve 37.5% area saving and

up to 66% reduction in power consumption compared to the frequently-used TMR scheme while providing superior fault tolerance.

Second, Design Disjunction based on non-adaptive group testing is developed to realize a low-overhead fault tolerant system capable of handling self-testing and self-recovery using runtime partial reconfiguration. Reconfiguration is guided by resource grouping procedures which employ non-linear measurements given by the constructive property of f-disjunctness to extend runtime resilience to a large fault space and realize a favorable range of tradeoffs. Disjunct designs are created using the mosaic convergence algorithm developed such that at least one configuration in the library evades any occurrence of up to d resource faults, where d is lower-bounded by f. Experimental results for a set of MCNC and ISCAS benchmarks have demonstrated f-diagnosability at the individual slice level with average isolation resolution of 96.4% (94.4%) for f=1 (f=2) while incurring an average critical path delay impact of only 1.49% and area cost roughly comparable to conventional 2-MR approaches.

Finally, the proposed Design Disjunction method is evaluated as a design-time method to improve timing yield in the presence of large random within-die (WID) process variations for application with a moderately high production capacity. Results for a set of benchmarks show an average gain in timing yield of up to 39.42%, 36.91%, and 57.45% for total variations of 25%, 15%, and 5%, respectively. The enhanced timing yield is attained while achieving reductions in mean delay of 9.96% 6.85%, and 3.58% for the same variability levels.

Dedicated to my family for their support throughout my educational life.

ACKNOWLEDGMENTS

I'm genuinely grateful for the financial support from the Ministry of Education of Saudi Arabia. I also wish to acknowledge with sincere thanks my academic advisor, Prof. Ronald F. DeMara, for his time, technical feedback, and for providing all needed resources to accomplish this work. Finally, I would like to extend gratitude to my doctoral committee for their helpful comments and suggestions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The increased demand for reliable and high performance computing for many applications including data-centers, medical devices, military aerospace, automobiles, power generation, electric rail systems, smart grids, and industrial manufacturing have fueled the growth of FPGA-based systems. In 2013, the global market for FPGAs was valued at $5.45 billion and it is forecasted to reach $9.88 billion by 2020 according to Grand View Research, Inc. The requirement for more reliable FPGA-based systems is further driven by the promotion of safety regulations in electric and hybrid electric automobiles and increased high availability requirements of virtualization, social, cloud, and mobile technologies. Furthermore, reliable FPGA-based systems are considered crucial elements for the success of upcoming space exploration missions. This chapter discuses motivation for techniques to addresses reliability challenges for reconfigurable hardware and highlights the contributions of this work.

1.1 Reliability Challenges of Reconfigurable Systems

Continued scaling of transistor feature size has exacerbated reliability issues, e.g. aging effects, latent faults, and temporary failures in integrated circuits (ICs). Consequently, the need for effective *fault tolerance (FT)* techniques has received increasing interest over the last decade. A well recognized approach for designing effective reliable systems leverages reconfigurable hardware such as FPGAs, which can provide exceptional computational capability at a high level of performance per area and power [1]. Hardware-adaptive devices and their ability to realize self-organization features are expected to become a key role in designing future dependable hardware architectures [2]. Additionally, autonomous FT operations realized by reconfigurable hardware have become attractive design decision for deployed systems

in remote and harsh environments where routine repair and service are either impossible or prohibitively expensive. At present, the most widely adopted reconfigurable architectures are SRAM-based FPGA devices whose capacity can exceed a million logic cells that can be leveraged to enable new FT techniques. SRAM-based FPGAs have become ubiquitous in application-specific embedded systems, high performance computing centers as well as safety-impacting, mission-critical, and high availability commerce-enabling systems. The FPGA devices themselves can occupy a significant role in overall reliability of FPGA-based systems [3]. Fortunately, the runtime partial reconfiguration capabilities of contemporary FPGAs can be utilized to maintain degraded-mode operation while enabling rapid recovery from a wide range of failure modes.

Over the last two decades, a considerable amount of research has been conducted on realizing FPGA-based systems that are robust to permanent and transient failures. Device-level and manufacturing techniques exist to increase FPGA lifetime and reliability at the expense of higher cost and dramatic decrease in performance as compared to commercial-off-the-shelf (COTS) components. FT techniques developed for FPGAs are often based on fault-masking using replication with majority voting. Alternatively, dynamic remapping of a single design implementation at the functional-module level or logic-tile level is prevalent to relinquish permanently damaged resources [4]. Commonly-deployed schemes based on voting analysis such as N-modular redundancy (NMR) can incur N-fold power and area overhead to tolerate temporary and permanent failure in up to $\lfloor \frac{N-1}{2} \rfloor$ modules. Effective techniques such as re-execution and reconfiguration *scrubbing* [5] [6] have also been adopted along with fault-masking schemes to provide low-cost recovery form temporary failures. On the other hand, techniques based on remapping of resources offer the potential to increase fault coverage at the expense of execution-time complexity. Some runtime remapping processes may entail on-board execution of the FPGA design flow including placement and routing which are

time-consuming tasks even for high performance fabric-embedded cores [7]. Thus, conventional dynamic remapping techniques typically require the system be taken offline for an undesirable interval of time. For time-sensitive and mission-critical applications, availability and mitigating downtime can be a crucial requirement for assuring system dependability.

1.2 Importance of Runtime Reconfiguration

With the rise of *reconfigurable hardware* (RH) over the last two decades, in-field reconfigurability has opened up new possibilities to incorporate pseudo-intelligent FT attributes such as self-repair and autonomous fault recovery [8]. Such attributes are key enablers for efficient and sustainable fault-tolerant systems. RH is expected to have an essential role in designing future dependable embedded systems [2]. Unfortunately, exploiting design flexibility of modern RH for runtime FT is encumbered by the heuristic nature and increasing complexity of design placement and routing mechanisms. FPGAs being the prominent example of RH can exemplify this challenge. Execution of a design flow targeting an SRAM-based FPGA can take an order of minutes to hours using a high-end multi-processing machine [9]. For low-performance fabric-embedded cores, the computational and energy constraints to execute in-field design reroute can be prohibitive [7]. We emphasize our observation here based on the current state of computer-aided design (CAD) tools used with available commercial off-the-shelf (COTS) reconfigurable components due to the increasing trend in using COTS-based embedded systems [10] [11]. In light of this major concern, design-time FT strategies that minimize reliance on runtime execution of design flow are sought which can be readily integrated with existing vendor tools. More specifically, the dominant implementation cost of reconfigurability feature can be mitigated by preparing an optimal set of design alternatives at design phase that properly cover the solution space for reliability exposures at runtime.

1.3 Challenges of Online Diagnosis of Reconfigurable Hardware

Test is the enabling technology by which fault tolerance can be ensured. The ability to obtain information about faulty resources is a critical factor to realize efficient self-repair. It facilitates fault evasion whereby faulty resources are avoided, or whereby partially-damaged resources can be reassigned other useful functionalities. online fault isolation and recovery approaches for FPGA logic using dynamic reconfiguration have relied on built-in self-test (BIST) [12] [13]. However, dedicated BIST structures including test pattern generators (TPGs) and output response analyzers (ORAs) are typically not available for FPGA platforms [13]. Modern FPGA architectures are also not fully scan-ready. Thus, scan chains, TGPs, and ORAs are frequently implemented in the fabric directly using look-up tables (LUTs) and shift registers. As a consequence, BIST-inspired methods can incur up to 50% increase in FPGA resources [14]. Alternatives that eliminate BIST area and power overheads, referred to operational testing techniques, conduct functional tests via input data that is simultaneously used for normal throughput [15]. These techniques attain availability by relying on runtime inputs, computational redundancy, and output comparison to assess the subset of resources currently used by an application.

1.4 Mitigating Process Variation Impact on Yield

There is a growing concern as design complexity and process variation continue to grow. The impact of process variation can affect both functional and timing yield of a final design. Functional yield describes the proportion of dies that achieve functional correctness, whereas timing yield necessitates that final designs adhere to specific timing requirements. For FPGAs, target applications at the time of manufacturing are likely unknown due to

the design flexibility nature of these devices. This adds further complexity to performance prediction and analysis to find the best design and architectural solutions to improve yield. Fortunately, target applications can take advantage of the reconfigurability of FPGAs by assigning the application functional blocks to faster resources and region inside a device.

1.5 Contributions of the Dissertation

The main contributions of the dissertation are listed below:

Probabilistic Group Testing Technique for Fault Isolation in Reconfigurable Logics: A probabilistic group testing-based method to isolate multiple faults in reconfigurable logic is presented. The proposed technique is aimed to locate faulty elements using a low number of configurations even under the impact of the low coverage problem of functional testing. The performance of the method was demonstrated using an 8x8 tree multiplier and data encryption standard (DES) implemented on Xilinx Virtex-2 pro and Virtex-4 FPGAs. The results have shown that the group testing techniques can be used to carry out fault isolation phase with a clear outlier behavior. This work is highlighted in Chapter 3.

Fast Online Diagnosis and Recovery of Reconfigurable Logic Fabrics using Design Disjunction: This research proposes novel *Design Space Exploration (DSE)* [16] approachES to realize a FT FPGA system. It provides a low-cost *fault localization / fault isolation* capability along with rapid *fault recovery* from temporary and permanent faults while incurring minimal perturbation to normal system operation. In particular, a small library of alternative *design configurations (DCs)* with f*-disjunct* resource usage are created during dealing time. The disjunctness of these alternatives enables fault localization using non-adaptive sparse recovery technique. Furthermore, fault resilience against large fault sce-

narios is achieved by reusing a subset of disjunct designs that ensure continual execution with minimal recovery time. The contributions of this work are summarized as follows:

1. The first attempt to utilize non-adaptive group testing techniques to provide runtime diagnostic analysis for reconfigurable hardware.
2. Extension of non-adaptive group testing to cases where false test outcomes can disrupt diagnosis resolution, and
3. An explicit approach to design the optimal number of DCs at design time for recovery from multiple logic and interconnect faults during the system lifetime.

To realize and validate the effectiveness of the proposed approach, the work is divided into two phases. The first phase addresses the need for dynamic creation of DCs on existing reconfigurable architectures and CAD tools with minimal design modifications. The second phase involves developing the simulation environment and fault injection mechanism to experimentally validate the work. The detail of this research is discussed in Chapter 4.

Hypergraph-Cover Diversity for Maximally-Resilient Reconfigurable Systems: In this work, a design-time FT strategy that minimize reliance on runtime execution of design flow are sought which can be readily integrated with existing vendor tools. More specifically, the dominant implementation cost of reconfigurability feature can be mitigated by preparing an optimal set of design alternatives at design phase that properly cover the solution space for reliability exposures at runtime. In this work, *hypergraph-cover diversity* (HCD) approach based on *graph and set theory* is proposed to attain this objective for the FT coverage problem on embedded reconfigurable fabric. The HCD method exploits the strong notion of *separability* [17] obtained by *union-free hypergraphs* [18] to model resource allocation

among distinct design alternatives for highly diverse and fault resilient designs. This work is presented in Chapter 5.

Process Variation Immunity of Alternative 16nm HK/MG-based FPGA Logic Blocks: Continued miniaturization of semiconductor technology to nanoscale dimensions has elevated reliability challenges of high density Field-Programmable Gate Arrays (FPGA) devices due to increasing impacts of Process Variation (PV). The issue is addressed in this work using a systematic bottom-up analysis by determining the relative influence of PV on alternate design realizations of FPGA logic blocks. Results for conventional design structures are obtained through detailed SPICE simulations and related to structural risk features. Namely, Transmission Gate (TG) and Pass Transistor (PT) based MUX architectures for realizing Look-Up-Tables (LUTs) are compared. PT-based designs that meet the 95% yield objective can exhibit as high delay variation which could lead to a considerable failure rate. PV impact can be reduced if TG-based LUT is considered as alternative. The impact of transistor sizing is also investigated as a method of mitigating PV susceptibility in FPGA structures. This detail and results analysis of this work is presented in chapter 6.

Mitigating the Impact of Process Variations via Disjunct Resource Utilization: The high reconfigurability nature of FPGAs which leads to effective solutions in the domain of fault tolerance can also be exploited for mitigating the challenge of process variation. However the cost of exercising reconfigurability feature is high if applied for each affected device. Thus, a universal set of designs that can be used for a large quantity is sought in this work through design-time approaches that can be integrated with current commercial toolchain. The resource diversity achieved by design disjunction allows higher likelihood that a design may circumvent resources affected by variability. In chapter 7, design disjunction is investigated for this objective with results that show effectiveness of the proposed design method to lower the impact of variability.

CHAPTER 2: BACKGROUND AND RELATED RESEARCH

FPGAs can be classified into many categories according to different attributes and criteria. The most common attribute is the memory technology used to store design configurations. SRAM-Based FPGAs are the most successful type due to the high density of SRAM manufacturing process. Other types found can be based on Flash, EEPROM or anti-fuse links, each of which has its advantages and target applications.

A great deal of research has been devoted on the ability of reconfigurable logics to build autonomous fault-tolerant systems. Reconfigurable hardware carry the advantage of reducing system complexity through offering high degree of flexibility that can ease designers job by focusing on high level system goals. FPGAs uses configurable logic block (CLB) as flexible logic resources to implement large diverse applications. They offer this high diversity without exceeding high power and space envelope. The configurability of these devices makes them a viable solution to build flexible systems that meet the requirements of fault-tolerant systems.

The primary objective of this work is to utilize reconfigurability of SRAM-Based FPGAs at fine resolution to ensure correct functionality over a given system lifetime while rapidly and efficiently mitigating occurred failures. For SRAM-based FPGAs, the primary short-term reliability challenge is single-event effects (SEUs). Fortunately, this challenge can be addressed through runtime reconfiguration capability of modern devices. Long-term reliability concerns include permanent damage in one or more elements of the device. The cause of permanent failures can be unpredicted which presents a challenge to system designers to find the optimal design decision for balancing area and power overheads and complexity of recovery execution time.

2.1 Common Failures in SRAM-based FPGAs

Failures in SRAM-based FPGAs can be classified into permanent and transient failures. Permanent failures include any irreversible damage to the physical resources, whereas transient failures are short-term events induced by external sources such as power supply noise, electromagnetic interference, or charged particles [19]. One primary source of failures in SRAM-Based FPGAs is single event effects (SEEs) which occur when high-energy particles, such as heavy ions and protons, collide with silicon atoms in the device transistors. The energy produced by this collision can be high enough to cause single-event-upsets (SEUs) which invert the logical value retained in a memory cell [20]. Particle-induced faults, also called soft errors, alter SRAM configuration bits and subsequently lead to a continual functional failure. The affected memory cell has to be rewritten with the correct logic value to maintain functional correctness. With the high SRAM density of modern FPGAs, the probability of SEUs occurrence becomes a major concern, particularly in a high radiation dose environment such as orbital vehicles and nuclear facilities. Another type of SEEs is single-event latchups (SELs) which arise when particle collision with silicon device forms a forward-biased diode structure in the device substrate causing high destructive current that could permanently damage the device. Existing manufacturing process techniques exist to mitigate this type. Another common source of permanent failures is attributed to wear-out, or aging, mechanisms such as negative bias temperature instability (NBTI), time-dependent dielectric breakdown (TDDB), and hot carrier injection (HCI). An effective technique to mitigate aging is achieved through wear-level mechanism where extra area overhead is introduced and the stress caused by switching activity on logics is distributed evenly among all available resources. The increase in lifetime due to the balanced stress distribution is proportional to the area overhead.

2.2 Partial Reconfiguration

Modern SRAM-based FPGAs enable partial reconfiguration (PR) which allows the configuration data of a region in an FPGA to be modified without disturbance to the functionality implemented in other regions of the device. This feature facilitates runtime adaptation to different system requirements. During design time, system designers identify partial reconfigurable regions (PRRs) along with the associated partial reconfigurable modules (PRMs) and use vendor CAD tools to create partial configuration bitstreams for the PRMs. The sizes of partial bitstreams can be considerably lower than the device full bitstream depending on the size of the PRRs. This leads to reduced storage requirement and substantial decrease in configuration time. During runtime, partial bitstream can be loaded to the device through either external configuration controller, e.g. microcontroller, or internal configuration port, e.g. Xilinx's internal configuration access port (ICAP). Currently, PR is supported by the top two FPGA vendors, Xilinx [21] and Altera [22].

2.3 Fault Tolerance using SRAM-based FPGAs

Fault tolerant system is described as having the property of continuing normal operation in the presence of failures [23]. Implementing fault tolerant systems has become an important part in many applications that require high degree of reliability and availability. Mission critical systems, e.g. those used in space exploration and harsh environments, can be exposed to a wide range of thermal and environmental stress, ultra-violate radiation, and unknown and unpredictable operational conditions not account for during design phase. Consequently, the need for of autonomous fault tolerant platforms to adapt to changing operational conditions. Typically, fault tolerant systems encompass two important strategies: *fault diagnosis* and

fault recovery, Figure 2.1.

Fault recovery is an important factor when developing a fault tolerant system. It can be defined as the ability to recover a system to a reliable operation when a fault has occurred and caused erroneous behavior. Fault recovery techniques normally rely on fault diagnosis to trigger and direct the recovery phase.

Most fault tolerant systems for reconfigurable devices can be divided into either device-level or configuration-level. Device level approaches use hardware spatial redundancy to implement fault tolerance by replacing faulty components with healthy resources or modules. Configuration level approaches take advantage of device reconfigurability to avoid faulty fabrics.

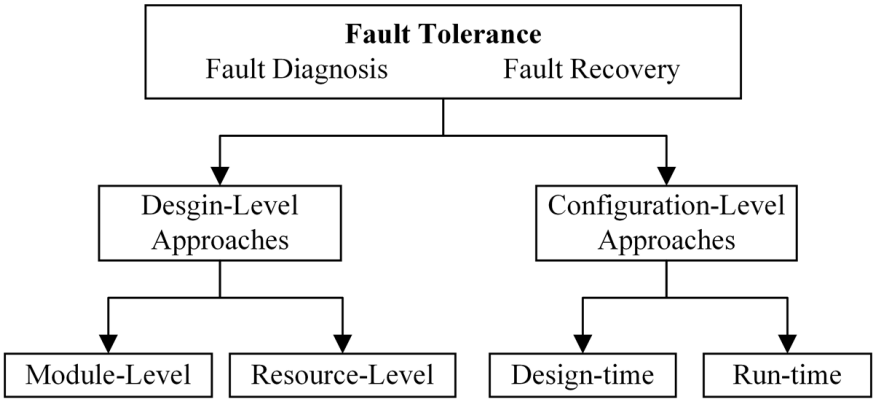


Figure 2.1: FPGA fault tolerance approaches classification.

Fault diagnosis, on the other hand, is the process of detecting faults and finding the cause of system failure. Fault diagnosis often requires two phases: *fault detection* and *fault isolation* or *fault localization* [24] as depicted in Figure 2.2. Fault diagnosis strategy can be either functional testing or resource-oriented testing. Majority of fault diagnosis methods are based on resource-oriented approaches which attempt to detect and isolate faults using

hardware-based mechanism. The common approaches under this category are: built-in-self-test (BIST) as in roving STARs approach [12], iterative logic array (ILA) [25], and array based techniques [26]. The drawbacks of these approaches are the fact that they require exhaustive testing and mostly force device under test (DUT) operation to be taken offline. Furthermore, these methods only provide fixed fault isolation granularity and can incur high detection latency. Functional testing approaches mitigate this problem by checking system functionality through voting elements, e.g. triple modular redundancy (TMR) [27], or output verification circuitry. However due to area overhead imposed by voting system, it can be less attractive to many target applications.

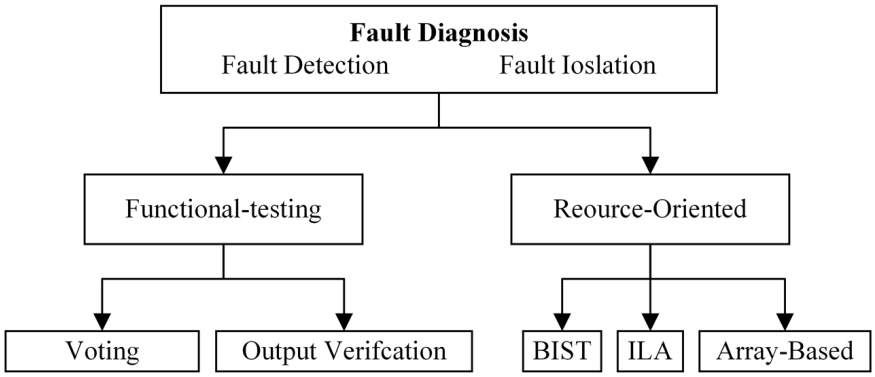


Figure 2.2: FPGA fault diagnosis approaches classification.

2.4 Online fault Diagnosis and Recovery of SRAM-based FPGAs

Fault tolerant reconfigurable systems are often deployed in mission-critical applications with strict high reliability and high-availability requirements that mandate minimal down-time [28]. A critical factor to maintain availability through fault recovery is fault localization overhead. Low-overhead fault isolation facilitates rapid fault evasion whereby faulty resources are bypassed and partially damaged resources can be reassigned useful function-

ality. Online fault localization and recovery often considers the structural heterogeneity of contemporary FPGAs. 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 for decades and the well-known MARCH algorithms [29] have been proven effective for testing and diagnosis of RAM cells in FPGAs. Previous runtime fault isolation and recovery approaches for FPGA logic using dynamic reconfiguration have relied on BIST [12] [13]. The roving STARs scheme [12] partitions the reconfigurable fabric into tiles, and online testing is carried out by continuously roving a BISTer from one tile to another while resources not used by the test task are dynamically reconfigured to maintain online operation. Although failures are resolved at a fine resolution, processing must continuously be suspended to copy state values prior to each tile movement. Resource recycling is facilitated, although fault detection depends on the latency of BISTers to rove the device before encountering faulty cells which, under worst cases, could lead to a large downtime.

Another recent BIST-based fault tolerant FPGA approach is illustrated by the reliable reconfigurable real-time operating system (R3TOS) [30] whereby 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 6 Block RAMs (BRAMs) and 500 configurable logic blocks (CLBs) protected with selective TMR whose resources also undergo periodic testing. The HWuK orchestrates redundant instances of critical circuits using a quarantining process based on BIST and is demonstrated for an inverter controller of a railway motor and CubeSat space-based application on a Xilinx-4 device. The impact of BIST overhead in this approach can be hidden by the use of hardware replication and

voting.

To reduce the high complexity and cost of BIST, application-dependent BIST testing [31] 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 [31] 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 [15]. 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 [32]. DWC that compares the Hamming distance between the outputs of two spatially redundant modules is compatible with recent multi-objective DSE approaches [33] which utilize a cost function that considers area requirements and resource utilization against overhead of reconfiguration time. In [34], 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 [**?**], most functional testing techniques are mainly used for test-

ing 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 [35] 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 run-time, 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 [15] 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 [36] 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 [37], [24], and [38], respectively.

CHAPTER 3: PROBABILISTIC GROUP TESTING TECHNIQUE FOR FAULT ISOLATION IN RECONFIGURABLE LOGICS

If a test is required to find d defectives among T elements, where d is unknown, then a straightforward procedure is to test each element individually. Assuming all tests are reliable, then the time complexity becomes $O(T)$. This cost can be very significant, especially if T is much larger than d. The cost can be substantially reduced by dividing the T elements into smaller g subsets, called groups. The collective results after testing each group are decoded to identify the d defectives. The challenge is to sample the minimal number of groups sufficient to identify defectives. This is the basic idea behind group testing which was first introduced by Dorfman [39] for screening a large numbers of soldiers via blood tests during World War II. Group testing has since been adopted to diverse applications such as testing for manufacturing defects, DNA library screening, coding theory, software testing, and BIST-based diagnosis in digital systems [40] [41]. Based upon how test groups are sampled, most group testing techniques can be classified into *adaptive* and *non-adaptive* categories. In this chapter, the use of adaptive group testing techniques (AGT) for fault isolation in reconfigurable logics is discussed and a novel probabilistic AGT based strategy is presented for multiple fault isolation at the slice level.

3.1 Adaptive Group Testing (AGT)

When using adaptive group testing, complete knowledge of how groups are sampled before the testing process begins is not specified. Groups are constructed iteratively during the

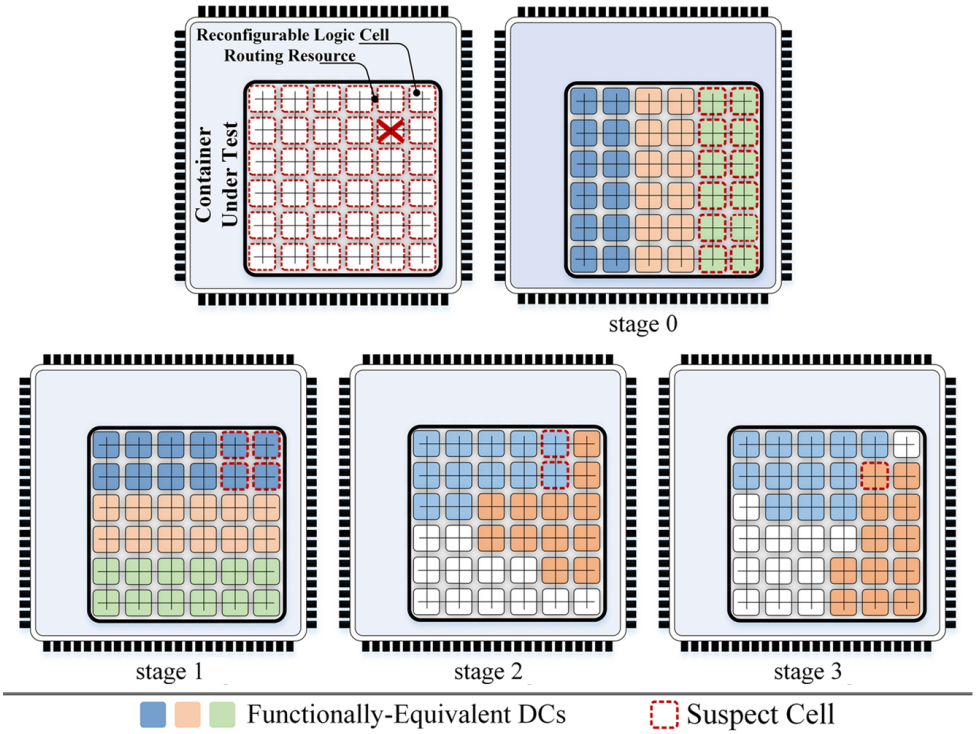


Figure 3.1: Example of adaptive group testing.

testing procedure based on test outcomes. As testing progresses, the iterative sampling of groups narrows down the candidate set of faulty elements until defectives are isolated. Adaptive group testing was first proposed for functional testing for FPGAs in [34]. Figure 3.1 shows a motivating example that illustrates how adaptive group testing isolates a single defective programmable logic cell within an array of 36 cells. The reconfigurable region under test is referred to as a container. Each test group is a set of resources that implements a functionally equivalent design configuration. Distinct groups are signified by color. Initially, all cells in the container are deemed suspect. The test starts by dividing the 36 suspect cells among three configurations. The suspect set is narrowed down to those that are used by the erroneous configuration. The suspect set is iteratively divided among a new set of configurations as manifested by stage 1 through stage 3. The algorithm terminates once the

suspect set contains only a single cell which identifies the defective cell. The complexity of this algorithm is logarithmic in the number of cells, T, and depends on the allowed maximum number of configurations in every test generation.

3.2 Proposed Probabilistic AGT Scheme

Finding faulty slices in FPGA using adaptive group testing can be very challenging due to the low coverage problem of functional testing. Depending on the target application, fault articulation may not be manifested under a large input space. This challenge can hinder any group testing based technique from being an efficient fault isolation method. Fault articulation is essential to the convergence of the algorithm. Even under a single fault assumption, the isolation process can be delayed or impossible to succeed if the fault articulation rate is low. A one way to approach this problem is to incorporate group testing with probabilistic measures. In the proposed scheme herein, two dedicated history matrices are used. A one matrix is employed to track the fault articulation count for each FPGA slice. The other matrix is used to determine what resource regions in the FPGA can be considered for sampling test groups in each stage. Group sampling and testing procedures of the proposed scheme are detailed as follows:

1. Initially suspect resources are divided into eight regions (labeled from r_1 to r_8) with equal sizes. Since the number of faulty slices is unknown at the beginning of the test, and no information are given about faulty elements, all resources are considered suspects. Thus, for a hardware utilization of approximately 1/2, each region should contain less than half of the required resources necessary to implement the target application.

2. For each test stage, eight configurations (labeled from C_1 to C_8) are created such that each configuration covers only two regions as expressed below and shown in Figure. 3.2:

$$C_1 = r_1 \wedge r_2$$
$$C_2 = r_2 \wedge r_3$$
$$C_3 = r_3 \wedge r_4$$
$$C_4 = r_1 \wedge r_4$$
$$C_5 = r_5 \wedge r_6$$
$$C_6 = r_6 \wedge r_7$$
$$C_7 = r_7 \wedge r_8$$
$$C_8 = r_5 \wedge r_8$$

   Each configuration utilizes only suspect resources from its two assigned regions. The region size may decrease to a point where two regions have no enough resources to realize the target application. In that case, a random number of non-suspects are chosen to complete the required resources. The random selection of non-suspects is crucial to overcome the low fault articulation that causes some faulty resources to be exonerated in earlier stages.

3. Configurations are tested against a small set of inputs. In this work, a subset of random inputs generated from a uniform distribution is considered for testing each configuration.

4. For configurations that manifest erroneous outputs, entries of the history and region matrices will be updated for the used resources and their respective regions.

5. The regions that exhibit no fault manifestation will be marked as non-suspect. The regions that exhibit erroneous outputs are grouped together and divided again into

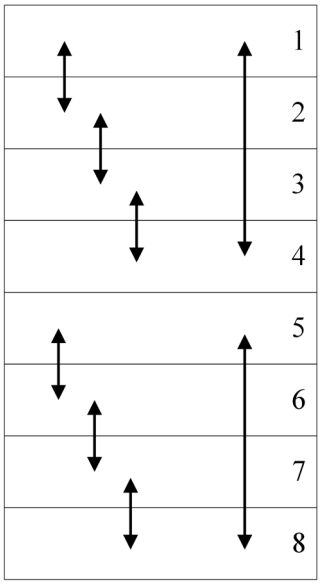


Figure 3.2: Strategy for region selection to create each test configuration.

eight regions. To further increase convergence towards the isolation of faulty elements, regions are ordered according to the articulation rates of their contained resources.

6. For each new test stage, the non-suspect individual that has the highest fault articulation count is introduced back to the suspect pool. Thus, the faulty slices that escape the suspect set will be marked again as suspect by the random selection of non-suspect resources described in step 2. As the algorithm progresses, exonerated faulty slices will accumulate higher articulation rates and hence can be brought back to the suspect set.

7. The region history matrix is reset for each new stage.

8. Steps 2 through 7 are repeated till the region size becomes small enough to observe the wide difference in articulation record of reconfigurable resources.

   In this scheme, resources that are utilized by faulty configurations will survive in the suspect pool; thus, increasing the chance for being grouped with other faulty resources

to manifest more fault articulations. The target number of faulty resources that can be isolated by the proposed scheme depends on the number of regions considered. For the 8-region scheme described herein, the number of faults that can be identified is at most three. By adapting the number of regions, a higher number of faults can be identified.

As the test progresses, it is expected that the region size will decrease. If all regions exhibit at least one faulty output, no region will be excluded in a proceeding test stage, thus new eight configurations are created and tested again. In addition, the number of suspects should also decrease as the region size decreases. The faulty slices can be identified once the region size becomes equal to 1. At this point, faulty slices have much higher accumulated entries in the history matrix than non-faulty elements. Some faulty resources may move to the non-suspect set, but they return to the suspect pool as their matrix entries increase. Since regions with higher articulation rates are forced to be at the top order of the described grouping scheme, faulty slices will drift to the regions having a lower order to increase the chance to exclude more regions as the number of test stages increases.

3.3 Evaluation Setup

A modified version of the fault injection and analysis tool (FIAT) tool [34] is developed to demonstrate the effectiveness of the proposed probabilistic AGT method. Two case studies are conducted in this evaluation. First, an 8x8 tree multiplier is implemented using a Virtex-2 pro FPGA. The required resources for the considered tree multiplier are 99 slices. Second, a data encryption standard (DES) core is implemented using a Virtex-4 FPGA. The DES core requires 322 slices in the target FPGA. The hardware utilization ratio for both case studies is set to 25%. Three stuck-at faults (SATs) are injected at randomly chosen LUT

inputs for each case study. In all experiments, the described AGT procedure is allowed to run till the region size becomes 1 and distinct difference between faulty and non-faulty slices is observed. In this evaluation, test vectors are 60 input patterns generated randomly to avoid any bias towards specific input dataset.

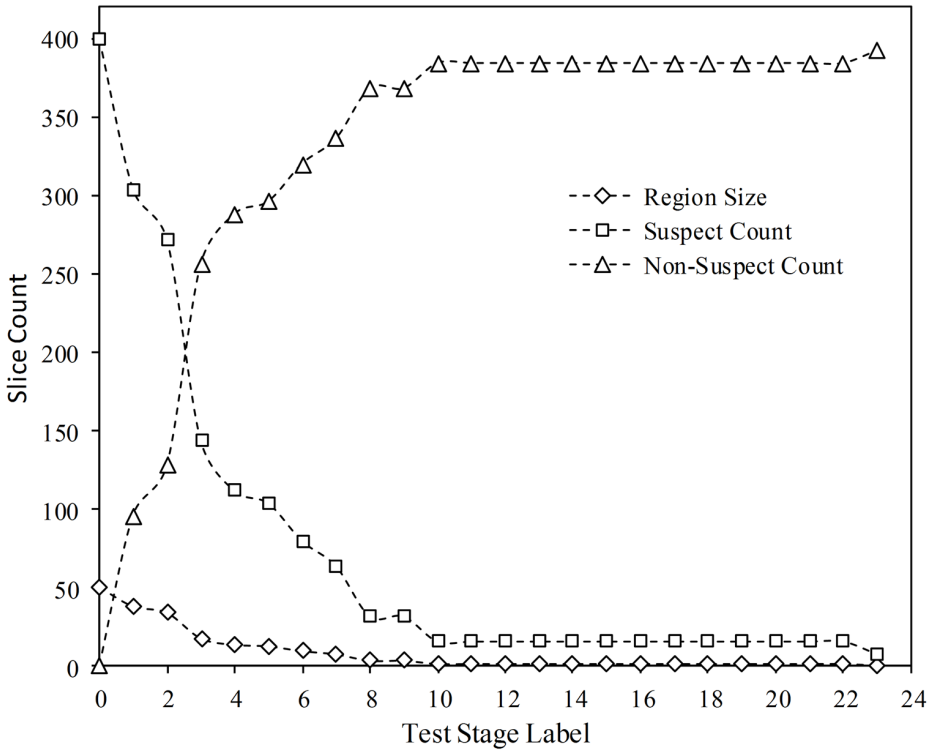


Figure 3.3: Convergence of suspect set, non-suspect set, and the region size for the tree multiplier

3.4 Results and Analysis

To observe how fault isolation is converged towards the locations of faulty slices, the numbers of suspects and non-suspects in every test stage are recorded for each case study. The region size is also examined as the test progresses for each case study. Figures 3.3 and 3.4 depict

the results for the tree multiplier and DES designs, respectively. As shown in both figures, the size of the suspect set in the first test stage equals the application size since all resources are considered faulty. The suspect set is narrowed down to those that implement a fault-affected configuration. The testing procedure terminates when the region size becomes one. The number of test stages needed to locate all faulty slices for the multiplier and DES designs are 24 and 20, respectively.

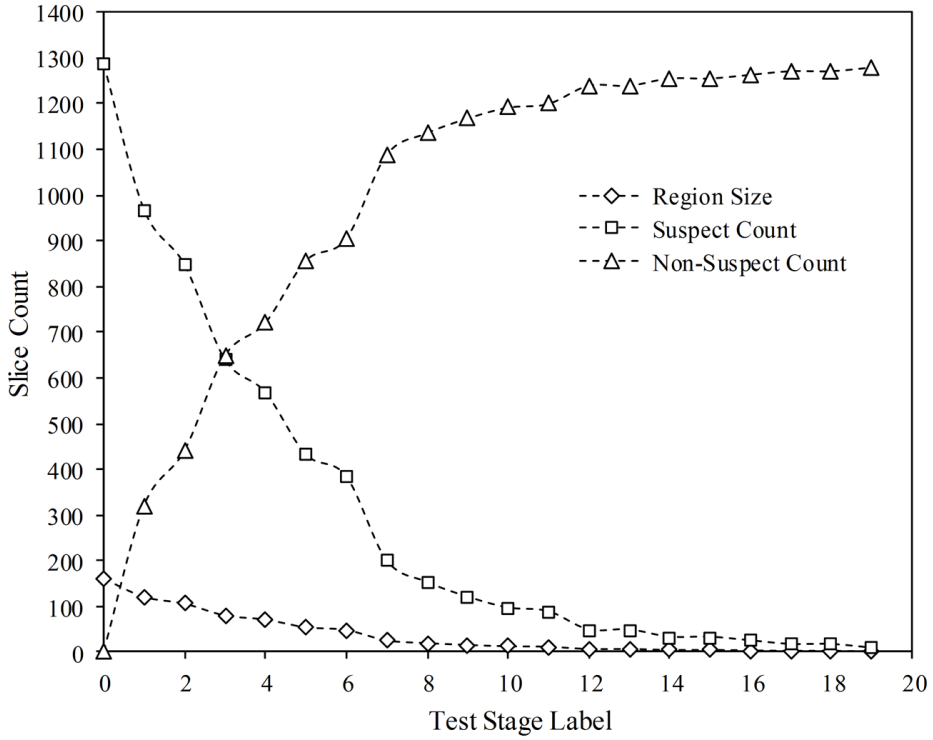


Figure 3.4: Convergence of suspect set, non-suspect set, and the region size for the DES

Figures 3.5 and 3.6 show the articulation count for each slice in the tree multiplier and DES designs, respectively. It is evident that the proposed procedure can identify all faulty slices after the testing procedure terminates. The maximum discrepancy count will favor faulty slices, leading to a distinct separation line between faulty and healthy slices. It is also observed from both figures that articulation data for healthy slices are clustered in a specific

range which can be used to determine the outlier status for each slice and thus the target isolation accuracy required to achieve reliability objectives.

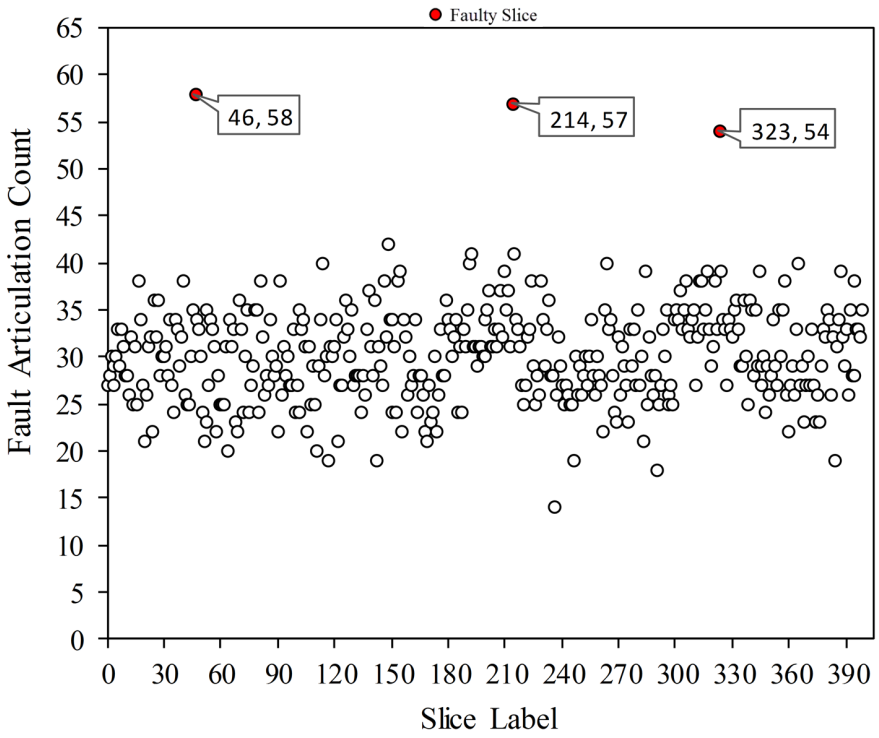


Figure 3.5: Slices articulation count for the tree multiplier

Table 3.1 lists the various algorithm parameters as the test progresses for the multiplier design. The maximum discrepancy count, denoted by H, reflects the successful fault articulation in each test stage; thus, leading to a fast convergence of the fault isolation approach.

3.5 Summary

In this chapter, a new probabilistic group testing scheme for isolating multiple faults in reconfigurable logics is described and evaluated using two case studies. The experimental results demonstrate successful fault isolation of up to three faulty elements using 24 and

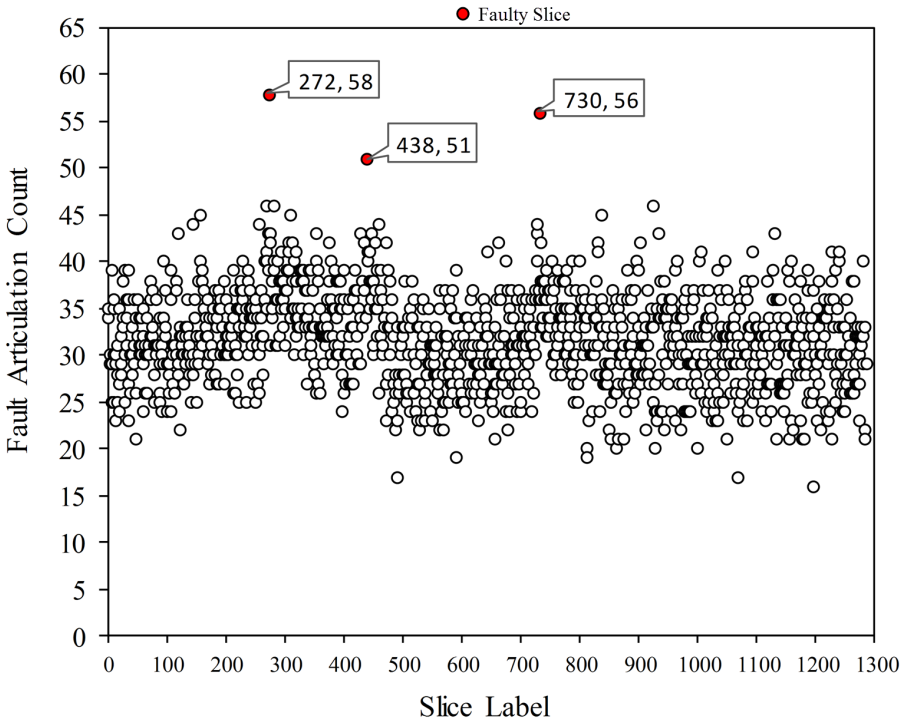


Figure 3.6: Slices articulation count for the tree DES.

20 test stages for an 8x8 tree multiplier and DES designs, respectively. The number of regions used in all experiments was fixed to eight, limiting the number of faults isolated by the method to at most three. However, the scheme can be further extended to include dynamic number of regions with more optimized resource selection mechanism to increase fault articulation rate.

Table 3.1: Parameters for Each Test Stage (DES Design)

Test Stage	Region Size	Suspect Count	Non-Suspect Count	H
0	50	400	0	0
1	38	304	96	2
2	34	272	128	4
3	18	144	256	6
4	14	112	288	8
5	13	104	296	12
6	10	80	320	13
7	8	64	336	14
8	4	32	368	16
9	4	32	368	17
10	2	16	384	18
11	2	16	384	20
12	2	16	384	22
13	2	16	384	28
14	2	16	384	30
15	2	16	384	32
16	2	16	384	34
17	2	16	384	36
18	2	16	384	38
19	2	16	384	40
20	2	16	384	43
21	2	16	384	45
22	2	16	384	47
23	1	8	392	48

CHAPTER 4: FAST ONLINE DIAGNOSIS AND RECOVERY OF RECONFIGURABLE LOGIC FABRICS USING DESIGN DISJUNCTION

In this chapter, 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 runtime 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% (94.4%) for $f = 1$ ($f = 2$). Results have also demonstrated millisecond order recovery with a minimum increase of 83.6% in fault coverage compared to N-modular redundancy (NMR) schemes. Recovery is achieved while incurring average critical path delay impact of only 1.49% and energy cost roughly comparable to conventional 2-MR approaches [42].

4.1 Design Disjunction

A new deterministic design space exploration (DSE) [43] [16] method is used to realize FPGA fault tolerance that achieves the availability and reliability objectives shown in Figure 4.1. 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.

The remainder of this chapter is organized as follows. First, an introduction to NGT and the property of f-disjunctness is provided, along with illustrative examples. Second, design for resource disjunction using the developed mosaic convergence algorithm is presented. Third, fault isolation and recovery schemes for reconfigurable fabrics using design disjunction are developed. Forth, evaluation results for several case studies are provided and discussed. Fifth, a comparison between the proposed work and existing schemes is presented. Finally, a brief conclusion is provided.

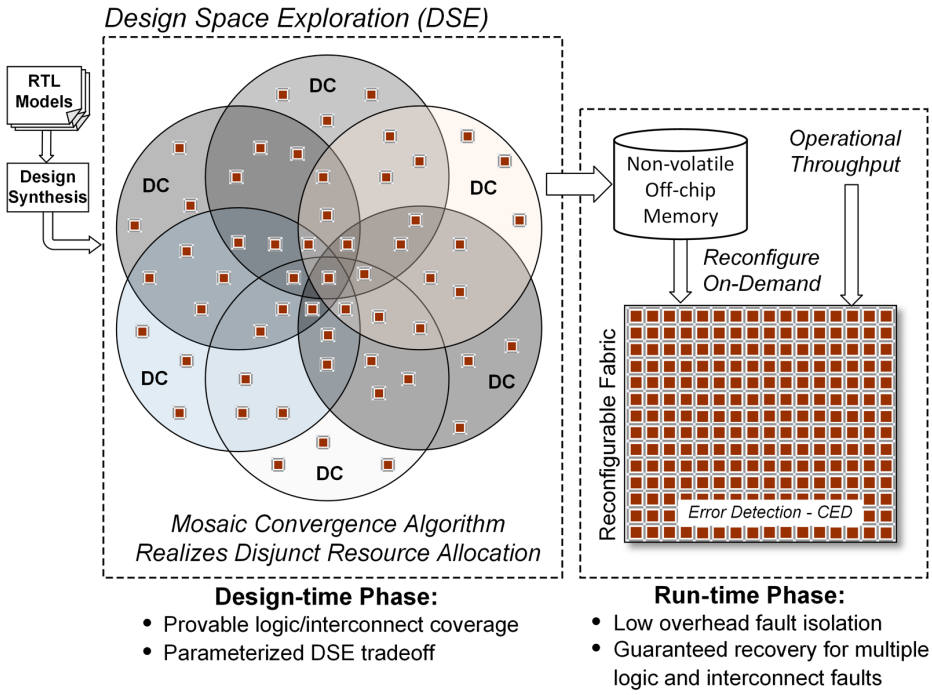


Figure 4.1: Objectives of proposed design disjunction technique.

4.2 Non-adaptive Group Testing

The proposed approach adopts a novel functional testing based on non-adaptive group testing to realize efficient online fault isolation and recovery in reconfigurable devices using *design disjunction*. Design disjunction is a group testing approach using a set of equivalent DCs, each of which utilizes a group of FPGA resources to implement functionality in the target device. These alternative DCs have been placed and routed such that the resource designation of each design is governed by the f-*disjunctness* property. This non-adaptive group testing approach provides highly-compressed diagnosability which can significantly lower testing costs for online fault localization. In addition, the combinatorial properties conferred by f-disjunctness allow low-overhead online fault evasion using a small number of reconfigurations with low requirement of storage capacity.

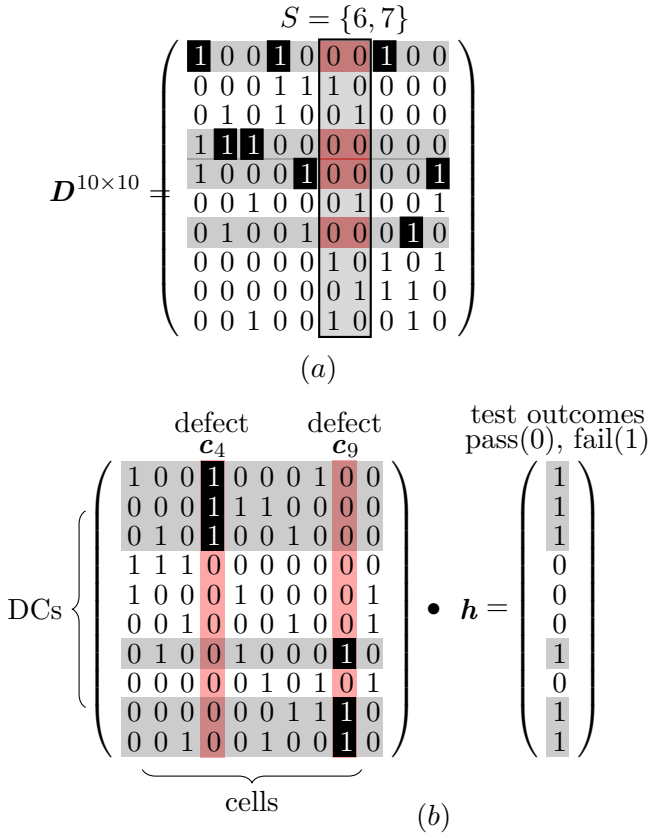


Figure 4.2: (a) Example of a 2-disjunct design matrix. (b) Conventional diagnosis decoder.

Table 4.1 summarizes features of the proposed scheme and other related approaches discussed in chapter 2.

Table 4.1: Comparison of Design Disjunction with Related Approaches

Approach	**Runtime Fault Isolation**	**Resource Coverage: Resolution**	**Provable Multiple-fault Coverage**	**Error-tolerant Fault Isolation**	**Recovery Latency**	**Intrinsic Wear-leveling**	**Recovery-time Routing**	**Advantage**
STARs [12]	Yes	Logic: LUT	Yes	No	Exhaustive BIST Overhead	No	Required	Resource Recycling
R3TOS [30]	Yes	Logic: LUT	Yes	Yes	Exhaustive BIST Overhead	No	Required	Robust Control Mechanism
Module Diversity [36]	No	Logic: CLB	No	No	μs$\rightarrow$ ms	Yes	Unnecessary	Effective Aging Mitigation
Hahanov et al. [35]	No	Logic: CLB	Yes	No	Routing Overhead	No	Required	Provable Coverage
AGT [34]	No	Logic: slice	No	No	PAR Overhead	No	Required	Intrinsic Adaptation
Consensus-Based Evaluation [15]	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*

For 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 $\boldsymbol{v}$ whose entries are one.
- $\omega(\boldsymbol{v})$ is the weight of binary vector $\boldsymbol{v}$, i.e. number of elements whose entries are one.
- Γ_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:

$$\boldsymbol{o}^{g\times 1} = \boldsymbol{D}^{g\times T} \cdot \boldsymbol{h}^{T\times 1} \tag{4.1}$$

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 [44]. 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 $\boldsymbol{D}^{g\times T}$ 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 [45]. This can be expressed as:

$$\forall S \in \Gamma_f^T,\ \sum_{i=1}^{g} \left(D_{i,j} = 1\ \wedge \bigcup_{k\in S} D_{i,k} = 0 \right) \geqslant \delta \tag{4.2}$$

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. (4.2). 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. Figure 4.2(a) shows a 2-disjunct matrix and a one subset of columns, S, of size 2 that meets the condition given by eq. (4.2) 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:

$$\boldsymbol{h} = \{h_k \mid h_k = \begin{cases} 1 & if \;\; \psi(\boldsymbol{c}_k) \subseteq \psi(\boldsymbol{o}) \\ 0 & otherwise \end{cases}, \; 1 \leq k \leq T\} \tag{4.3}$$

Figure 4.2(b) illustrates how the same 2-disjunct matrix is used to single out the two defective resources, 4 and 9, using the described decoding method. In this example, the sparse health vector is given as:

$$\boldsymbol{h} = (\; 0\; 0\; 0\; 1\; 0\; 0\; 0\; 0\; 1\; 0\;)^T \tag{4.4}$$

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. 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 this chapter, we address these two challenges and present results demonstrating the feasibility and advantages of proposed approach.

4.3 Design for Disjunction on Reconfigurable Architectures

The structural regularity of modern reconfigurable devices and the efficient diagnosability of non-adaptive group testing can be exploited to realize an effective FT scheme. The conceptual idea behind design disjunction is to realize a set of disjunct DCs and employs them to evade and locate defective resources during system operation while maintaining optimal availability. These DCs are functionally equivalent, but disjunct in terms of the physical resources that they utilize. Fault tolerance can be achieved by runtime reconfiguration to load any design in the DCs set that do not utilize defective resources. The diversity in resource utilization of DCs can also be leveraged during normal system operation for stress relief to extend system lifetime. As DCs are implemented prior to deployment, only partial reconfiguration overhead is imposed during testing and recovery. Thus, the need to invoke the computationally intensive design flow during testing and recovery is eliminated.

The constructive f-disjunctness property provides an effective way for extracting highly fault-resilient DCs against logic and interconnect failures. Contemporary FPGAs have different levels of programmable cell granularity. For instance, the basic logic elements such as LUTs, and flip-flops in the Xilinx FPGA families are organized into logic blocks called *slices*. Slices are also grouped into CLBs. In this work, design disjunction is examined at the slice level. Thus, in the rest of this dissertation, the terms slice and resource are used interchangeably. To achieve design disjunciton at slice level, the columns in the design matrix correspond to slices and rows represent DCs. We focus on logic fault localization. However, without loss

of generality, the proposed scheme can be combined with a low-cost application-dependent interconnect testing such as [31] to achieve diagnosis resolution at the interconnect point level. Moreover, incidental fault recovery of interconnect resources is also seen, as demonstrated in case studies.

4.4 Constructing Disjunct DCs

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 [44]. 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 4.4.1 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 4.4.1. *Disjunction ratio* (DR) is the proportion of Γ_f^T elements that satisfy the condition stated in eq. (4.2).

Algorithm 4.4.1: 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
δ: Disjunction Factor
Output: Design Matrix, $\boldsymbol{D}^{g \times T}$.

```
1  φ := (T choose f) = T!/(f!(T−f)!)
2  ε := φ × (T − f)  // binary check count
3  DR := 0
4  Generate a random row vector v, s.t.: length(v) = T and ω(v) = R
5  g := 1  // point to the first row of D
6  D_g := v  // insert v as the first row of the design matrix
7  g := g + 1
8  C := Γ_f^T  // set of all f-combinations out of T
9  λ^{φ×T} := [δ]^{φ×T}  // initialize binary coverage matrix entries to δ
10 DR_func(v)  // call function DR_func to update DR after inserting the row vector v
11 while (DR ≠ 1) do
12     v := [1]^{1×T}  // start with a row vector v s.t. length(v) = ω(v) = T
13     S_max := C_{z_max}
14     for each k ∈ S_max do
15         v_k := 0
16     while (ω(v) ≠ R) do
17         max := 0
18         for i := 1 to T do
19             if (v_i ≠ 0) then
20                 t := v
21                 y := λ_{z_max}
22                 t_i := 0
23                 count := 0
24                 for each S ∈ C s.t. i ∈ S do
25                     for j := 1 to T do
26                         if (t_j = 1 ∧ y_j ≠ 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     D_g := v
34     g := g + 1
35     DR_func(v)

   // update DR after inserting a new row
36 Function DR_func(a)
37     count := 0
38     max := 0
39     for z := 1 to φ do
40         S := C_z
41         if (∀ k ∈ S, a_k = 0) then
42             for j := 1 to T s.t. j ∉ S do
43                 if (λ_{z,j} ≠ 0 ∧ a_j = 1) then
44                     λ_{z,j} := λ_{z,j} − 1
45                     count := count + 1
46         if (ω(λ_z) > max) then
47             z_max := z
48             max := ω(λ_z)
49     DR := DR + count/(ε × δ)
```

The binary coverage matrix λ (line 9) tracks whether each combination $S \in \Gamma_f^T$ has satisfied the condition in eq. (4.2). 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 λ 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 $\Re \subseteq \Gamma_f^T$ that have not yet satisfied the condition expressed in eq. (4.2). 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.

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 $\delta = 1$. Figure 4.3 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 [46] [40]. Figure 4.3 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.

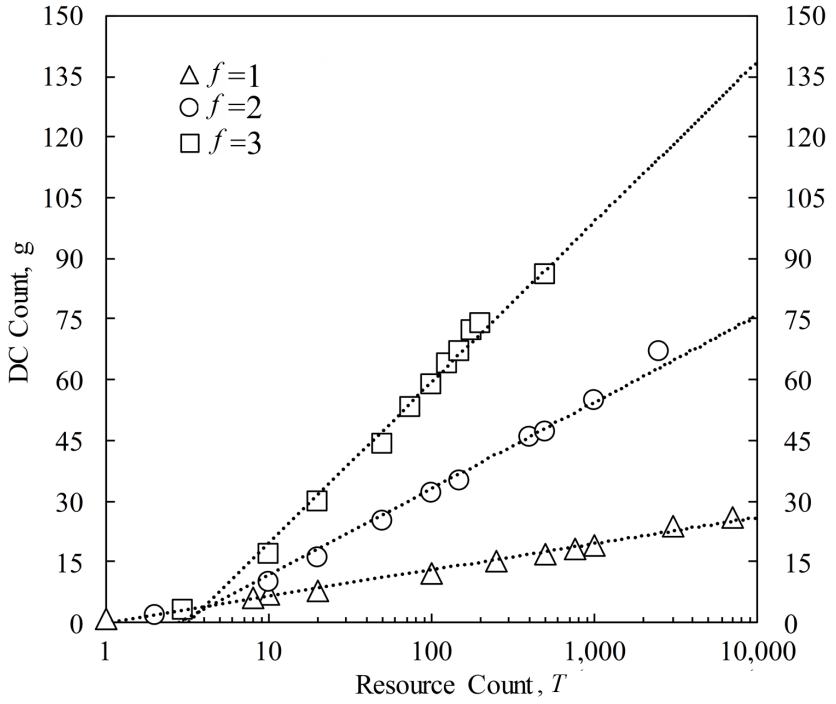


Figure 4.3: Required number of DCs vs. resource count for typical values of f ($\delta = 1$).

4.5 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:

$$FSR_i = \frac{\left| \bigcup_{k=1}^{g} D_{k,i} \mid \ D_{k,i} = 1 \ \wedge \ o_k = 1 \right|}{\omega(\mathbf{c}_i)} \ , \ 1 \leq i \leq T \tag{4.5}$$

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

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.

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

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. Figure 4.4 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 4.4.1, 16 DCs

(indexed 1-16) are found sufficient to achieve 1-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 4.5.1. *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% indicates that $\lfloor 998 \times 95\% \rfloor = 948$ of non-faulty resources score lower FSR values than the two defects.

Definition 4.5.2. *Fault coverage* is the proportion of all combinations of faulty resources of size up to f that attain a specified isolation accuracy.

Figure 4.5 shows the required number of DCs, g, to reach various isolation accuracies and their fault coverage values. The results also demonstrate how Algorithm 4.4.1 progresses towards the termination criteria, i.e. $DR = 100\%$, 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% isolation accuracy. The value of g can be considerably reduced while maintaining a high isolation accuracy. A reduction of 36.4% (61.8%) in g results in a slight decrease in isolation accuracy of 1% (5%). 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

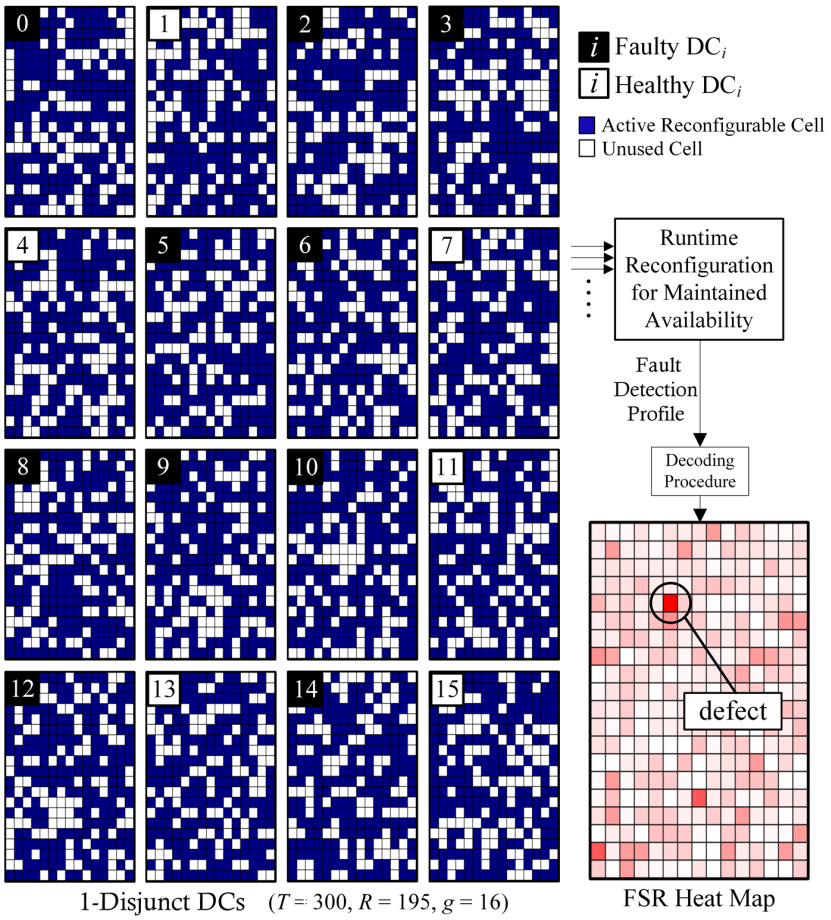


Figure 4.4: Fault diagnosis using FSR metric.

self-repair. It is important to note again that these simulation results are collected under the conditions of reliable tests.

4.6 Fault Recovery using Design Disjunction

The combinatorial characteristics of f-disjunct design matrices add another advantage for design disjunction. The definition expressed in eq. (4.2) implies that any f-disjunct set of DCs should guarantee that for any possible accumulation of f faulty resources there exists

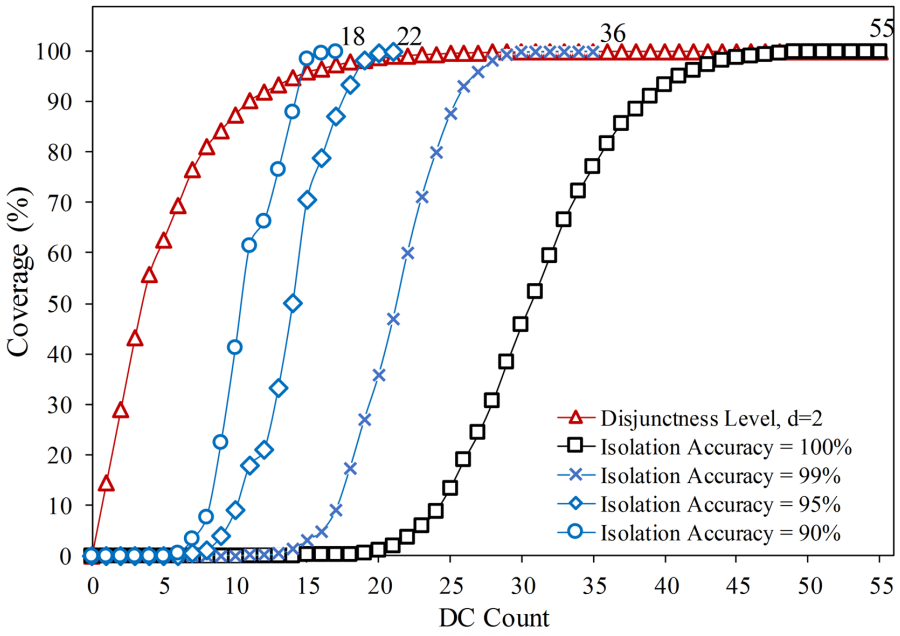


Figure 4.5: Test coverage vs. number of DCs (T=1000, d=2).

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:

$$p_{dc\_nf}(d) = \prod_{k=1}^{d} \left(1 - \frac{R}{T-k-1}\right), \; d \geqslant 1 \tag{4.6}$$

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

$$RC(d) = 1 - [1 - p_{dc\_nf}(d)]^{g} \;, \; d \geqslant 1 \tag{4.7}$$

where g is the number of DCs.

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. Figure 4.6 compares simulation results against our model given by eq. (4.7). 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. (4.6). 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.

4.7 Incidental Disjunction for Interconnect Fault Tolerance

Contemporary reconfigurable devices utilize hundreds of thousands of routing points. For instance, the Xilinx 7-series FPGA family fabricated in 28nm process allows over 3,500 Programmable Interconnect Points (PIPs) to be defined in each switch matrix tile of the device. This enormous amount of routing flexibility presents a significant challenge for runtime interconnect testing and diagnosis. Specialized functional testing for interconnects based on output pattern analysis [31] has been shown effective for diagnosis at the net-level of a single design. However, a given net can utilize a considerable number of PIPs spanning multiple interconnect tiles which can prolong the self-repair process. Therefore, an optimal set of pre-compiled DCs that can evade a large set of faults is highly desirable for rapid recovery. Since interconnect utilization is precipitated by mapping and placement

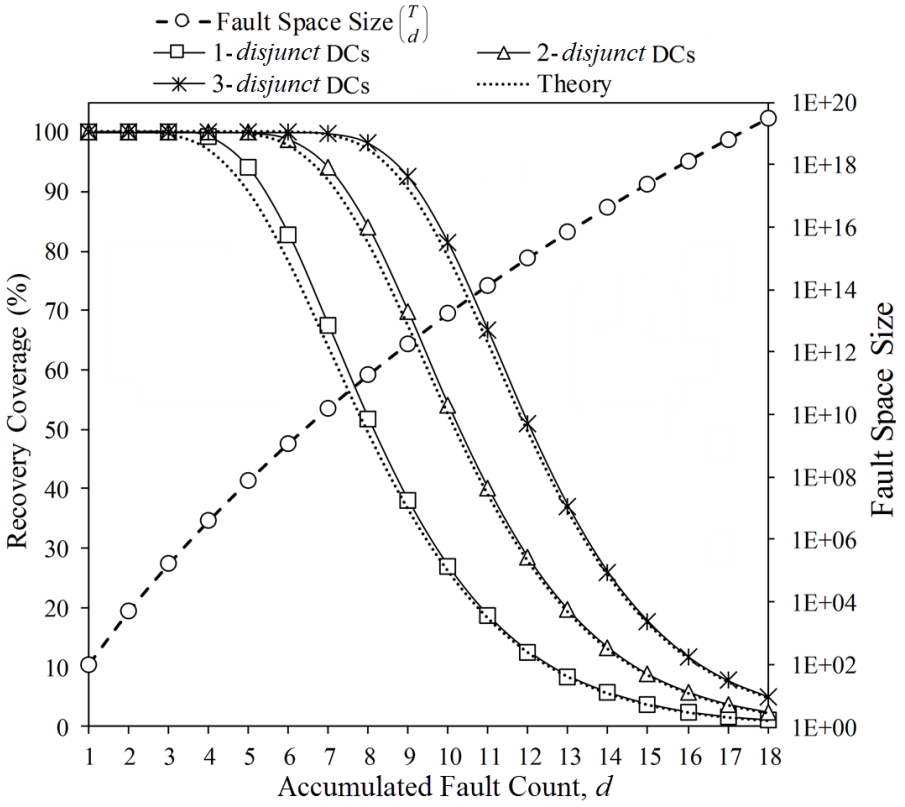


Figure 4.6: Fault evasion coverage for f-disjunct set of designs.

of logic blocks, a strong design disjunction in the logic fabric has been demonstrated to also precipitate very strong incidental disjunction in interconnect resources. This property effectively extends non-adaptive fault recovery to routing fabrics as demonstrated later in the case studies.

4.8 Inarticulate Functional Tests

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 guar-

anteed 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 [40] [47]. 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 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 4.4.1 is also used to realize design disjunction for $\delta > 1$. Figure 4.7 shows the number of DCs for 1-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 the evaluation section, the effect of increasing δ on fault diagnosis for various case studies are studied in which we compare the isolation accuracy under the low coverage of operational testing.

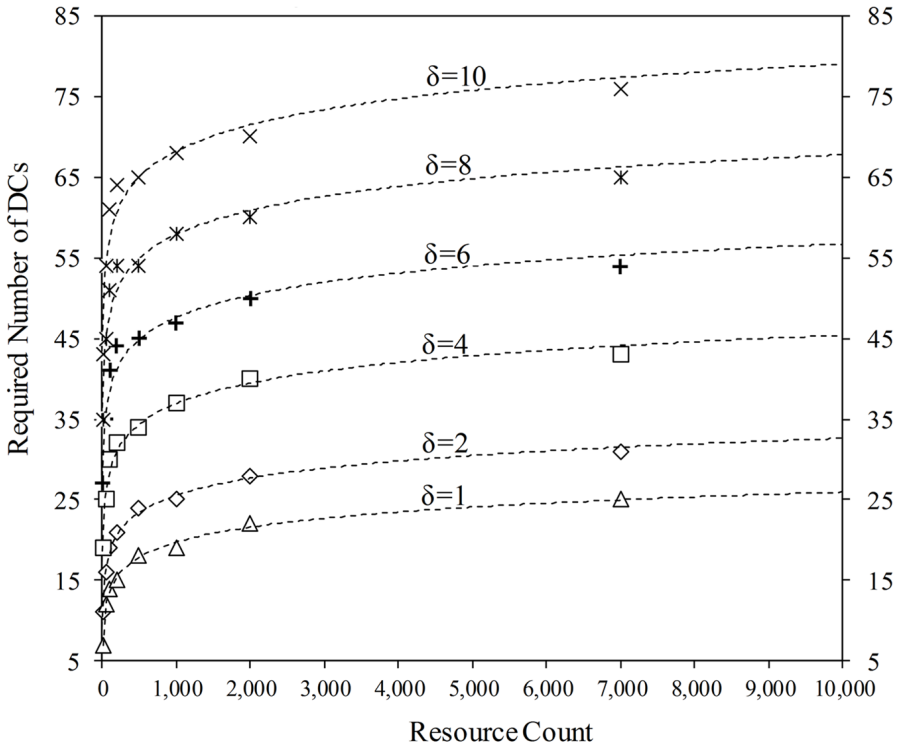


Figure 4.7: DC count for increasing e ($f = 1$).

4.9 Case Studies

The proposed design disjunction was evaluated on a set of MCNC and ISCAS to show its applicability to a broad range of designs. Two real world applications: AES128 encryption core and discrete cosine transform (DCT)-based image processing application are also implemented in an FPGA test board to demonstrate autonomous fault diagnosis and recovery using design disjunction. These two real world examples represent two class of applications. The AES encryption core is an example of applications that are sensitive to failure and do not provide inherent fault tolerance. The 2D-DCT image processing core is a known example to a class of applications that allow degraded performance mode whereby system functionality can still provide a useful throughput even under the presence of multiple faults. Addition-

ally, these applications can adapt diverse ways of fault detection methods. We show how fault isolation and recovery performance of the proposed technique can work with different detection techniques such as parity-based, duplication with comparison, and system-level metrics such as the peak signal-to-noise ratio(PSNR).

For all case studies, Xilinx 7-series FPGAs using Xilinx design and simulation toolsets were used to generate DCs. For hardware implementation, the Xilinx KC705 embedded board [48] is adopted, Figure 4.8. This FPGA test board features: 28nm Kintex-7 FPGA, 1 GB DDR3 memory, 128MB linear flash memory, and a USB JTAG port.

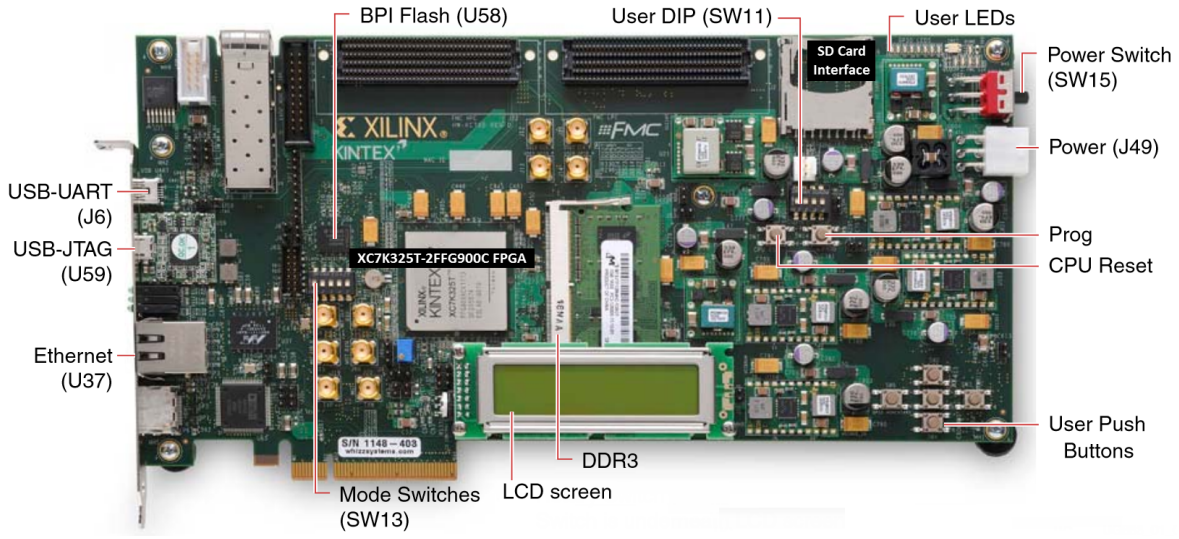


Figure 4.8: KC705 board components.

Other distinction in implementation and design flow will be described for each case study.

4.9.1 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: 28nm-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 Figure 4.13. 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 (RM) inside the RP. Resource allocation for each RM is dictated by the design matrix constructed for the target application according to the discussed design parameters. 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 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.

4.9.2 Case Study 1: ISCAS and MCNC Benchmarks

For each MCNC and ISCAS benchmark, two f-disjunct sets of DCs are generated for $f = 1$ and $f = 2$. Table 4.2 lists the isolation accuracy results averaged over $1,000$ experimental runs on all benchmarks for $f = 1$ and $f = 2$. Results include the 95% 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 4.2 also shows that the average isolation accuracy over all benchmarks for $f = 1$ ($f = 2$) is 96.4% (94.4%). 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. Figure 4.9 indicates the articulation rate for each benchmark included in this evaluation which supports the need for a design measure to overcome the low coverage of operational testing. It is also worthy to note that test coverage 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.

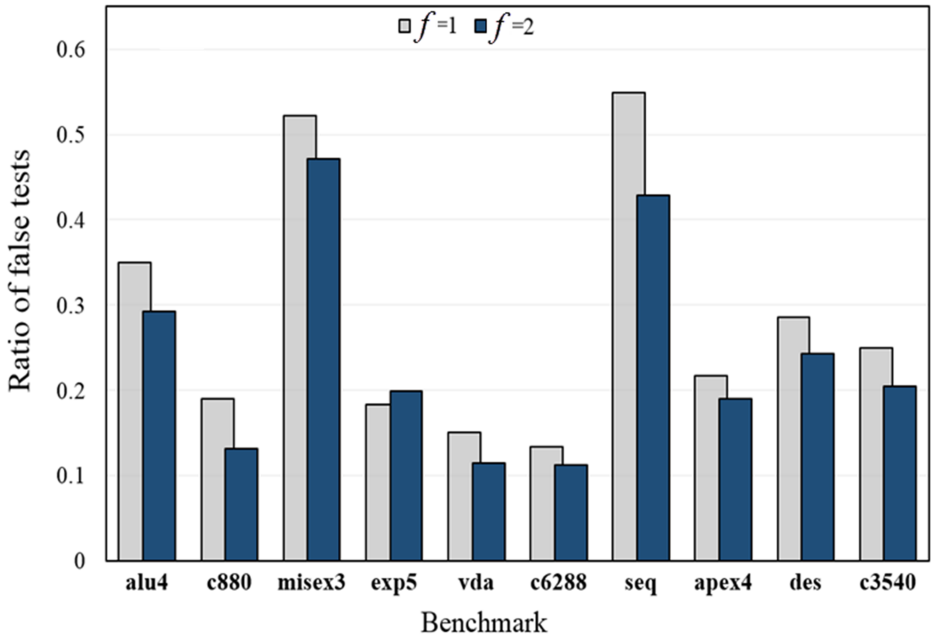


Figure 4.9: Articulation rate for the ISCAS and MCNC benchmarks (f=1, $\delta = 1$).

Design disjunction for $\delta > 1$ is also evaluated to demonstrate feasibility to reach optimal fault isolation under inarticulate testing. Table 4.3 shows how design disjunction for a moderate increase in disjunction factor δ results in a greater than 99% 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 $> 99\%$ given by the upper 95% 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.

Figure 4.10 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 4.2. 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.

Table 4.2: Isolation Accuracy Results ($\delta = 1$)

Benchmark Circuit	R	$f=1$ T	g	τ_{mc} (ms)	Isolation Accuracy (%) μ	95% CI lower	95% CI upper	$f=2$ T	g	τ_{mc} (s)	Isolation Accuracy (%) μ	95% CI lower	95% CI 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

Table 4.3: Isolation Accuracy vs. δ for Selected Benchmarks ($f = 1$)

δ	misex3 g	misex3 τ_{mc} (ms)	misex3 Isolation Accuracy (%) μ	misex3 95% CI lower	misex3 95% CI upper	c3540 g	c3540 τ_{mc} (ms)	c3540 Isolation Accuracy (%) μ	c3540 95% CI lower	c3540 95% CI upper	alu4 g	alu4 τ_{mc} (ms)	alu4 Isolation Accuracy (%) μ	alu4 95% CI lower	alu4 95% CI 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	–	–	–	–	–	–	–	–	–	–

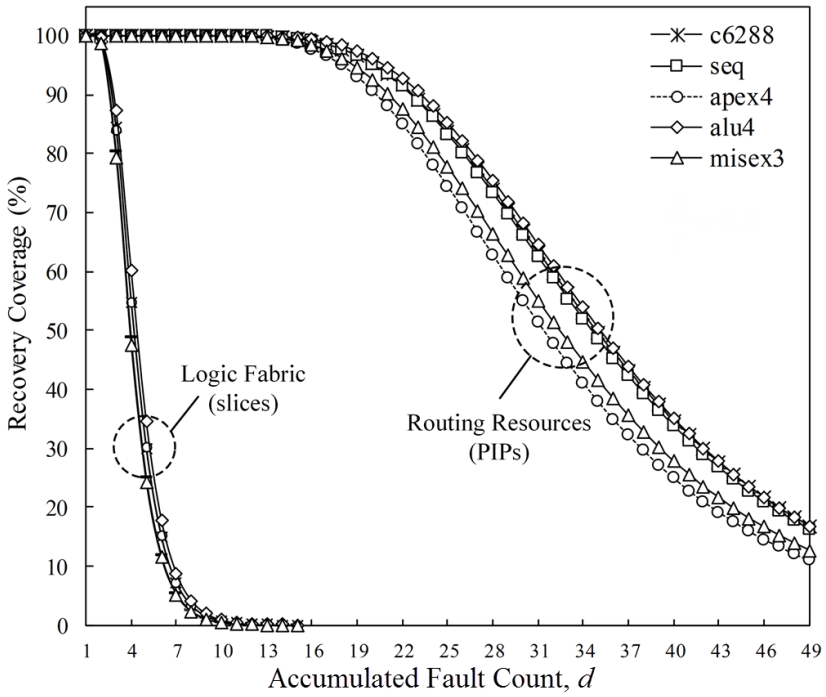


Figure 4.10: Fault recovery coverage (f=1, $\delta = 1$).

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:

$$s = t_{target} - t_{total} = t_{target} - [t_{cp} - t_{cps} + t_{cu}] \tag{4.8}$$

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 4.11 shows s and t_{cp} data for each benchmark. The average increase in t_{cp} compared to the baseline design is 1.49% and the average decrease in the ratio of the total slack to the total delay is only 1.78%. It is also observed that the top-performing DC can be slightly faster than the baseline design due to the stochastic nature of placement and routing algorithms which does not guarantee convergence to the optimal solution.

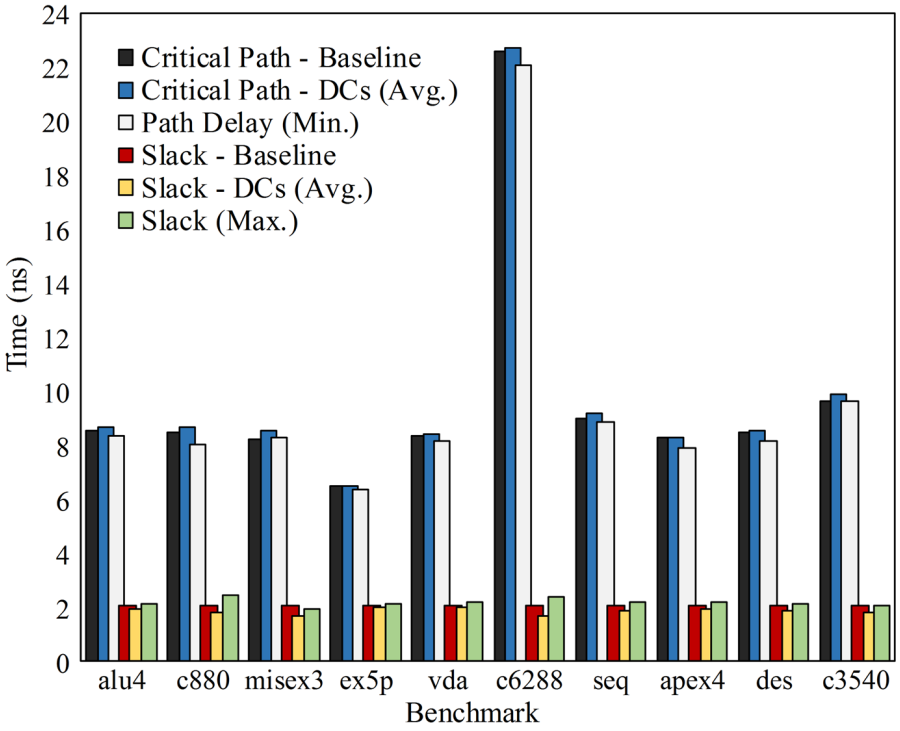


Figure 4.11: Effect of design disjunction on system performance.

4.9.3 Case Study 2: AES-128 Encryption Core

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-

ure 4.12 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 [49]. 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.

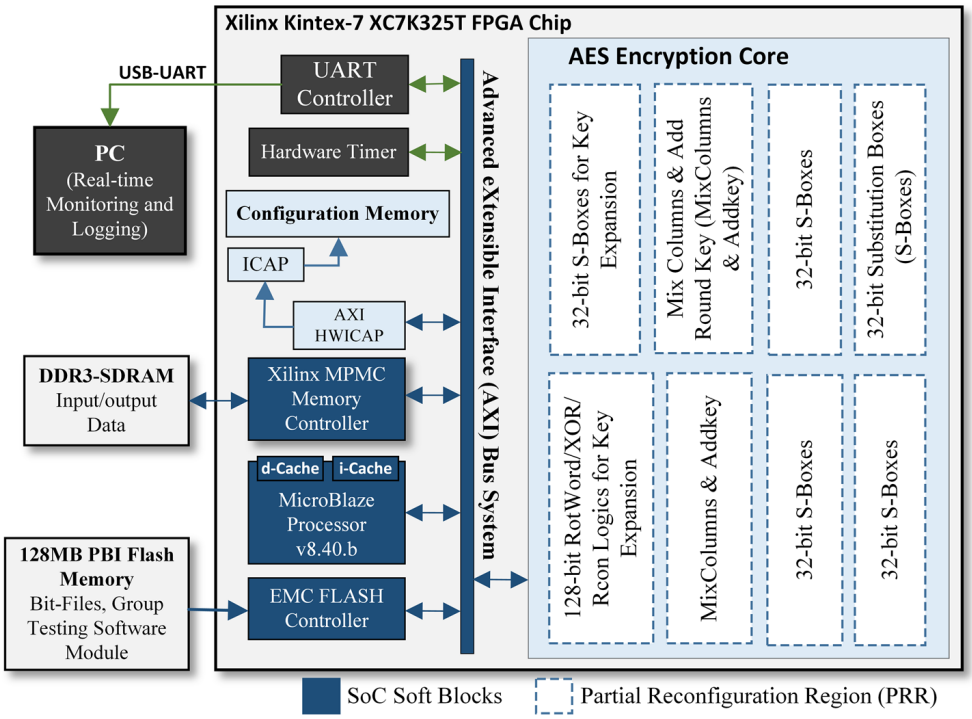


Figure 4.12: Hardware implementation block diagram for proposed FT scheme.

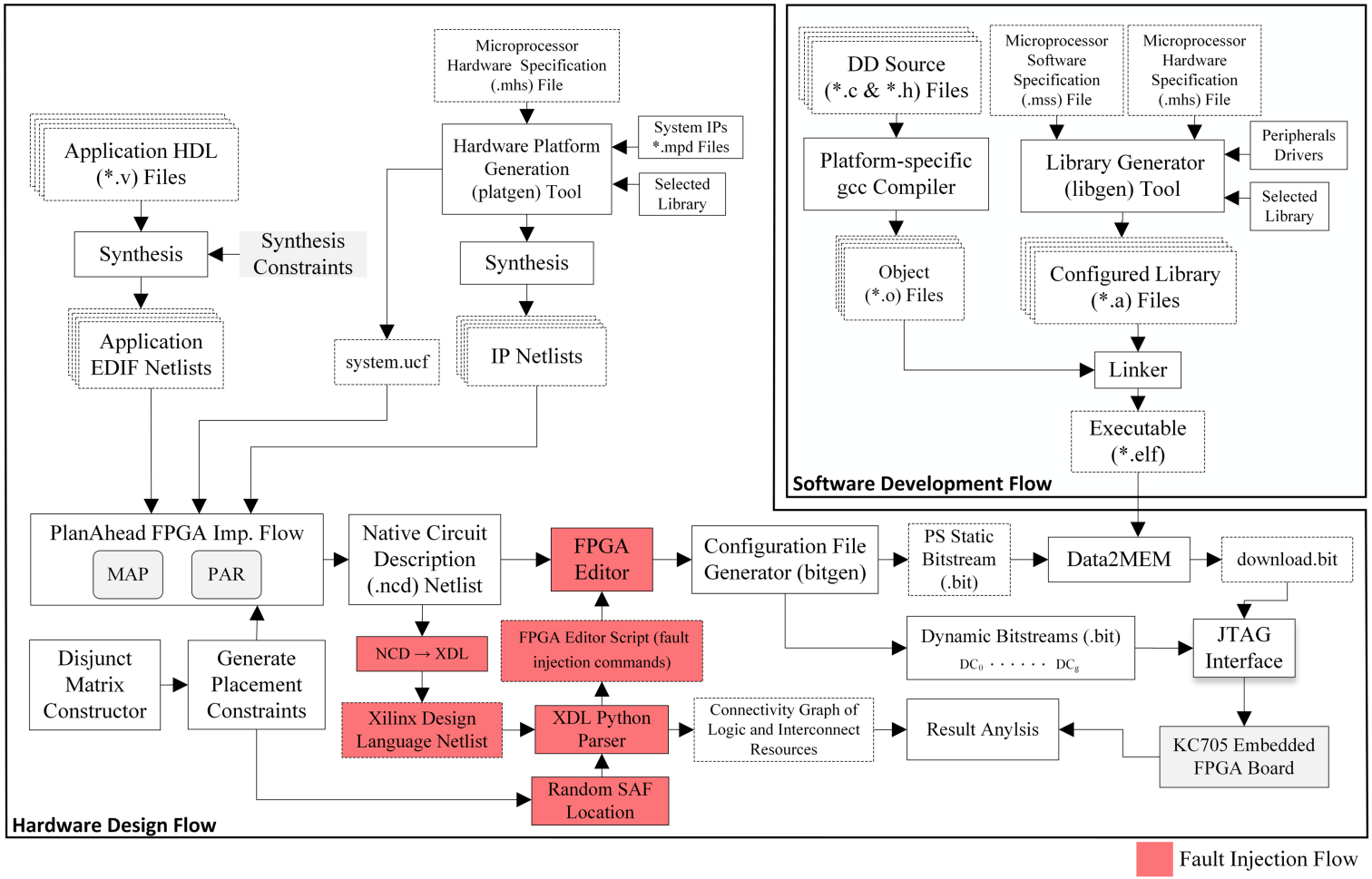


Figure 4.13: Design flow and fault injection for hardware implementation.

Table 4.4: Design Parameters for AES Modules

Module	R	T	δ	g	τ_{mc}(ms)	Bitstream Size	Detection Scheme
32-Bit s-boxes	60	119	3	24	41	57.9 KB	Parity-based [49]
Mix Columns & Add Round Key	55	111	3	24	39		
128-bit Rotate/Rcon Logics for Key Expansion	52	102	3	23	32		DWC

Table 4.4 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 Figure 4.12. 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.

Figure. 4.14(a) and Figure. 4.14(b) 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. Figure 4.14(a) 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 Figure 4.14(b) 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.

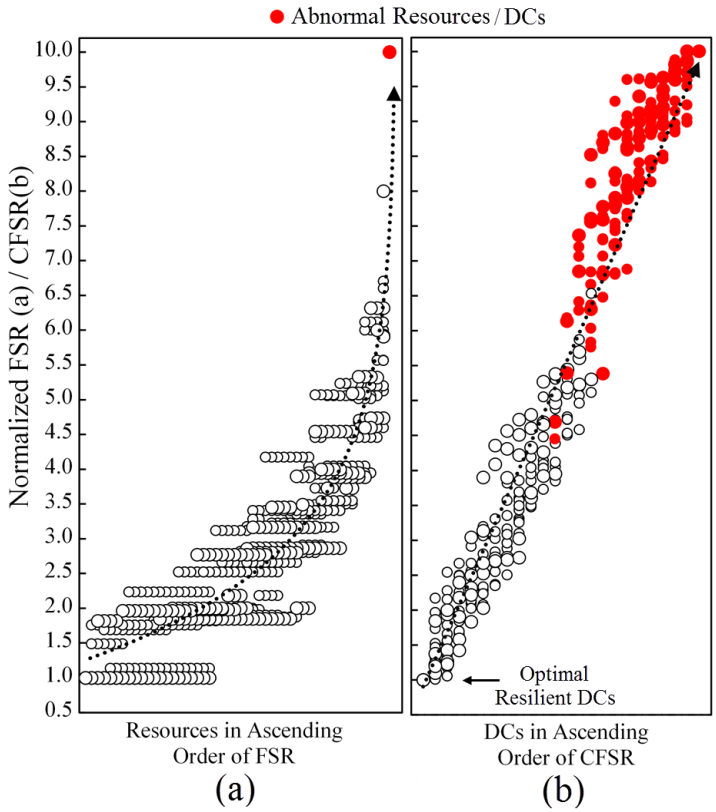


Figure 4.14: Diagnostic results for resources and DCs ($f = 1$, $e = 2$).

Figure. 4.15 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.

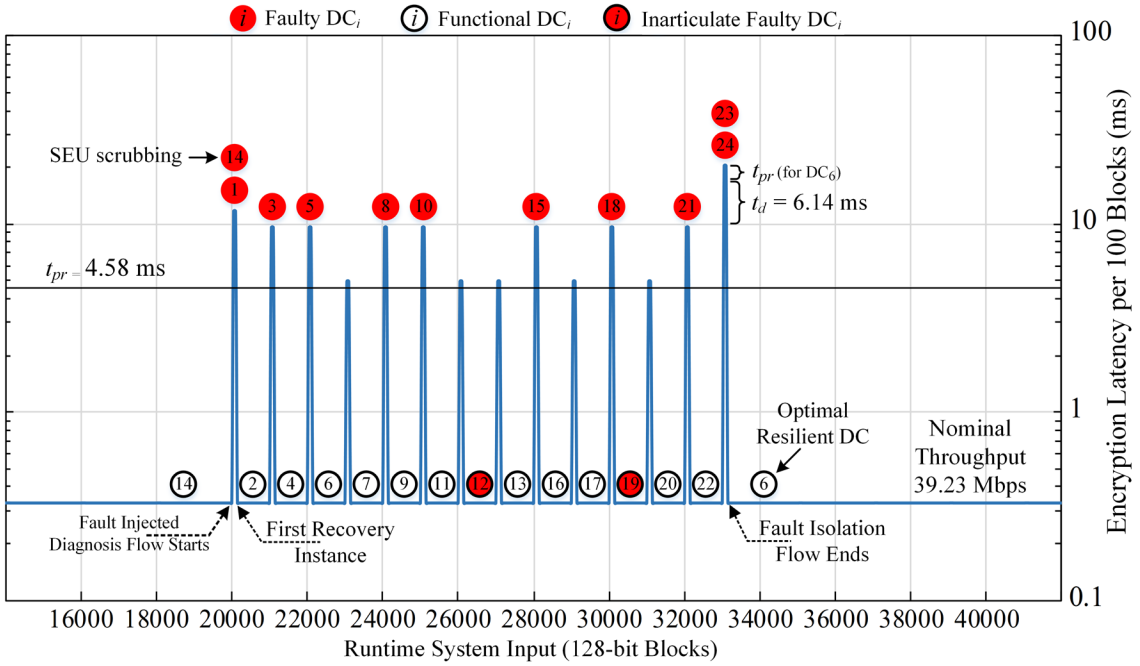


Figure 4.15: Execution of isolation phase on an AES module.

The hardware prototype illustrates and validates the proposed disjunction technique with an embedded processor. For practical applications, hard-core processing systems instead of soft-core IPs can enable lower processing and reconfiguration overheads.

4.9.4 Case Study 3: 2D-DCT Image Processing Core

The DCT-based Motion-JPEG (MJPEG) was selected due to its widespread use for image and video compression. Image and video compression applications are commonly used in space exploration missions, earth satellites, and monitoring systems in high radiation-dose environments. MJPEG core comprises different blocks, most of which are not computationally intensive and can be implemented in software with a reasonable efficiency. Contrarily, the DCT block in MJPEG core carries out high computational workload; thus, to reach a balance among MJPEG pipeline stages and get far better efficiency, it is favorable to implement DCT block using a dedicated reconfigurable hardware. Figures 4.16 and 4.17 present a few overhead results collected during implementation phase on the KC705 FPGA board which support these facts.

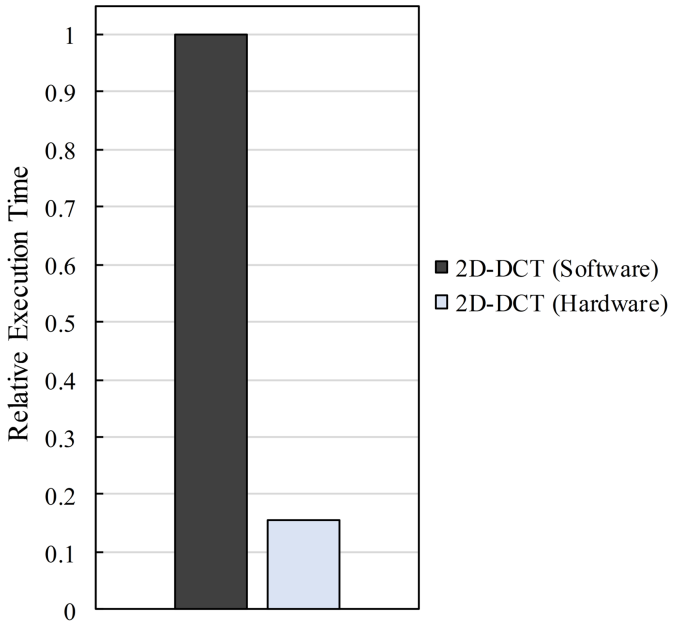


Figure 4.16: Hardware implementation speed-up for the 2D-DCT block on the KC705 board.

In this case study, we show how fault diagnosis and recovery based on proposed approach

can be effective for this type of applications. By using quantifiable characteristics such as the PSNR [50], the need for hardware error detection can be eliminated, thereby a large saving in area and power can be obtained. PSNR is based on mean square error (MSE) computed by averaging the intensity difference between pixels of recovered and original images. The recovered image is obtained by a fast software-based inverse discrete cosine transform (IDCT).

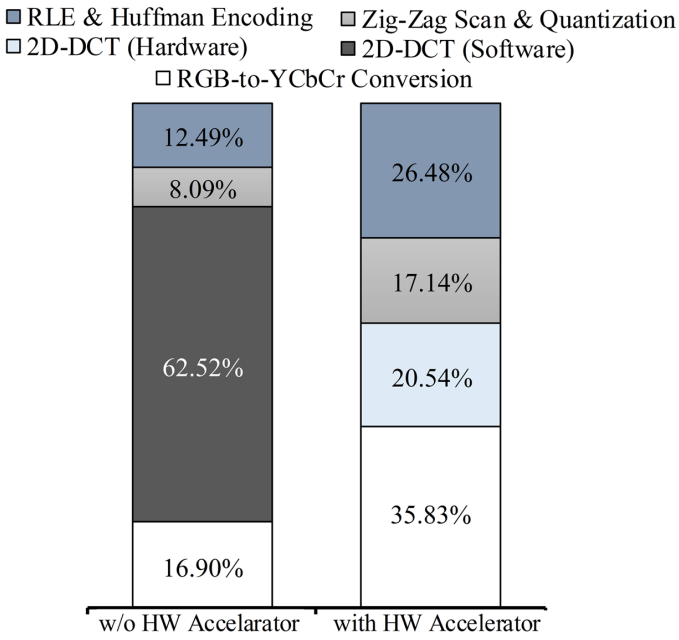


Figure 4.17: Overhead distribution of M-JPEG blocks on the KC705 board.

Table 4.5 shows the required number of resources to implement the DCT accelerator on the KC705 board, the total dedicated resources for its RP, the number of DCs required for 1-disjunctness, and bitstream size for its RP.

Fault isolation using still images are considered in this work. To achieve fault isolation using the PSNR metric, the PSNR value for each DC is computed for the same image frame. The problem of finding which DC is faulty turns out to be equivalent to finding the *mode* of the

Table 4.5: Design Parameters for the DCT Hardware Accelerator

R	T	δ	g	Bitstream size	Detection Scheme
422	930	1	19	193 KB	PSNR (System-level Metric)

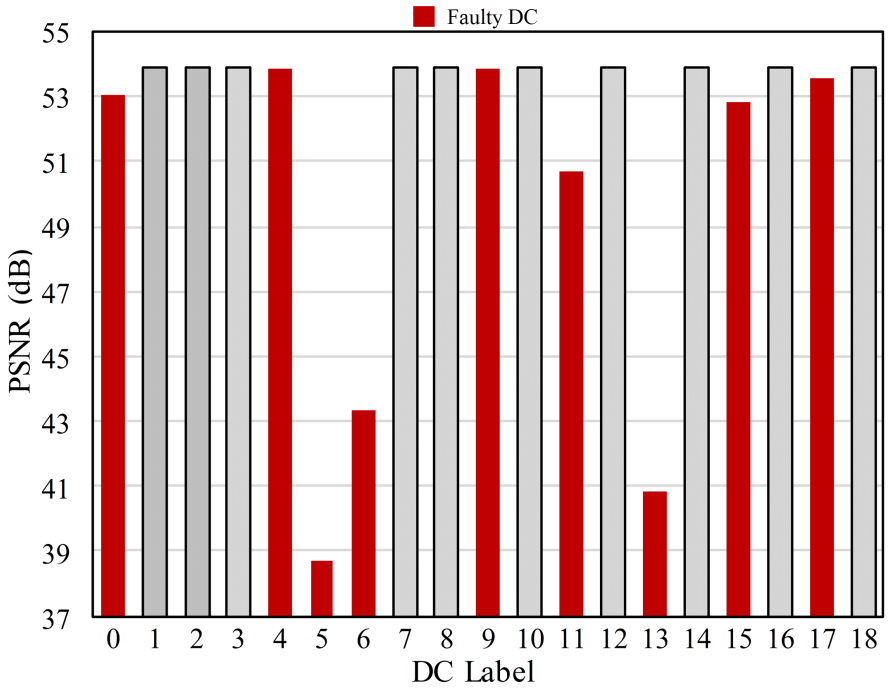


Figure 4.18: Obtained PSNR value for all DCs.

collected PSNR values. Since each DC uses resources in a different way, the PSNR value for each faulty DC tends to be unique. In contrast, healthy DCs maintain a similar PSNR value. By finding the mode, the described decoding scheme using the FSR and $CFSR$ metrics can be used to isolate faulty resources. Figure 4.18 shows the PSNR values for all DCs after injecting a single random SAF. As seen in the figure, healthy DCs can be clearly distinguished by simply finding the most commonly repeated PSNR value. Figure 4.19 illustrates the computed FSR in ascending order for all resources. The resource having the

highest FSR value identifies the faulty resource, i.e. `X113Y228`.

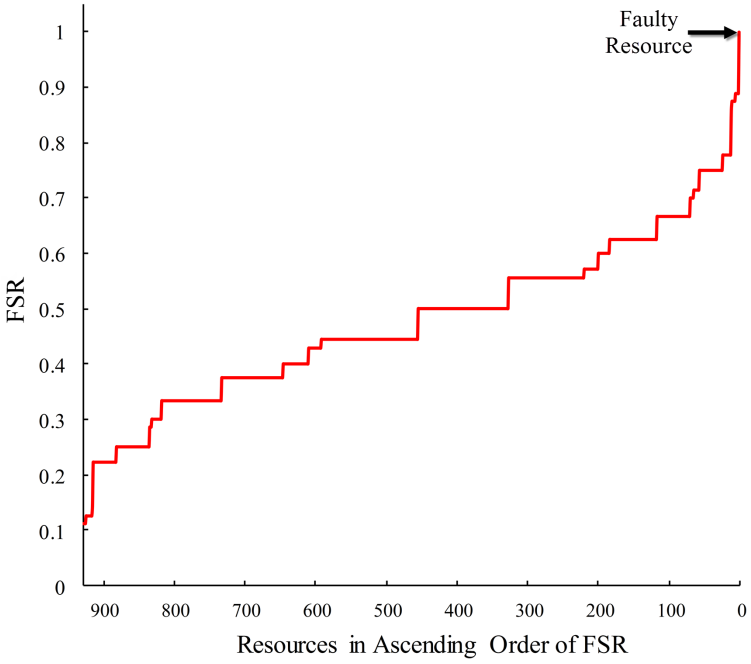


Figure 4.19: FSR in ascending order for all resources.

Fault recovery was also demonstrated against a large fault rate. The best operational DC is determined by the highest recorded PSNR. Figure 4.20 shows fault recovery rate using the proposed approach for the DCT core. Figure 4.21 shows the best JPEG images that can be generated from all DCs under different fault rates. These images reflect the highest image quality identified by the PSNR metric. Image quality was observed to deteriorate as the number of injected faults increases beyond 30 faults.

4.10 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

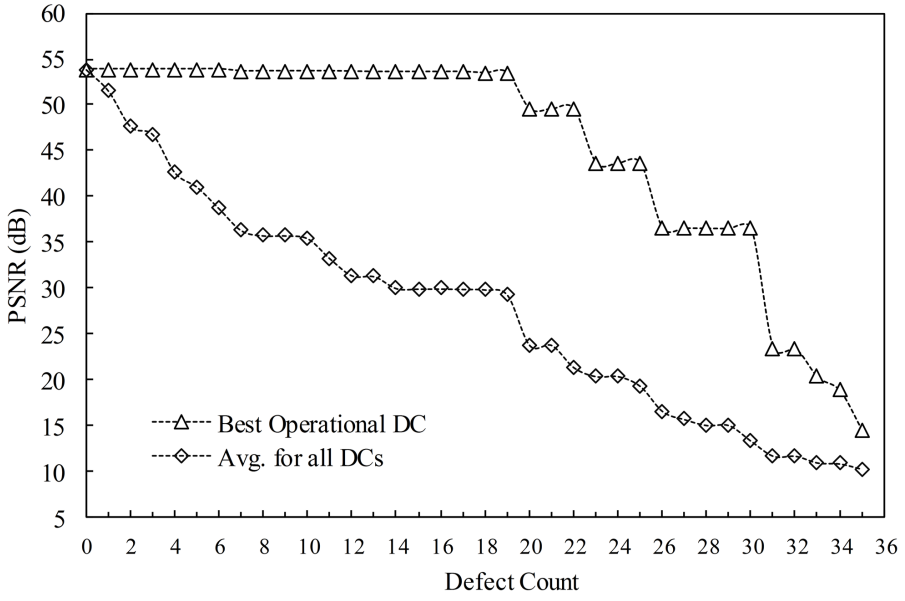


Figure 4.20: Average and top PSNR results for partial recovery.

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:

$$FR = \sum_{d=1}^{T} RC(d) \tag{4.9}$$

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}

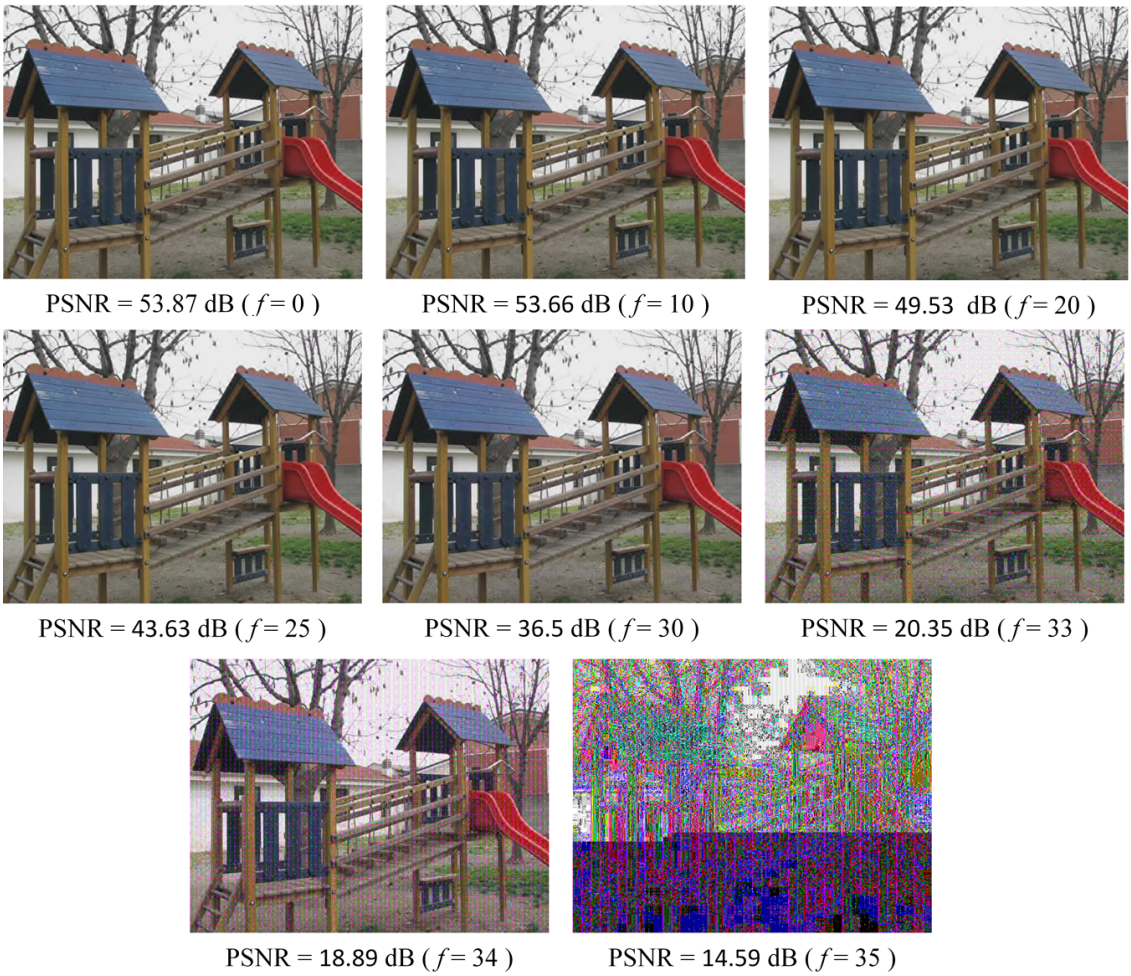


Figure 4.21: Partial recovery results on a test image.

is computed as follows:

$$RC_{NMR}(d) = \frac{|\{x \in \Gamma_d^T \; s.t. \; m_f \leqslant \lfloor \frac{N-1}{2} \rfloor\}|}{|\Gamma_d^T|} \tag{4.10}$$

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 $\left[10 \cdot |\Gamma_d^{2A_m}| - 15 \cdot |\Gamma_d^{A_m}|\right] /|\Gamma_d^T|$, respectively. Fig. 4.22(a) 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. (4.7) and thus improves FR.

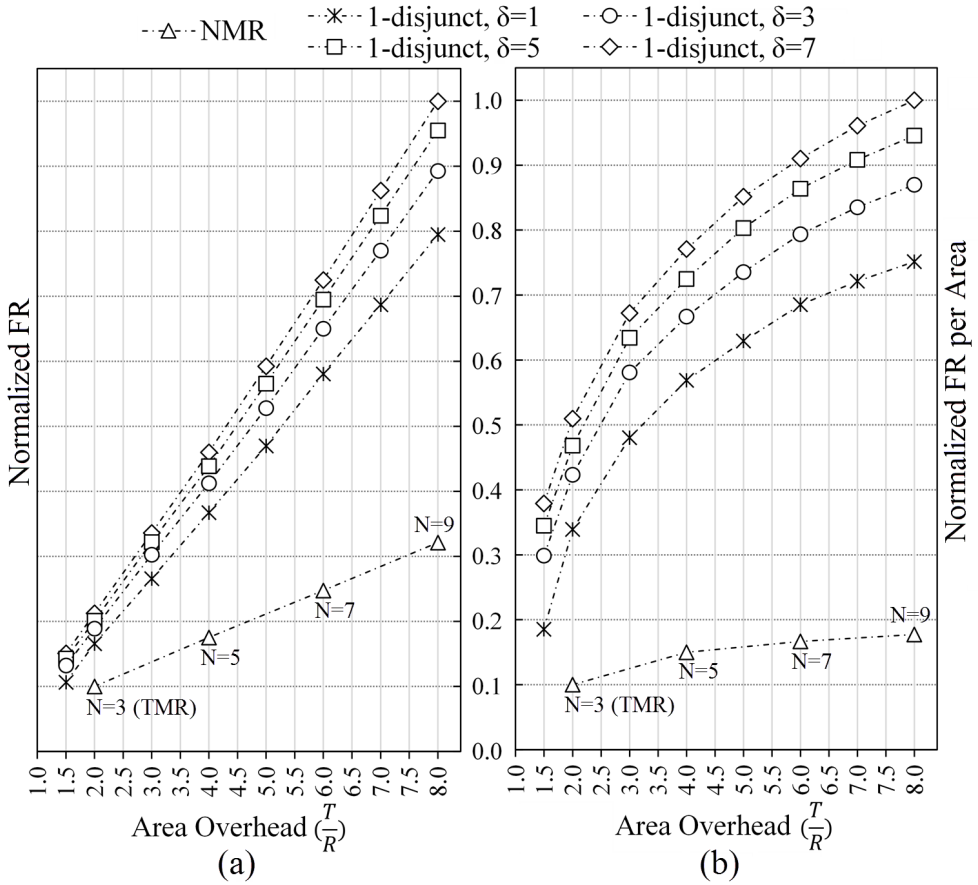


Figure 4.22: Area efficiency of design disjunction.

As depicted in Fig. 4.22(a), 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% (143.3%) 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. 4.22(b) 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 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 [51]. The reconfiguration structure is not limited to ICAP. For instance, Xilinx has recently introduced processor configuration access port (PCAP) interface [52] 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).

4.11 Summary

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% (94.4%) 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.

CHAPTER 5: HYPERGRAPH-COVER DIVERSITY FOR MAXIMALLY-RESILIENT RECONFIGURABLE SYSTEMS

Scaling trends of reconfigurable hardware (RH) and their design flexibility have proliferated their use in dependability-critical embedded applications. Although their reconfigurability can enable significant fault tolerance, due to the complexity of execution time in their design flow, in-field reconfigurability can be infeasible and thus limit such potential. This need is addressed by developing a graph and set theoretic approach, named *hypergraph-cover diversity* (HCD), as a preemptive design technique to shift the dominant costs of resiliency to design-time. In particular, *union-free hypergraphs* are exploited to partition the reconfigurable resources pool into highly separable subsets of resources, each of which can be utilized by the same synthesized application netlist. The diverse implementations provide reconfiguration-based resilience throughout the system lifetime while avoiding the significant overheads associated with runtime placement and routing phases. Two novel scalable algorithms to construct union-free hypergraphs are proposed and described. Evaluation on a Motion-JPEG image compression core using a Xilinx 7-series-based FPGA hardware platform demonstrates a statistically significant increase in fault tolerance and area efficiency when using proposed work compared to commonly-used modular redundancy approaches [53].

5.1 Introduction

The exponential growth in number of switching devices in very-large-scale integration (VLSI) designs dictated by Moore's law has rapidly increased the likelihood of fault occurrence in hardware systems. This challenge is aggravated further by the advent of nanofabrication

technology which introduced unprecedented reliability issues. Thus, *fault tolerance* (FT) techniques for computing systems have received a considerable attention in recent years. Traditionally, FT techniques have relied on passive modular redundancy which have a limited benefit per unit area and power. For many critical embedded systems such as those used in space and avionics platforms, unmanned aerial vehicles (UAVs), land and ocean-based remote sensing units (RSUs), environmental stress and conditions can result in a failure rate beyond what fault-handling capability of passive approaches can resolve. For such applications, immediate hands-on maintenance and repair is infeasible and hence a duly deployed autonomous method which caters to fault tolerance using available healthy resources is crucial for maintaining reliable long-term operation.

With the rise of *reconfigurable hardware* (RH) over the last two decades, in-field reconfigurability has opened up new possibilities to incorporate pseudo-intelligent FT attributes such as self-repair and autonomous fault recovery [8]. Such attributes are key enablers for efficient and sustainable fault-tolerant systems. RH is expected to have an essential role in designing future dependable embedded systems [2]. Unfortunately, exploiting design flexibility of modern RH for runtime FT is encumbered by the heuristic nature and increasing complexity of design placement and routing mechanisms. *SRAM-based field programmable gate arrays* (FPGAs) being the prominent example of RH can exemplify this challenge. Execution of a design flow targeting an SRAM-based FPGA can take an order of minutes to hours using a high-end multi-processing machine [9]. For low-performance fabric-embedded cores, the computational and energy constraints to execute in-field design reroute can be prohibitive [7]. We emphasize our observation here based on the current state of computer-aided design (CAD) tools used with available commercial off-the-shelf (COTS) reconfigurable components due to the increasing trend in using COTS-based embedded systems [10] [11].

In light of this major concern, design-time FT strategies that minimize reliance on runtime execution of design flow while can easily be integrated with existing vendors tools become more favorable. More specifically, the dominant implementation cost of reconfigurability feature can be mitigated by preparing an optimal set of design alternatives at design phase that properly cover the solution space for reliability exposures at runtime. In this work, *hypergraph-cover diversity* (HCD) approach based on *graph and set theory* is proposed to attain this objective for the FT coverage problem on embedded reconfigurable fabric. HCD method exploits the strong notion of *separability* [17] obtained by *union-free hypergraphs* [18] to model resources allocation among distinct design alternatives for highly diverse and fault resilient designs.

The remainder of the paper is organized as follows. Section II provides background and summarizes related work. Section III describes the graph model for proposed work. Section IV discuses the case study adopted for evaluation and experimental results. Finally, conclusions are given in Section V.

5.2 Background and Related Work

FPGA-based embedded systems have become ubiquitous computing platforms owing to the increasing embedded system functionality and the low non-recurring engineering costs (NREs) of embedded processor development using FPGAs. A typical FPGA-based embedded system may comprise application specific integrated circuits (ASICs) blocks, e.g. digital signal processors (DSPs), peripheral interfaces, memory controllers to interface with external non-volatile and main memory, a system bus, and a reconfigurable fabric tightly coupled to a general purpose processor (GPP). The reconfigurable fabric can act as a general hardware accelerator for performance-critical functions or as flexible design circuitry to apply runtime

changes such as protocol extensions, bug fixes, or advanced features to existing implementation [54]. Resources in the reconfigurable fabric are organized in a regular 2D array of identical tiles. Each tile includes a single configurable logic blocks (CLBs) as well as switching and connection blocks to facilitate inter and intra CLB connections. A CLB is also referred to as a logic array block (LABs) depending on FPGA device vendor. Each CLB is a group of identical programmable logical cells called slices. Each slice can have several look-up tables (LUTs), flip-flops (FFs), multiplexers, carry chains, and dedicated gates for combining LUTs to realize more complex Boolean functions.

5.2.1 Previous Work on FT on Reconfigurable Hardware

Regularity and logic density of contemporary reconfigurable architectures are particularity well suited for provision of runtime FT. A great deal of research has been conducted in the area of FT of reconfigurable hardware [55] [56]. Most FT approaches that exploit runtime reconfiguration for fault recovery can be classified into *covering approaches* or *embedding approaches* [57].

In the *covering approaches*, a set of spare resources are predefined and made available to replace faulty elements [8] [28]. The reconfiguration problem is to find the optimal assignment of spares to faulty resources such that large combinations of faulty resources can be covered. In [8], assignment of spare columns of resources is proposed for single-fault tolerance. A faulty column is avoided by shifting the column-based design implementation to a different set of healthy columns. Since number of spare assignments is low, they are implemented at design-time and used during system lifetime to provide low-overhead fault recovery. A single faulty resource in a column renders the whole column unusable and hence this method can result in a low area efficiency. The work in [28] distributes locations of individual spare

resources evenly across the fabric boundary. Distance-based evaluation score is used to define spare assignment at runtime and incremental runtime re-routing is required to achieve fault recovery. Although, this technique can cover a large set of multiple faults, its practicality can be very challenging given the complexity and routing overhead of contemporary FPGAs.

Alternatively, *embedding approaches* make no distinction between spare and normal resources. A system should have more resources than what is required to implement an application. Fault tolerance is achieved by remapping (embedding) fault-effected design into (in) remaining healthy resources. The challenge in this approach is to define a minimal set of alternative designs that achieve the target level of fault tolerance. A heuristic search algorithm to generate a set of diverse designs for single-fault tolerance at the CLB level was recently proposed in [36]. To the best of our knowledge, no formal technique based on a theoretical concept has been proposed as an embedding approach to real-time FT in reconfigurable systems. The work in [58]

In this paper, we exploit the notion of set separability given by the union-free hypergraphs [18] to identify highly diverse and fault resilient designs. A different hypergraph model was previously used in [59] to define a FT connection topology as a yield enhancement technique for processor arrays. Hypergraph was also used to study the spare assignment problem in the covering approach of FT processor arrays [60].

The following subsection will describe graph model developed herein with examples.

5.2.2 Union-free Hypergraphs

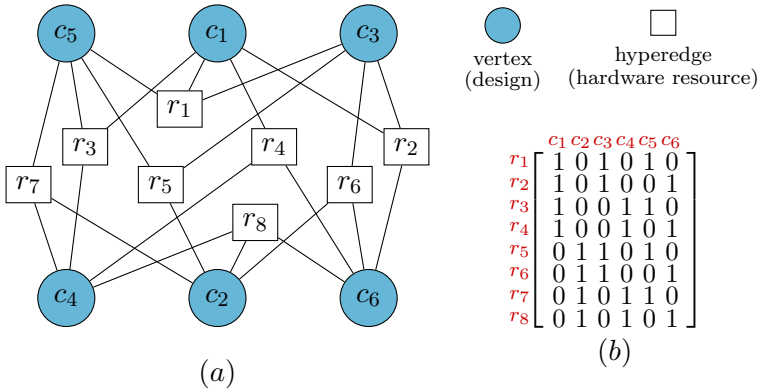
A hypergraph $H = (V\ ,\ E_h)$ can be described as a generalization of a graph in which edges E_h, or hyperedges, can connect any number of vertices. For the interest of this work, resources

will be represented by hyperedges and vertices are considered the distinct designs. Thus, a hypergraph in this representation defines how resources (hyperedges) are allocated to (connect to) different designs (vertices). Each hyperedge can be expressed by the subset of vertices it connects to. In hypergraph theory, a highly strong notion of set separability is described by a class of hypergraphs known as union-free hypergraphs. Consider he_x, he_y, and he_z to be three distinct hyperedges in H. Based on the original definition of union-free property [61], H is union-free if:

$$\forall\ he_x, he_y, he_z \in E_h\ ,\quad he_x \cup he_y \neq\ he_z \tag{5.1}$$

This definition implies that there are no two distinct hyperedges (resources) in H connected to (utilized by) the same set of vertices (designs). Figure 5.1(a) shows an example of a union-free hypergraph with 6 vertices and 8 hyperedges and its incidence matrix in Figure 5.1(b). Columns of the incidence matrix represent hyperedges (resources) and rows represent vertices (designs). An element x_{ij} of the incidence matrix is one if hyperedge i connects to vertex j. Therefore, the number of one elements in each column indicates the subset of resources used by each design.

We propose a systematic way for constructing such a union-free hypergraph using trivial binary matrix manipulations. Given that embedding approaches require more resources than what are needed for design implementation, *resrouce utilization ratio U* is expected to be less than 1. In the next section, we describe two novel algorithms to construct the described hypergraph model for utilization ratio $U = 1/2$ and $U = 2/3$. These target values result in a low area overhead compared to commonly-used modular redundancy schemes, e.g. Triple Modular Redundancy (TMR).



(a) (b)

Figure 5.1: Example of a (a) Union-free hypergraph. (b) Its incidence matrix.

5.3 Hypergraph-Cover Diversity

To construct a union-free hypergraph H suitable to generate resilient designs, two properties have to be maintained:

- **First**, condition (1) must be satisfied.
- **Second**, H is a regular hypergraph[1] with a degree[2] $\gamma(H) = A$, where A is the minimum number of resources required to implement the application.

We propose the first algorithm, Algorithm 5.3.1, for $U = \frac{1}{2}$ which achieves union-free property using only $\lceil 2 \cdot log_2(2 \cdot A) \rceil$ designs. For the sake of clarity, the incidence matrix is used for the illustration of proposed algorithm. The corresponding steps in the graph domain are given in the pseudo-code. As an example, assume the size of the target application is $A = 6$ FPGA slices, the algorithm starts with a zero incidence matrix having a number of rows $r = 2^x$, where $x = \lceil log_2(2 \cdot A) \rceil$, thus $r = 16$, and a number of columns equals

[1] A *regular* hypergraph is a graph where each vertex has the same number of edges

[2] The *degree* of a regular graph is the number of edges connected to each vertex

$c = 2 \cdot x$, hence $c = 8$, (lines 1 through 7 of Algorithm 5.3.1). For odd-indexed columns $(1, 3, ..., r-1)$, the elements indicated by 'one' are distributed according to the binary tree pattern depicted in Figure 5.2(a). Each of the even-indexed columns $(2, ..., r)$ is obtained by simply complementing its immediate lower-indexed column. These steps are more formally defined in lines 10 through 21 of the pseudo-code.

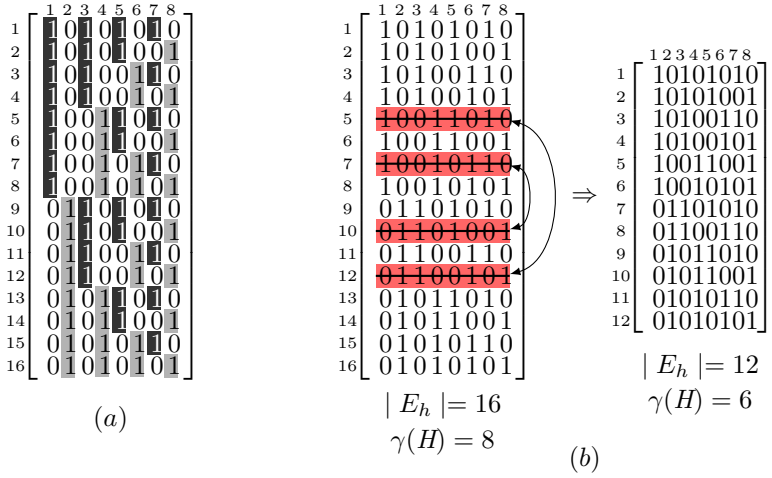


Figure 5.2: (a) Constructing union-free hypergraph using its incidence matrix. (b) Reducing degree of hypergraph by deleting disjoint hyperedges.

The hypergraph given by resultant incidence matrix should satisfy condition (1). A further step is needed to reduce the degree of the graph to $\gamma(H) = A$. We observe that complimentary rows (or disjoint hyperedges) can be deleted without violating condition (1). Deleting a pair of disjoint hyperedges will decrement $\gamma(H)$ by 1 as shown in Figure 5.2(b). This step is repeated until $\gamma(H) = A$ (lines 22 through 28 of Algorithm 5.3.1).

Algorithm 5.3.1: Algorithm for Constructing Union-free Hypergraph for $U = 1/2$

procedure construct hypergraph $H = (V\ ,\ E_h)$

Input: A: Resource Count Required to Implement Application

Output: Hypergraph $H = (V\ ,\ E_h)$

```
1  V := ∅ // set of vertices
2  E_h := ∅ // set of hyperedges
3  x := ⌈log_2(2 · A)⌉
4  r := 2^x // initial number of hyperedges
5  c := 2 · x // initial number of vertices
6  V := {v_1, v_2, ...., v_c} // initialize V with c vertices
7  E_h := {he_1, he_2, ...., he_r} // initialize E_h with r hyperedges
8  k := 0 // level index in the binary tree
9  c_index := 1 // index to loop through each pair of vertices
10 while (c_index < c) do
11     p := 2^k // parameter to determine number leaves in each level of the
           binary tree
12     e := r/(2·p) // parameter to determine subset of hyperedges connected to each
           pair of vertices
13     i := 1 // index to loop through all r hyperedges
14     while (i < r) do
15         for m := i → (i + e) do
16             he_m := he_m ∪ {v_{2·k+1}} // inclusion of odd-indexed vertex
17         for m := i + e → (i + 2 · e) do
18             he_m := he_m ∪ {v_{2·k+2}} // inclusion of even-indexed vertex
19         i := i + 2 · e
20     k := k + 1
21     c_index := c_index + 2
   // reduce degree of hypergraph H by deleting disjoint hyperedges, one pair
       at a time
22 if γ(H) ≠ A then
23     for every he_x, he_y ∈ E_h | x ≠ y do
24         if he_x ∩ he_y = ∅ then
25             remove he_x and he_y from H
26             if γ(H) = A then
27                 return H
28                 end
```

In a similar manner, an algorithm can be devised for hardware utilization $U = \frac{2}{3}$. Algorithm 5.3.2 achieves union-free resource assignments for $\lceil 3 \cdot log_3(\frac{3}{2} \cdot A) \rceil$ designs. Assuming $A = 5$, the algorithm starts with a zero incidence matrix having a number or rows $r = 3^x$, where $x = \lceil log_3(\frac{3}{2} \cdot A) \rceil$, thus $r = 9$, and a number of columns equals $c = 3 \cdot x = 6$, as given in lines 1 through 7 of Algorithm 5.3.2.

The distribution of the one-encoded elements in the incidence matrix follows a ternary tree pattern among three groups of columns: (1,4,7,..,etc), (2,5,8,.., etc), and (3,6,9, .., etc). Since $U = 2/3$, two thirds of each column's elements must be one. As shown in Figure 5.3(a), the order of those thirds, from top to bottom, is (1^{st}, 2^{nd}), (1^{st}, 3^{rd}), and (2^{st}, 3^{rd}) for the three groups of columns, respectively (lines 10 through 23 of Algorithm 5.3.2). This arrangement results in a union-free hypergraph in which $\gamma(H) = \frac{2}{3} \cdot r$.

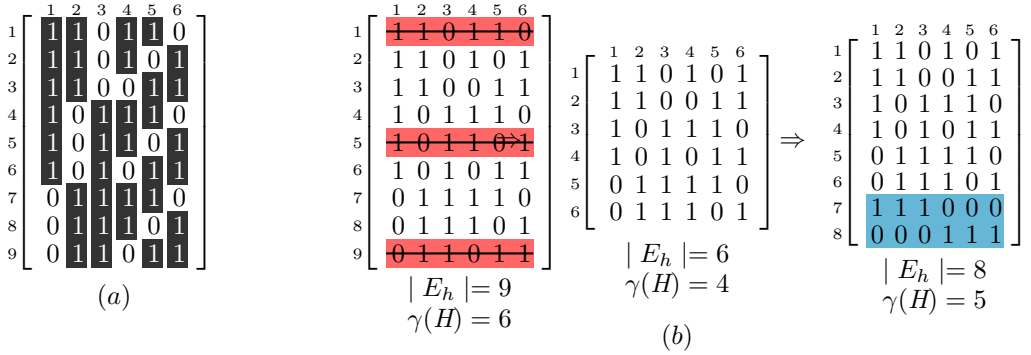


Figure 5.3: (a) Constructing hypergraph using its incidence matrix. (b) Adjusting degree of hypergraph to fit target application size.

By removing hyperedges he_x, he_y, he_z such that $|he_x \cap he_y| = |he_x \cap he_z| = |he_y \cap he_z| = 2$, $\gamma(H)$ is decremented by 2 while preserving the union-free property (lines 24 through 30). If A is odd number, a last step to pad the incidence matrix is necessary to reach $\gamma(H) = A$.

Padding is conducted by simply adding a pair of disjoint hyperedges (lines 31 through 36 of Algorithm 5.3.2) or complementary rows as shown in Figure 5.3(b) to attain $A = 5$.

A salient attribute of the proposed HCD technique that is crucial to its practicality and effectiveness is the ability to swap all hyperedges in the hypergraph without violating condition (1) or degrading the achieved separability of resultant designs. Figure 5.4 depicts how resources are assigned to diverse designs using a union-free hypergraph for a reconfigurable region of size $9 \cdot 9 = 81$ and application size $A = 54$. Algorithm 5.3.2 is used in this example to define 12 separable designs. The 81 resources are ordered from left to right and up to down. Figure 5.4(a) shows arrangement of resources defined by Algorithm 5.3.2 without changing the order of hyperedges. Figure 5.4(b) shows how this arrangement is mapped to an equivalent one by randomly swapping all hyperedges of the constructed hypergraph. Both arrangements exhibit the same resource separability dictated by the union-free property. This feature is important because routing congestion of tightly-closed resources (as in Designs 1,3,7, and 9 of Figure 5.4(a)) can lead to a failed routing. In addition, timing violations can occur if distant groups of resources (as in Designs 2 and 8 of Figure 5.4(a)) are used to implement design alternatives.

Algorithm 5.3.2: Algorithm for Constructing Union-free Hypergraph for $U = 2/3$

procedure construct hypergraph $H = (V\ ,\ E_h)$
Input: A: Resource Count Required to Implement Application
Output: Hypergraph $H = (V\ ,\ E_h)$

$V := \emptyset$ // set of vertices
$E_h := \emptyset$ // set of hyperedges
$x := \lceil log_3(\frac{3}{2} \cdot A) \rceil$
$r := 3^x$ // initial number of hyperedges
$c := 3 \cdot x$ // initial number of vertices
$V := \{v_1, v_2, ..., v_c\}$ // initial number of vertices
$E_h := \{he_1, he_2, ..., he_r\}$ // initialize E_h with r hyperedges
$k := 0$ // level index in the ternary tree
$c_{index} := 1$ // index to loop through each triad of vertices
while *($c_{index} < c$)* **do**
    $p := 3^k$ // parameter to determine number leaves in each level of the ternary tree
    $e := \frac{r}{3 \cdot p}$ // parameter to determine subset of hyperedges connected to each triad of vertices
    $i := 1$
    while *($i < r$)* **do**
        for $m := i \rightarrow (i + e)$ **do**
            $he_m := he_m \cup \{v_{3 \cdot k+1}, v_{3 \cdot k+2}\}$ // inclusion of 1^{st} & 2^{nd} vertices of each triad
        for $m := i + e \rightarrow (i + 2 \cdot e)$ **do**
            $he_m := he_m \cup \{v_{3 \cdot k+1}, v_{3 \cdot k+3}\}$ // inclusion of 1^{st} & 3^{rd} vertices of each triad
        for $m := i + 2 \cdot e \rightarrow (i + 3 \cdot e)$ **do**
            $he_m := he_m \cup \{v_{3 \cdot k+2}, v_{3 \cdot k+3}\}$ // inclusion of 2^{nd} & 3^{rd} vertices of each triad
        $i := i + 3 \cdot e$
    $k := k + 1$
    $c_{index} := c_{index} + 3$
// reduce degree of hypergraph H by deleting appropriate hyperedges, one triad at a time
if $\gamma(H) \neq A$ **then**
    for $he_x,\ he_y,\ he_z\ \in E_h \mid x \neq y \neq z$ **do**
        if $|he_x\ \cap\ he_y| = |he_x\ \cap\ he_z| = |he_y\ \cap\ he_z| = 2$ **then**
            remove he_x, he_y, and he_z from H
            if $\gamma(H) = A$ **then**
                return H
                end
            if $\gamma(H) < A$ **then**
                // add a one pair of disjoint hyperedges to attain $\gamma(H) = A$
                $he_1 := \{\lceil \frac{c}{2} \rceil$ random vertices from $V\}$
                $he_2 := \{v \mid v \notin he_1\}$
                add he_1 and he_2 to E_h
                return H
                end

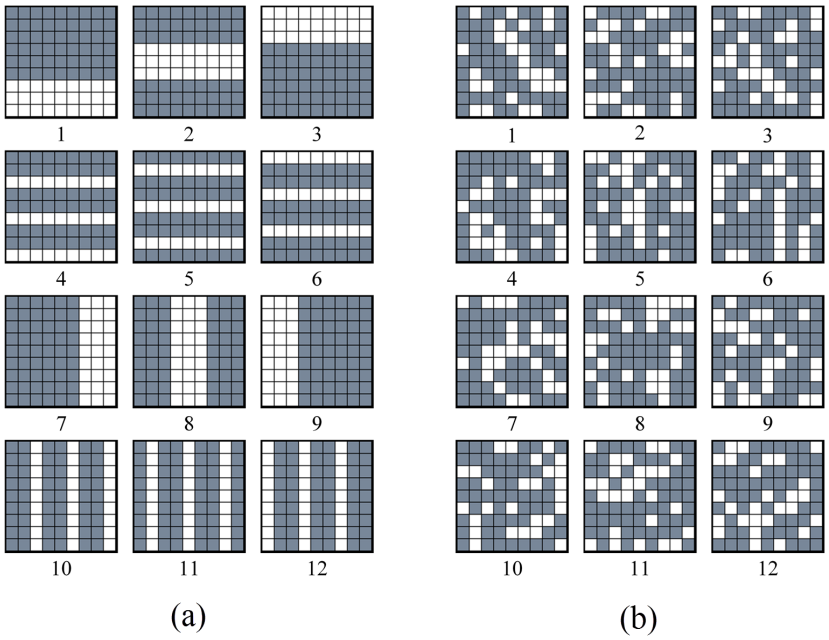


Figure 5.4: Example of two equivalent sets of separable HCD resource allocations on 2D array.

The number of distinct arrangements that can be constructed is intractably vast which allows CAD tools a large freedom for finding the performance and power consumption goals under proposed HCD design technique.

5.4 Evaluation

A Discrete Cosine Transform (DCT)-based Motion-JPEG (MJPEG) compression core was selected as a case study for this work due to its widespread use for image and video compression. Image and video compression applications are commonly used in space exploration missions, satellites, and monitoring systems in high radiation-dose and harsh environments. Such systems require rigid requirements in system reliability and durability. This includes

ability to operate for a long lifetime while providing adequate processing to meet increasing future demands. For instance, systems used in low-orbit satellites are expected to remain in service for up to 25 years [62]. A failure in an on-board system can render a mission objective obsolete. For survivable systems under these conditions where hands-on human maintenance is infeasible, a form of redundancy is mandatory to replace or mask failed components.

Evaluation of HCD method is conducted on an FPGA-based computing system using the commercial Xilinx KC705 FPGA evaluation board [63]. The KC705 board includes a 28nm Xilinx 7-series FPGA, DDR3 and FLASH memories. Xilinx Embedded Development Kit (EDK) and PlanAhead tools are used to generate the bitfiles for all of the implementations in this evaluation. Bitfiles are stored in the external flash memory before evaluation procedures begin. An embedded MicroBlaze soft processor is employed to control reconfiguration flow for autonomous fault recovery. We used the Xilinx Software Development Kit to develop the software module that runs on the embedded processor to reconfigure HCD designs onto the device and to control the FT flow. The processor can be protected against soft and hard faults using proper techniques such as TMR, rollback, check-pointing, and reconfiguration-based approaches [64] [65]. Partial reconfiguration using Xilinx Internal Configuration Access Port (ICAP) is utilized in this work for rapid reconfiguration.

The MJPEG core comprises different blocks, most of which are not computationally intensive and hence implemented by software processing using the embedded MicroBlaze processor. Contrarily, the 2D-DCT block of the MJPEG core carries out a high computational workload as reflected by the latency ratio of all blocks in the left stacked bar of Figure 4.17(a). To achieve balanced critical paths among MJPEG pipeline stages, it is favorable to implement DCT block as a hardware accelerator on reconfigurable fabric. Figure 4.17(b) shows the normalized latency of DCT Implementable using both software and hardware processing on the KC705 board. The right stacked bar of Figure 4.17(a) shows modified latency ratios of

MJPEG blocks after a DCT accelerator is implemented on the reconfigurable fabric. The proposed HCD scheme is applied to the DCT hardware accelerator. Resource assignments of HCD designs are adopted at the slice level and enforced using placement constraints applied by defining the placement `AREA_GROUP` and `CONFIG_PROHIBIT` statements in the User Constraints File (`.ucf`) used by the Place and Route (PAR) tools.

By using a quantifiable image quality metric such as the Peak Signal-to-Noise Ratio (PSNR), the need for hardware error detection can be eliminated, thereby a large saving in area and power is achieved [50]. PSNR is based on Mean Square Error (MSE) computed by averaging the intensity difference between pixels of recovered and original images. The recovered image is obtained by a fast software-based implementation of the Inverse Discrete Cosine Transform (IDCT). The input test image in this evaluation is a high-resolution satellite image placed in the external DDR3 memory.

Table 5.1: Design Parameters for Implemented TMR and HCD

Technique	Design Alternatives	Power Consumption (mW)		Resource Count (slices)	
Hypergraph-based Diversity	18	Avg.	24.11	Total	844
		Min.	23.29	Active Resources	422
		Max.	25.40		
TMR	1	71.01		Total	1350
				Active Resources	1270

In this evaluation, HCD is compared to the most frequently used FT technique, TMR, which is the conventional strategy used by the Xilinx X-TMR FT tool. Table 5.1 shows the resource count and power consumption of DCT accelerator for HCD and TMR implementations. 422 slices are required to implement a single DCT accelerator. Algorithm 5.3.1 is used to generate HCD alternatives for hardware utilization $U = \frac{1}{2}$; hence total number of resources

is 844 slices. Only 18 design alternatives were found sufficient to achieve union-free property. Figure 5.5 depicts a screen-shot of the PlanAhead tool layout of the Partial Reconfiguration Region (PRR) for an HCD implementation of DCT accelerator.

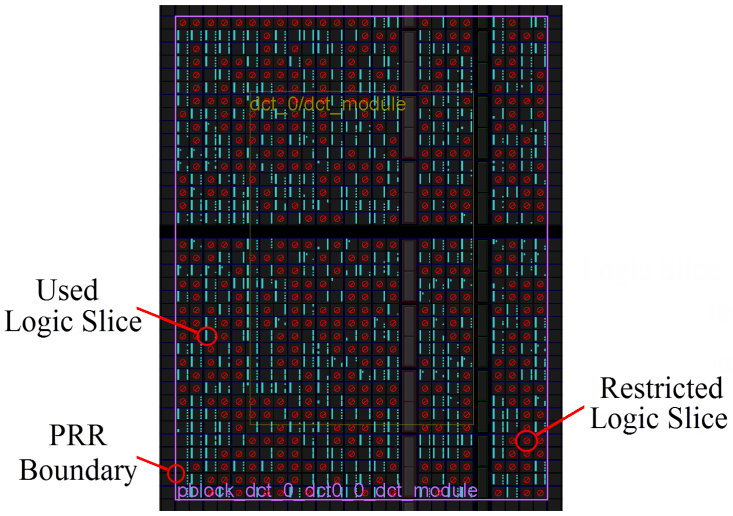


Figure 5.5: PlanAhead layout for a aingle HCD design alternative.

Since FPGA routing resources are configured during routing phase to connect logic slices to realize the structure of target design, we would expect a low overlap for routing resources among separable designs. Figure 5.6 shows the proportion of all used Programmable Interconnect Points (PIP) according to their utilization count among HCD designs for the DCT implementation. We observe an steep exponential decline in the number of PIPs that are utilized by more than one design. This indicates fault resilience of HCD method against routing failures as evaluated herein.

For TMR implementation, three replicas of DCT accelerator with a voting logic are realized using $3 \cdot 422 + 4$ (for voter) $= 1270$ slices. For each replica extra unused slices are required for successful routing. This issue does not affect the implementation of HCD designs since half of the total resources are unused.

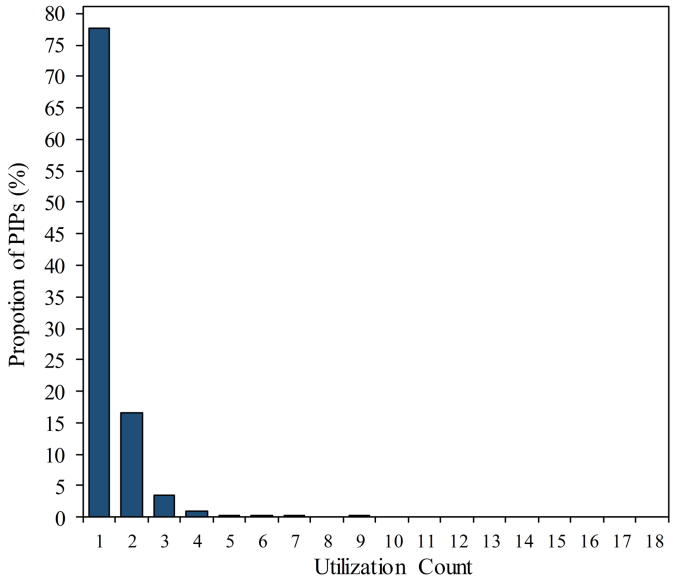


Figure 5.6: PIPs usage overlap among HCD designs.

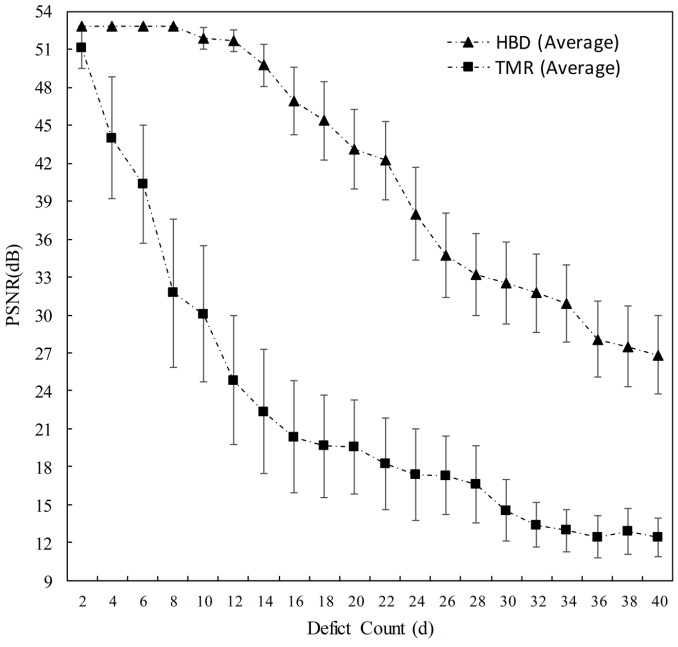


Figure 5.7: PSNR results for proposed HCD and TMR.

To compare effectiveness of fault tolerance of HCD and TMR, up to 40 Stuck-at Faults (SAFs) at LUTs input terminals are randomly injected with a uniform distribution on each experimental run. Input terminals are chosen to exercise both logic and routing faults. Figure 5.7 shows retrieved PSNR values over 25 experimental for HCD and TMR schemes. Due to the large set of possible combinations of distinct faults, the upper and lower whiskers at each point indicate the 95% confidence interval of the results. The PSNR values of HCD represent the highest PSNR values achieved by any of the 18 design alternatives. The proposed FT scheme maintains substantially a higher image quality throughout the evaluation. It is observed that image quality of TMR design deteriorates quickly for a defect count d less than 14. HCD scheme maintains roughly twice the image quality of TMR counterpart for d higher than 10.

Figure 5.8 and Figure 5.9 show output JPEG images for an experimental run using proposed HCD and TMR methods, respectively. The results here reveal the considerable advantage of HCD to tolerate faults compared to modular redundancy. The effect of hardware defects on image quality of TMR design can be apparent to a human perception for $d > 8$. In the case of the proposed scheme, the image quality differences are hardly distinguishable even for $d = 40$.

Table 5.1 also indicates the area and power measurements of the two implementations. The proposed scheme results in a total resource saving of 37.5%. The average power consumption of HCD design alternatives is 24.11 mW. Compared to TMR, HCD results in a power saving of 66%. This is attributed to the fact that only one implementation of DCT accelerator is active at a time as opposed to the three replicas of the TMR scheme.

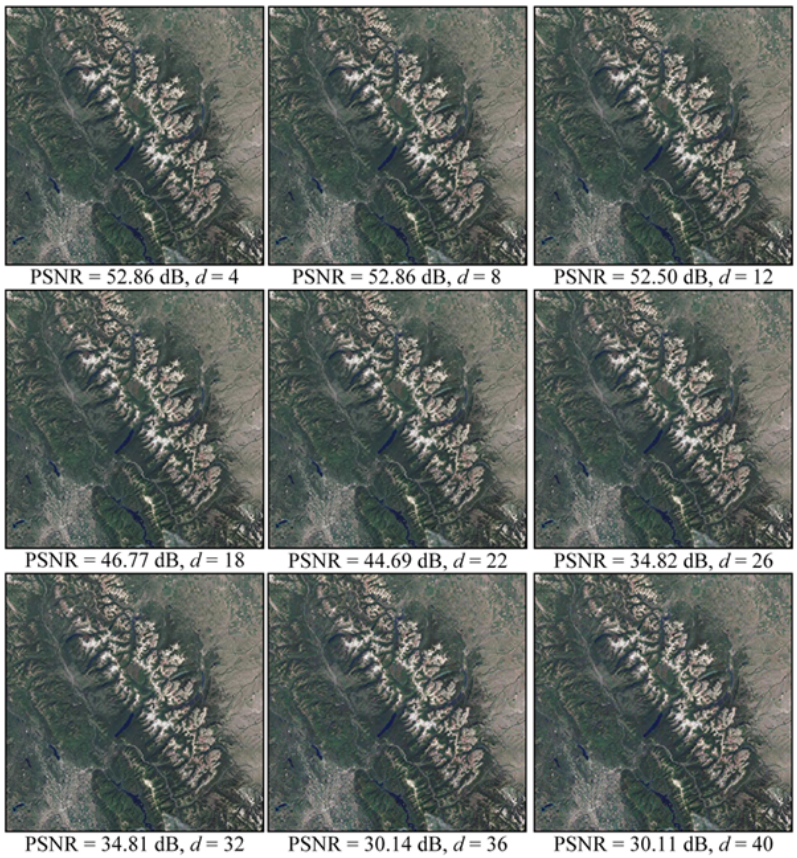


Figure 5.8: Image quality under varying defect count for HCD.

5.5 Conclusion

In this work, we show that set separability defined by hypergraph theory can be used effectively to create highly resilient designs at design-time for provision of low-overhead fault recovery during system lifetime. Systematic algorithms to construct design diversity using union-free hypergraph model for different target hardware utilization values are also presented. Results have demonstrated the potential of the proposed FT method to achieve 37.5% area saving and up to 66% reduction in power consumption compared to the frequently-used TMR scheme while providing superior fault tolerance.

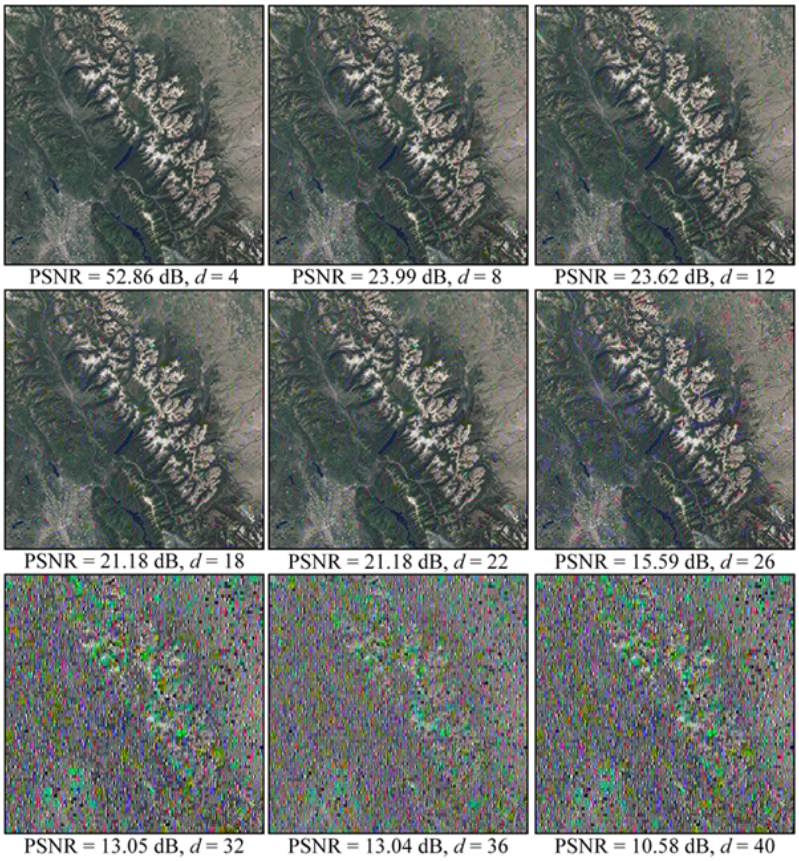


Figure 5.9: Image quality under varying defect count for TMR.

CHAPTER 6: PROCESS VARIATION IMMUNITY OF ALTERNATIVE 16NM HK/MG-BASED FPGA LOGIC BLOCKS

Continued miniaturization of semiconductor technology to nanoscale dimensions has elevated reliability challenges of high density Field-Programmable Gate Arrays (FPGA) devices due to increasing impacts of Process Variation (PV). The issue is addressed herein using a systematic bottom-up analysis by determining the relative influence of PV on alternate design realizations of FPGA logic blocks. Results for conventional design structures are obtained through detailed SPICE simulations and related to structural risk features. Namely, Transmission Gate (TG) and Pass Transistor (PT) based MUX architectures for realizing Look-Up-Tables (LUTs) are compared. At threshold voltage variation $\sigma_{V_{th}} = 14\%$, PT-based designs that meet the 95% yield objective can exhibit as high delay variation as 23.3%. PV impact can be reduced to 4.9% if TG-based LUT is considered. Finally, the impact of transistor sizing is investigated as a method of mitigating PV susceptibility in FPGA structures [66].

6.1 Introduction

Advancement of CMOS manufacturing technology to reduce device dimensions has ushered in significant challenges resulting from *Process Variation (PV)* [67]. Significant sources of variation in sub-45nm manufacturing processes include imprecise lithography, etching, deposition, and dopant implantation [68]. These can lead to Random Dopant Fluctuation, Line-Edge Roughness, and structure dimension variance, e.g. channel length and oxide

thickness. Variation in these physical parameters translates into deviation in device electric characteristics, such as V_{th} and drive current I_{dsat}, from the intended specifications. Therefore, PV can lead to slow, weak, or defective transistors, thus affecting yield, final product performance, efficiency, and reliability. The International Technology Roadmap for Semiconductors (ITRS) has estimated that V_{th} variation, given by three-sigma ($3\sigma_{V_{th}}$), has already reached 42% ($\sigma_{V_{th}} = 14\%$) and can reach up to 79% ($\sigma_{V_{th}} = 26\%$) for near-future process technology, according to table DESN10 in [69]. Fortunately, PV exhibits a statistical nature which makes it feasible to study at various levels of design abstraction, which are compared in this paper for alternate functional realizations. Traditionally, Statistical Static Timing Analysis (SSTA) technique is used to predict design behavior at design-time and accordingly devise the appropriate mitigation strategies to minimize PV effects and increase yield. Traditionally, Statistical Static Timing Analysis (SSTA) technique is used to predict design behavior at design-time and accordingly devise the appropriate mitigation strategies to minimize PV effects and increase yield.

SRAM-based *Field Programmable Gate Arrays (FPGAs)* have been at the frontier of technology scaling owing to the increased demand for high performance and low-power reconfigurable systems. Thus, the design of FPGA logic blocks have an increasing need to cope with PV issues emerging at each new process node. Contemporary SRAM-based FPGAs designs are composed of array of tiles, which contain Logic Clusters (LCs), Connection Boxes (CBs), and Switching Box (SBs). A logic cluster contains Look-Up-Tables (LUTs) and flip-flops which implement logic functionality. Connection and switching boxes provide the required connectivity among LCs and routing channels. Commercial FPGAs have utilized multiplexers (MUXes) to implement LUTs, CBs, and SBs due to the lower required number of control inputs and favorable area-delay product [70] [71]. Because of the uniform fabric of modern SRAM-based FPGAs, multiplexers can be viewed as a dominating fundamental logic struc-

ture in these devices besides SRAM cells. To reduce cost and area, FPGA vendors have relied on NMOS Pass-Transistors (PTs) as the preferred fundamental switching elements for realizing multiplexers. NMOS PTs are known for conveying a weak high logic signal level at a saturated output of $V_{DD} - V_{th}$. As aggressive scaling of devices continues, the voltage difference between V_{DD} and V_{th} decreases; thus half-latch restoration logic has been used to mitigate resultant reliability and performance issues. Recently, the work in [70] investigated the use of Transmission Gates (TGs) as an alternative design option to implement FPGAs blocks while achieving lower area-delay product. In [72], PV-induced failure rate in a PT-based multiplexer without the restoration logic is studied. However, the work did not consider that transistor sizing can substantially reduce defect rate as demonstrated later in this work. To the best of our knowledge, there has not been a study on how PT and TG-based structures compare as design options for FPGA structures under the effect of variations. In this paper, we study the impact of variation on these two design alternatives and report the *Defective Rate*, *Delay*, and *Energy Delay Product (EDP)*.

The remainder of the paper is organized as follows. Section 2 of the chapter provides a background on PV and the FPGA structures to be considered for the evaluation. Section 3 describes the evaluation framework and toolset adopted for simulation. Results and Conclusions are discussed in Section 4.

6.2 Effects of Process Variation

Process variation is a key challenge for continued technology scaling. It can affect functional, leakage, or timing yield and power efficiency of final design due to the need for wider voltage margins. Variation can include any nanoscopic imprecision in manufacturing processes during physical realization of design layout. PV can be manifested as Die-to-Die (D2D) or

WithIn-Die (WID) variations. WID variation becomes a significant factor in the impact of variation and thus is the scope of this work. At the individual transistor-level, the prominent negative effect of WID variation is observed as a variation in device threshold voltage and the amount of current flow during transistor ON-OFF states. Threshold voltage variation $\sigma_{V_{th}}$ is essentially a function of device dimensions and dopant density in the channel as expressed below [73].

$$\sigma_{V_{th}} \propto \frac{t_{ox}}{\epsilon_{ox}} \sqrt{\frac{n_{ch}}{3 \cdot w \cdot l}} \tag{6.1}$$

where w and l are the channel width and length respectively, t_{ox} is the gate oxide thickness, ϵ_{ox} is the permittivity of oxide layer, and n_{ch} is the concentration of channel doping. Since device delay t_g is tightly dependent on V_{th} as given by the well-cited *alpha-power law* in (6.2), high variation can severely impact transistor speed and cause timing yield loss.

$$t_g \propto \frac{l_{eff} \cdot V_{DD}}{(V_{DD} - V_{th})^{\alpha}} \tag{6.2}$$

where l_{eff} is the effective channel length, α is a constant depending on process technology. Similarly, threshold voltage affects the transistor ON saturation current $I_{D_{sat}}$ as given in (3) which results in a weak or slow driving transistor.

$$I_{D_{sat}} = \frac{w \cdot \nu_{sat} \cdot \epsilon_{ox}}{t_{ox}} \cdot (V_{gs} - V_{th} - V_{d_{sat}}) \tag{6.3}$$

where V_{gs} is the gate voltage, and $V_{d_{sat}}$ is the saturation drain voltage. This effect leads to a situation where a transistor is either not a strong enough to trigger downstream gates, or does so at a slow pace causing higher sub-threshold current to flow in fanout gates. Thus, PV can cause a functional yield loss and power constraint violation even if timing

requirements determined by the proportion of gates in the critical path are met. In this paper, we consider the case where PV can cause FPGA structures to functionally fail. Previous work for the effects of PV in FPGAs have considered the timing and leakage yield [74] [75], whereas in [72] functional failure is studied for a single MUX design without considering other design alternatives or device-level mitigation strategies such as transistor sizing to combat variation. FPGA group testing studied for hard faults can offer an emerging alternative [34].

The key logic structures for realizing SRAM-based FPGAs are SRAM cells and MUXes. Due to their ubiquitous applications, variation in SRAM cells has been extensively studied [76]. In the case of FPGAs, the effect of PV on SRAM cells is less limited since SRAM cells are not packed in an array structure in the same organization used in other custom VLSI designs, e.g. cache memory. In addition, SRAM cells in FPGAs are not often written; thus, the overhead imposed by any deployed technique to avoid PV-induced write failures can be negligible. To that end, we focus on MUX-based structures and consider two commonly accepted design options for implementing them in FPGAs [77]. Namely, the effect of variation on PT-based MUXes with half-latches and TG-based counterparts are compared.

6.2.1 Pass Transistor-based Multiplexers with Half-latch

Figure 6.1(a) shows an example of 2:1 PT-based multiplexer. Two NMOS pass transistors t_0 and t_1 with complementary control inputs are used to select which input signal to pass to the multiplexer output. Due to their higher mobility, NMOS transistors are favored for relative driving strength over PMOS transistors. Since NMOS transistor passes a weak high logic level with a voltage swing ranging from 0 to $V_{DD} - V_{th}$, some restoration logic, or *half-latch*, is required to pull-up the weak-1 output to recover a strong-1 level. The half-latch is an inverter with a pull-up transistor t_r controlled by the inverter output. When

a weak-1 is propagated to the inverter input, inverter output transition to a low voltage will activate the pull-up transistor t_r to boost the inverter input to a strong-1 for a stable operation. The restoration logic can be placed after every two or more cascaded PTs in a large multiplexer, as depicted in Figure 6.1(b), to reduce area and latency overhead at the expense of less reliable signal propagation. High variation in a PT-based MUX can lead to a low $V_{DD} - V_{th}$ voltage difference insufficient to trigger the inverter, precipitating functional failure as demonstrated in Section IV.

6.2.2 Transmission Gate-based Multiplexers

A transmission gate is composed of NMOS and PMOS transistors connected in parallel to avoid the issue of passing a weak-1 or weak-0 at the expense of the added area of a PMOS transistor. The two transistors are controlled by two complementary signals to activate both during TG transparent state. Figure 6.2(a) depicts the structure of a 2:1 TG-based MUX. Two TGs t_{g0} and t_{g1} are connected to complimentary signals to select one of the two inputs. Restoration logic is not needed as TG can pass both strong-0 and strong-1.

6.3 Evaluation Framework

To facilitate valid comparisons, a fracturable 6-input LUT that can be utilized as 6-input or 5-input LUT is adopted as a case study. 6-input LUTs are considered the optimal size in terms of area-delay product [78] and are also used in latest commercial FPGAs, e.g. Xilinx Virtex 7 and Altera Stratix V. Design and simulation for this evaluation are based on the High-K Metal-Gate (HK/MG) 16nm Predictive Technology Model (PTM) from Arizona State University. Moreover, the insight gathered from this case study can be generalized to

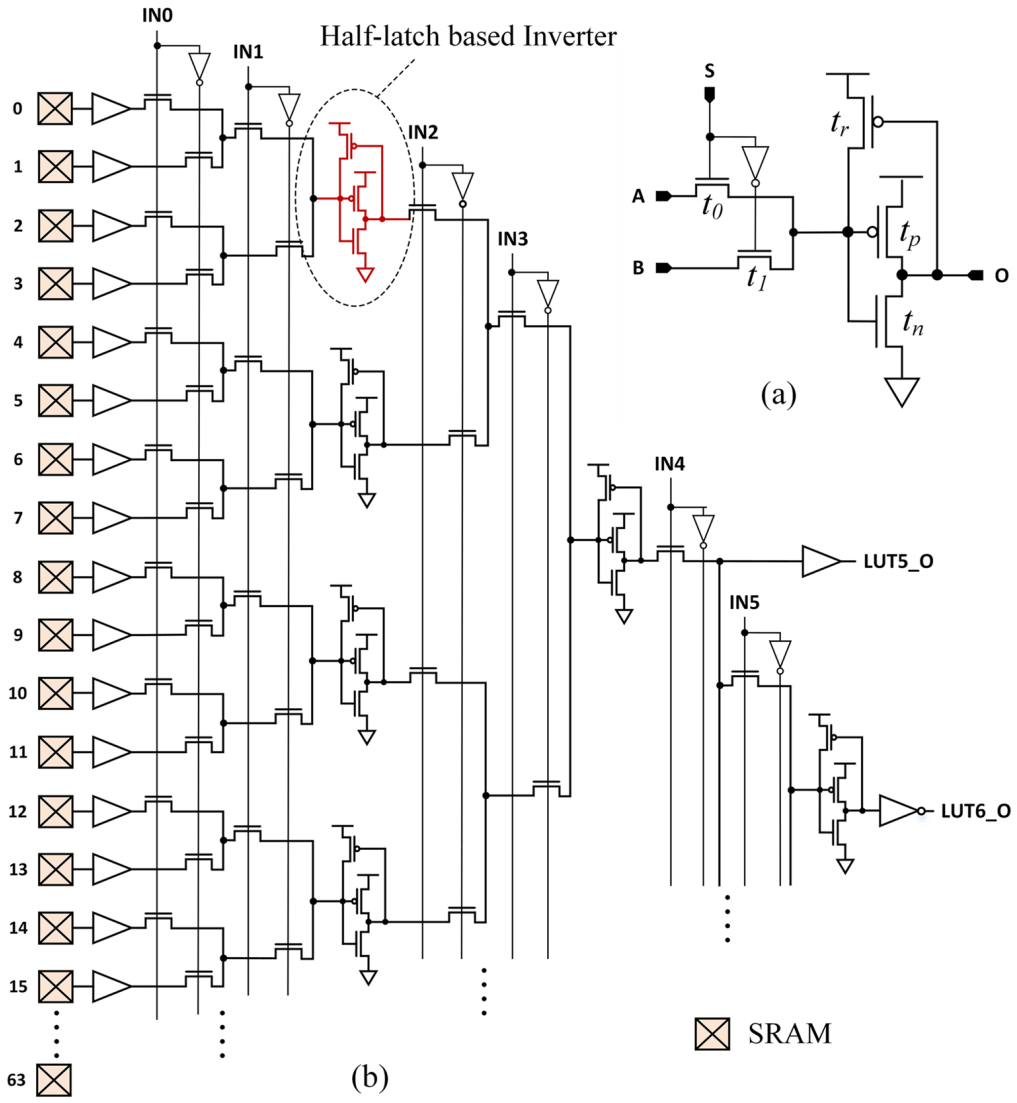


Figure 6.1: Netlist for (a) 2:1 PT-based MUX. (b) PT-based 6-input LUT (partial view).

other MUX-based FPGA structures. The 6-input LUT is implemented as a fully-encoded MUX tree using $2^6 - 1 = 63$ multiplexers. Figure 6.1(a) and Figure 6.2(a) depict a partial view of PT-based and TG-based implementation. Internal re-buffering with proper sizes are also used to maintain optimal latency. The baseline sizes for NMOS and PMOS transistors

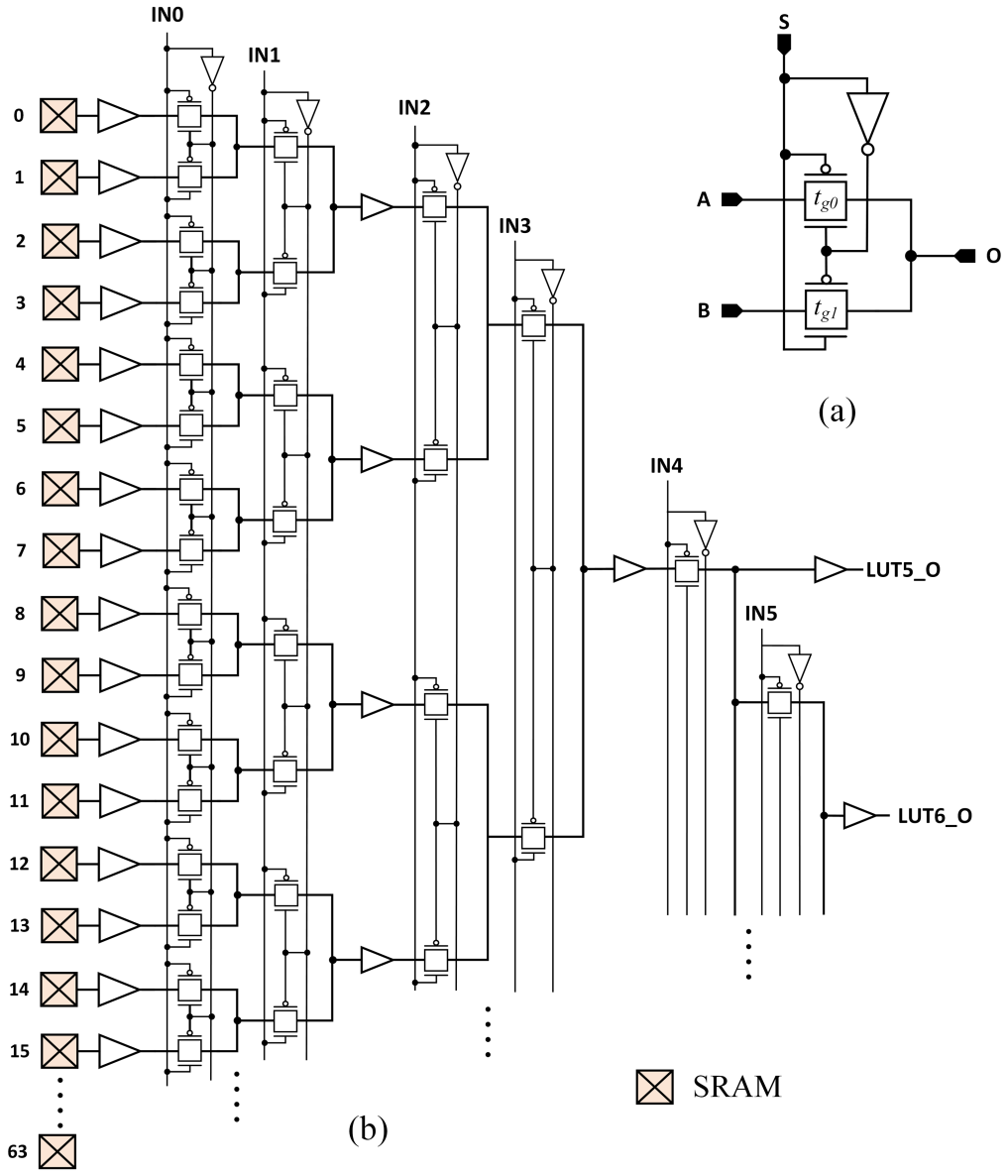


Figure 6.2: Design diagram for (a) 2:1 TG-based MUX. (b) TG-based 6-input LUT (partial view).

are determined based on the optimal DC Voltage Transfer Characteristic (VTC) curve.

Initially, Cadence Virtuoso platform with Spectre simulator was used to determine optimal transistor sizes and extract corresponding threshold voltages for each sized device. The Gaus-

sian random variable [79] was used to study the effect of WID variation on delay, efficiency, and output correctness. For each LUT implementation, 1,000 Monte Carlo samples are generated by assigning a random deviation in threshold voltage using a Gaussian distribution. Monte Carlo simulations are carried out using Synopsys HSPICE. To check transition faults at all MUX nodes, the commercial Synopsys TetraMAX APTG tool was used to generate the minimum number of test patterns that can be applied as LUT inputs and configuration values to check all possible transition at MUXes ports. The generated input sequence was used to test each Monte Carlo sample during SPICE simulation. Delays and power consumption data are collected for each sample. Due to the limited drive of pass transistors, different PT-based designs with increasing transistor sizes were included. The unit size parameter w signifies multiplication factor for transistor size whereby $w = 2$ indicates twice its original w/l ratio.

6.4 Results and Conclusions

Figure 6.3 (a) and (b) display the obtained frequency distribution of delay values for the 1,000 Monte Carlo sample designs of PT-based and TG-based LUTs, respectively, at $\sigma_{V_{th}} = 10\%$. The red vertical line in each figure indicates delay value for baseline design without variation $d_{baseline}$. It is evident that the effect of PV on performance follows a Gaussian distribution. It is also observed that the mean of delay distribution $\mu_{V_{th}}$ is higher than delay of baseline design.

The test patterns used during simulation provide high coverage to check against transition failures caused by any failed multiplexer. The defect rate defined by the proportion of 1,000 LUT designs that fail at least one test pattern for different design implementations is given in Figure 6.4. These simulation results reveal interesting observations. As expected,

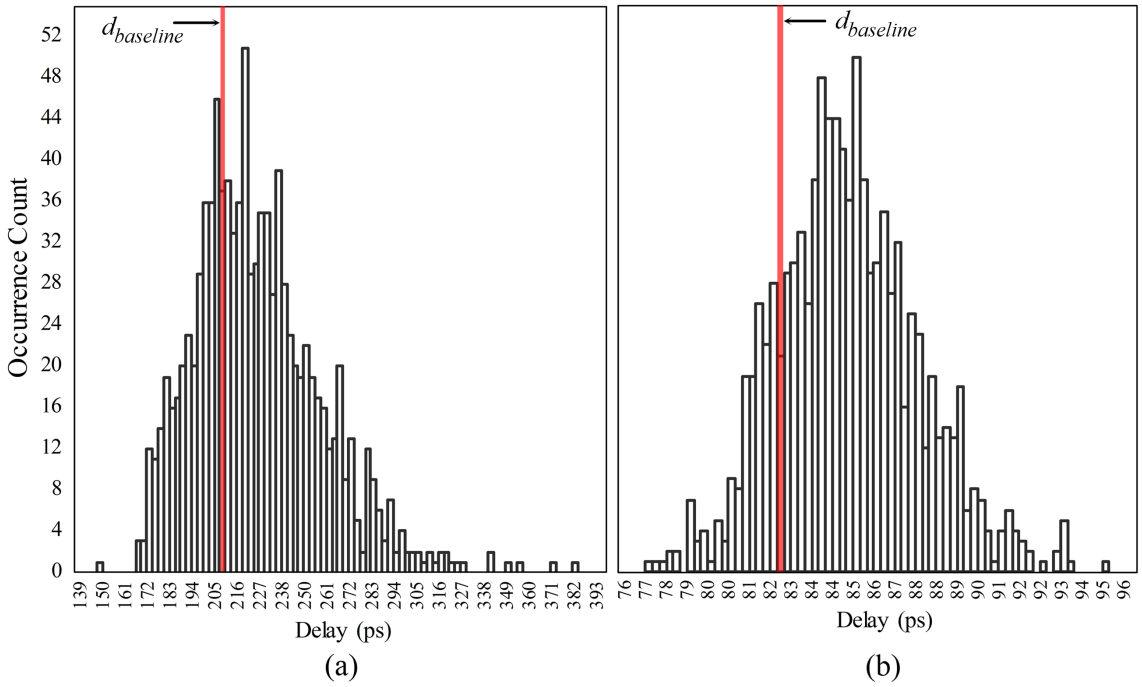


Figure 6.3: Delay distribution for (a) PT-based 6-input LUT ($w = 2$). (b) TG-based 6-input LUT ($w = 1$) at $\sigma_{V_{th}} = 10\%$.

designing a structure using PT-based multiplexers without proper transistor sizing results in a substantially high failure rate that exponentially increases as variation increases beyond 6%. Results also show that sizing pass transistors has considerably decreased defect rate as seen in the design cases where $w = 2, 3$, and 4. A diminishing improvement in variation tolerance is also observed as w increases. On the contrary, TG-based structure offers much less sensitivity to variation than any PT-based design in this evaluation. This is achieved while using the minimum optimal w/l ratio. The results here are restricted to a maximum variation of 30% which covers the ITRS expected variation range for current and future technology.

Figure 6.5 shows variation impact on mean delay for Monte Carlo simulations across variation range where functional yield $\geqslant 95\%$, i.e. defect rate is less than 5%. The TG-based

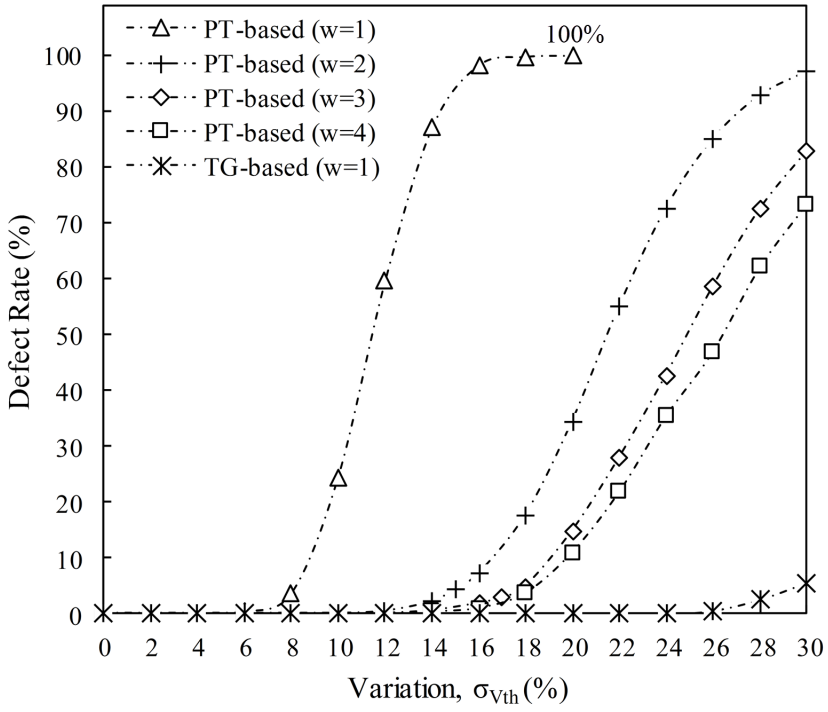


Figure 6.4: LUT defect rate vs. variation $\sigma_{V_{th}}$ using 16nm PTM model $1 \leq w \leq 4$.

implementation maintains another significant advantage in terms of latency. For instance, at $\sigma_{V_{th}} = 14\%$, TG-based LUT enables 64.4% (57.9%) reduction in latency compared to PT-based alternatives for $w = 2$ ($w = 3$). Transistor sizing for PT-based designs allows expected reduction in delay at a diminishing rate as w increases.

The effect of threshold voltage variation $\sigma_{V_{th}}$ on delay variation σ_{delay} is shown in Figure 6.6. The results reveal a pseudo-linear relation with exponential amplification under high variation. PT-based designs are indeed much more susceptible to variation than TG-based structures. Results also show that transistor sizing for pass transistors has a minor effect on mitigating design delay variation. At $\sigma_{V_{th}} = 14\%$, PT-based designs that meet the 95% yield objective can exhibit as high delay variation as 23.3%. This variation impact can be reduced to 4.9% if TG-based LUT is considered. Variation impact on design efficiency given

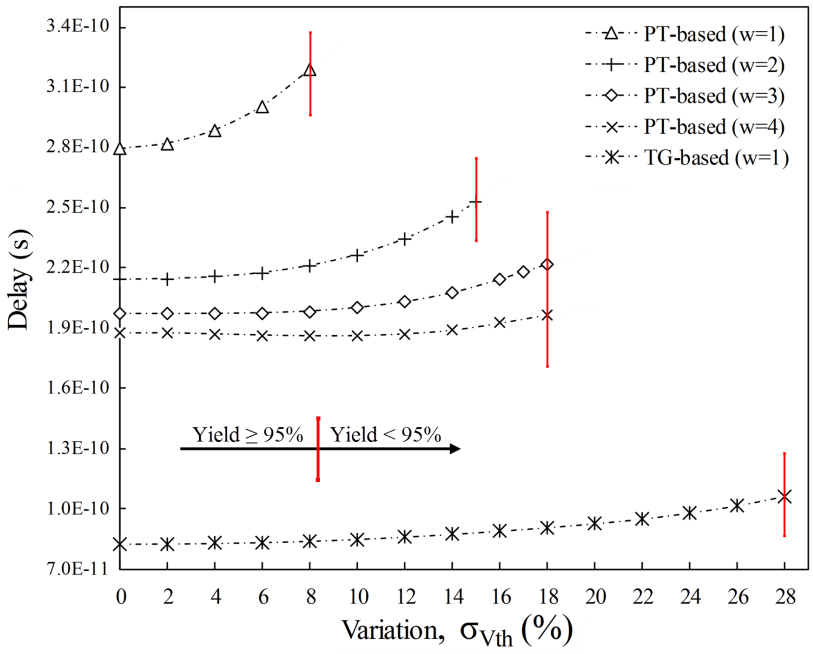


Figure 6.5: Effect of variation on mean delay using PT and TG MUXes.

by EDP is shown in Figure 6.7. We observe that the EDP increases with variation while TG-based design provides substantially lower EDP than any PT-based design. Results show that TG-based designs offer a substantially superior resilience to WID variation compared to other PT-based alternatives. This is achieved while using the optimal minimal transistor sizing.

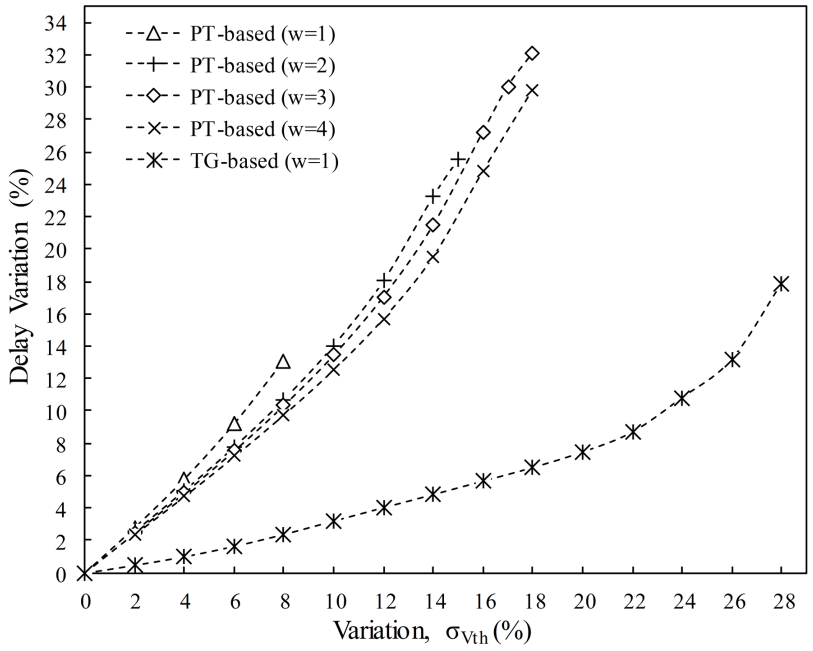


Figure 6.6: Design delay variation vs. $\sigma_{V_{th}}$ of PT and TG MUXes.

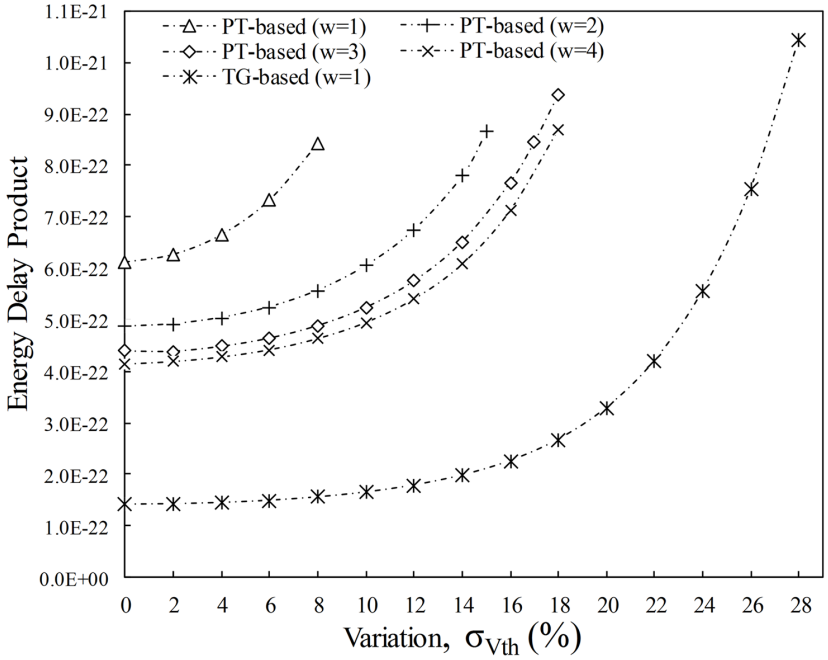


Figure 6.7: Energy-delay product vs. $\sigma_{V_{th}}$ for PT and TG MUXes.

CHAPTER 7: MITIGATING THE IMPACT OF PROCESS VARIATIONS VIA DISJUNCT RESOURCE UTILIZATION

This chapter investigates the applicability of using design disjunction as a post-silicon technique to increase the timing yield induced by process variation for reconfigurable hardware. The conventional method to reduce timing yield loss has replied on speed binning techniques to increase profit, which works effectively against time variation between dies. However, with the increasing within-die variability in current and future technology processes, the effectiveness of speed binning methods is expected to diminish in the nearest future; thus elevating the need for new alternative methods to address this problem. In this work, the performance of diverse resource utilization defined by design disjunction is assessed under varying levels of within-die process variation. Since design disjunction is demonstrated to provide an effective recoverability against hard failures, this advantage is extended here to circumvent resources affected by parametric variation. Experiments conducted on multiple designs have demonstrated up to 9.96% increase in performance and up to 57.45% gain in timing yield.

7.1 Introduction

Process variation presents a challenging problem for the design of digital circuit on scaled technologies. The impact of process variation can be observed as a fluctuation in final design performance and power efficiency. The impact of process variation on design performance can reach up to 30% [80]. Parametric variation can also lead to a reliability variation in which the lifespan of a design can significantly vary from one chip to another [81]. Thus analytical approaches to predict how final designs could behave should incorporate statistical methods to enhance their accuracy. The advancement in FPGAs architectures and their

capabilities have led to their widespread use in high performance applications. This entails a due consideration to mitigate the effect of process variation on reducing timing yield.

Process variation can be classified into inter-die and intra-die variation [82]. Inter-die variation refers to variation in parameters across identical dies. Intra-die variation is deviation of a design parameter within any one die. Inter-die variation can exhibit spatial correlation such that devices with a close proximity have a higher likelihood of having identical parameters than those that are far from each other. Independent variation can also occur within a die on each device [83]. For advanced technologies, inter-die, or within-die, variation has become a major source of variation [74]. Inevitable impact of inter-die variation can be masked by techniques such as speed pinning [84]. The objective of speed binning is to isolate high performing chips from chips that are affected by variation and sell each to the appropriate target market. This works effectively when different identical ASICs chips with known target applications in which performance of a design can be tested to classify each chip. However, considering the increasing impact of within-die variation, a design implementation on an FPGA may perform differently depending on where design blocks are placed and routed in the chip. In addition, the increasing variability on interconnect parasitics in advanced technology [85] [86] pushes for more consideration to mitigate process variation for FPGAs whose performance is largely determined by interconnects. It is also relevant to note that due to FPGA reconfigurability features, target applications are likely unknown after chips are manufactured, thus effective architectural optimization can be hard to realize.

To cope with the increasing sensitivity to within-die variation for FPGAs, researchers have proposed variation-aware placement and routing techniques [87] [88] [89]. These techniques rely on the delay and leakage data for each block in the chip before the optimal placement and routing can be determined. Extracting delay and leakage data for each chip can be computationally expensive given the increasing density of contemporary devices. Design

techniques, such as body-bias controlling and architecture enhancements methods [90] [74] are also proposed as timing optimization approaches. Due to the increasing trend in using COTS-based reconfigurable devices, new methods to address this problem should favorably be readily integrated with existing commercial devices and design tools. Thus, design solutions that dictate architectural changes in the FPGA design or are not possible to apply using closed-source vendors' tools are not considered in this work.

Another strategy to mitigate within-die variation is to use a set of distinct configurations [91] prepared before chips are arrived and the best performing configuration which meets design constraints can be used. The idea is to generate mutually exclusive critical path configurations (MECPCs) by identifying critical and near critical paths and reroute them to generate multiple independent configurations while placement is fixed. Results obtained using the academic tool VPR show a reduction of up to 49% for 30% variation in V_{th}. Although the method is easy, it requires a modification to the existing commercial place-and-route tools which is not possible; thus, it may not be utilized. The described work in this chapter provides an equivalent approach using design disjunction which can be integrated with the latest Xilinx toolchain. The proposed work provides the first study and assessment of Xilinx tools capability to produce highly disjunct designs for the purpose of increasing timing yield. This study also considers systematic and random within-die variations of both logic and routing resources.

The remainder of the paper is organized as follows. Section 2 describes the statistical design modeling under process variation used for the evaluation of this work. Section 3 discuses evaluation framework. Section 4 presents the experimental results. Finally, conclusions are given in Section 5.

7.2 Delay Modeling under Process Variation

Accurate modeling of process variation can be challenging due to the spatial correlation among all transistors in a chip. The two conventional methods for modeling process variation considering correlation are the principal component based model [92] and the quad-tree model [93] [94]. For the principal component based model, the high computational complexity required to analyze correlated parameters is simplified by transforming the set of correlated parameters to their uncorrelated principle components. This transformation depends on a covariance matrix which describes the level of spatial correlation among parameters. Spatial correlation using the principal component based model is normally based on an exponentially decaying function.

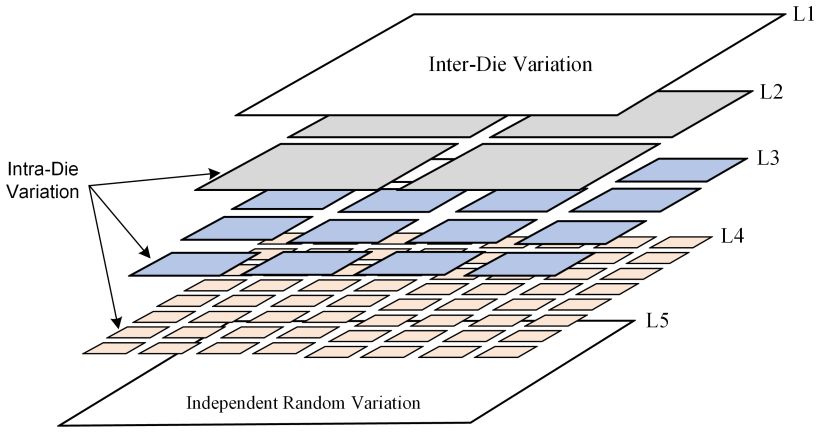


Figure 7.1: Quad-tree model using five layers.

The quad-tree method models process variation using a hierarchical quad tree structure. Figure 7.1 shows an example of a quad-tree model structure with five hierarchical layers. Each layer defines how design chip is partitioned into equal grids. Each grid gets a random variable with a variance identical to other grids in the same layer; and each layer has its own independent variation. The first top layer has a one gird of a size equivalent to the die area,

this layer is dedicated to represent inter-die variation. The rest layers, with the exception of the last bottom layer, are used to represent the effect of spatial within-die variation. The last bottom layer is used for independent random variation at the individual device level. To obtain a variation map for a design parameter p in a chip, the grid location of each layer k that covers the device is found and based on the variation value assigned to each grid, the effective parameter value is obtained by simply adding deviation values across all layers to the nominal value of the parameter as follows:

$$p_{total} = p_{nominal} + \sum_{k=1}^{n} \Delta p_k \tag{7.1}$$

where n is the number of layers in the model.

Spatial correlation between devices in this model is captured by the cells they share across these layers. The more layers added to this model the more accurate the results; however, the computational cost will increase. The total variation given by this model can be computed from the total variance as provided below:

$$\sigma^2_{total} = \sum_{k=1}^{n} \sigma^2_k \tag{7.2}$$

In this work, a quad-tree model with six layers for high accuracy is adopted. To reflect the effect of worsening within-die variation in our evaluation, the variations across the layers are made progressively increasing from top to bottom layers. The increased variation is governed

by the following form:

$$\sigma^2_{total} = \sum_{k=1}^{n} (\omega_k \sigma_k)^2 \tag{7.3}$$

where $\omega_l = \omega_{l-1} + 1$ for all $l \in [1, n]$, and $\omega_1 = 1$

Normally, variations across all layers are made equal as considered in [74]. However, for advanced technology, within-die variation is expected to become a major component of the total variability; thus, justifying the above consideration for allowing higher variability weights for lower layers. The quad-tree model is also used to model interconnect variability. To consider within variation of interconnects, the corresponding total variation values for an interconnect segment will depend on the location of its driving device within each model layer.

7.3 Evaluation Framework

In this work, the Xilinx 7-series was considered as a baseline architecture for the evaluation of design disjunction as a process variation mitigation technique. As demonstrated in chapters 4 and 5, Xilinx toolchain provide high flexibility to extract detailed post-place-and-route delay data for any given design. For the purpose of this work, delay data for each LUT and net used by critical and near critical paths are obtained using Xilinx's static timing analysis tool. Critical paths are the longest combinational paths between any two flipflops triggered by the same clock signal. Since the number of critical and near critical paths can be high which can later be computationally expensive to analyze, a cutoff of 80% from the longest critical path was used to identify the set of paths to be included in the analysis. In other words, only paths whose delays are greater than 80% of the longest critical path

are considered in this evaluation. The reason these near critical paths are considered is to account for the likelihood that process variation may cause a near critical path to emerge as a new top critical path for the design.

For each considered path, the utilized logic and routing resources from source node to destination node(s) are identified through the resource utilization description given by the XDL design file for post-place-and-route netlist. The Xilinx static timing analysis tool reports only the total delay for each defined net in a path. A net can be realized using several PIPs and wire segments located at distant sites on a chip. Thus, to model the effect of process variation using a quad-tree model for a net, the total delay variation for the net is computed by obtaining the weighted average variation for all PIPs used by that net. The weight for each PIP depends on the ratio of the length of the wire segment it drives to the total length of the net. The length of a wire segment is given by the number of tiles it spans.

To illustrate this delay model, consider net n1 used by design DC_x as depicted in Figure 7.2 which comprises six segments ($s_{11}, s_{12}, s_{13}, s_{14}, s_{15}$, and, s_{16}). The total delay of n1 equals the sum of arrival time at each PIP along the net which, for sake of simplicity, is equivalent to the sum of wire segments' delays. Since only the total delay is provided by the tool, the net variation is computed as follows:

$$\Delta_{n1} = \frac{\sum_{k=1}^{6} len(s_{1k}) \times \Delta_{s_{1k}}}{len(n1)} \tag{7.4}$$

where $len(x)$ denotes the length of interconnect element x

Decomposition of net delay in terms of its segments' latency is crucial to capture how distinct design implementations perform under identical variation map. Figure 7.2 also depicts net n2 used by design DC_y. Both nets have a mutual resource utilization of segments s_{14}, or s_{23}.

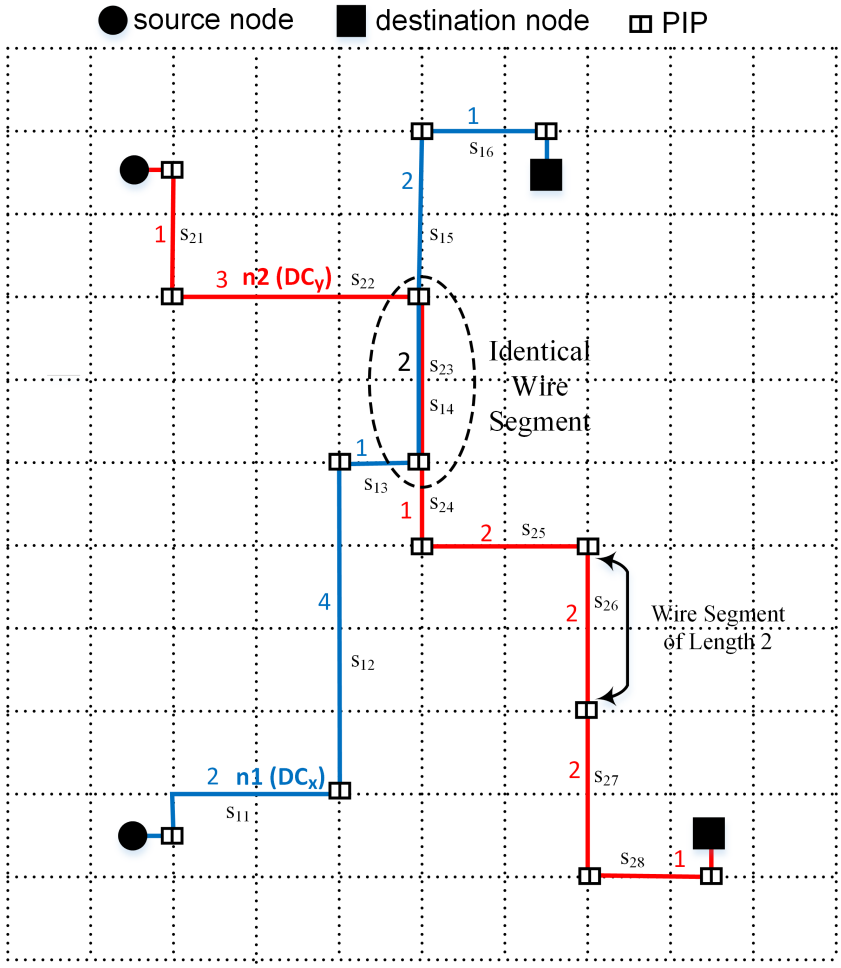


Figure 7.2: Example of routing resource utilization for two nets.

Given a variation map obtained by the quad-tree model, delays of nets n1 and n2 should be affected by the same variation observed at segment s_{14}.

After delay and resource utilization information of a design are extracted by parsing post-place-and-route design files, a variation map generated using the described quad-tree model, is applied to the utilized resources to find the amount of delay variation for each resource based on its coordinate location in the chip. The considered critical and near-critical paths are then updated according to the delay variation on all logics and nets on each path. The

largest path delay under variation is identified for each DC and used for comparison with other DCs. Top performing DC is the DC that reports the fastest critical path which is evaluated against the critical path of the baseline design under the same variation map. This latter step is repeated for 1000 randomly generated variation maps to get a statistically accurate assessment. Three different benchmarks: AES, s38417, and adpcm are considered for the evaluation in this work and results are reported in the following section. Design disjunction is evaluated for $\delta = 1$ and $\delta = 3$ to show the effect of increasing the number of disjunct configurations g on reducing variation effects. Three levels of total variations: 25%, 15%, and 5% are evaluated to assess the performance of the proposed design method as variation increases.

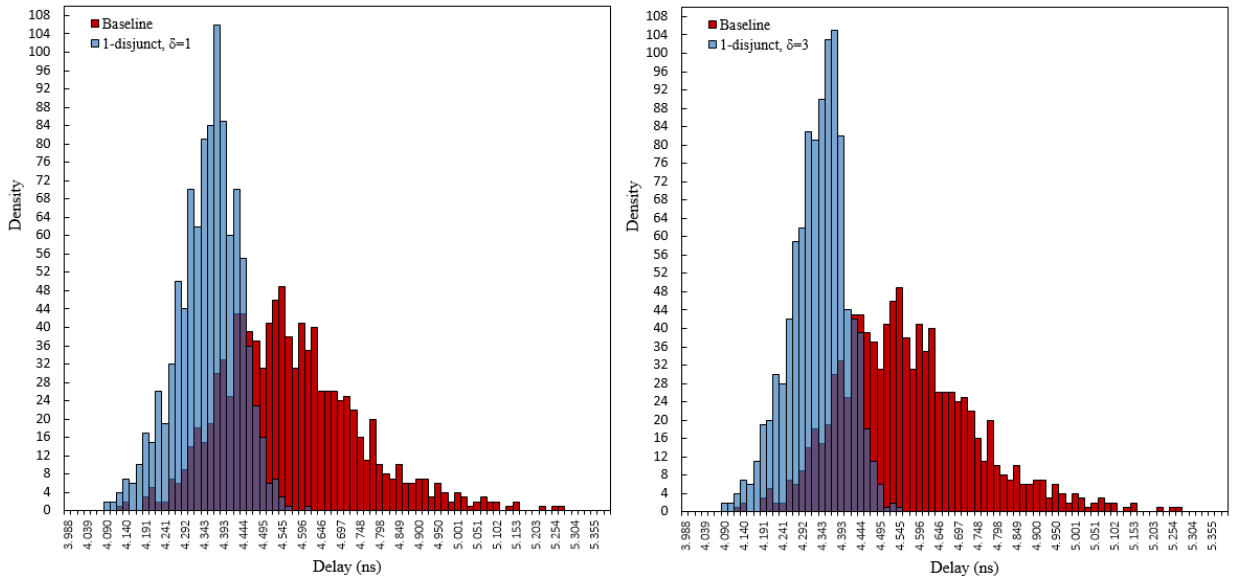


Figure 7.3: Density of delay of critical paths for AES benchmark.

7.4 Results and Anylsis

Figures 7.3, 7.4, and 7.5 show delay density of critical path for the three considered benchmarks after applying 1000 randomly generated variation maps for both baseline design and 1-disjunct set of DCs. It is evident that delay density of DCs exhibits a lower variation than that of the baseline design and also a lower mean for each benchmark. Reduction in delay mean μ and variance σ can be captured by the $\mu + 3\sigma$ value which reflects the upper 99.7% confidence limit for the probability density function. Table 7.1 shows the obtained delay data for critical paths of the three benchmarks considered in this work. Results for the upper 99.7% confidence limit show average reductions in critical path of 8.88%, 5.95%, and 2.78% under total variations of 25%, 15%, and 5%, respectively. These results indicate that design disjunction can provide a higher delay reduction as total variation increases. The reduction can be slightly increased to reach 9.96%, 6.85%, and 3.58% if disjunction ratio is increased from $\delta = 1$ to $\delta = 3$. This moderate increase in reduction demonstrates that increasing the number of alternative designs to get a lower critical path is not effective; thus, disjunct design using low δ values, offers the largest impact on reducing the effect of variability.

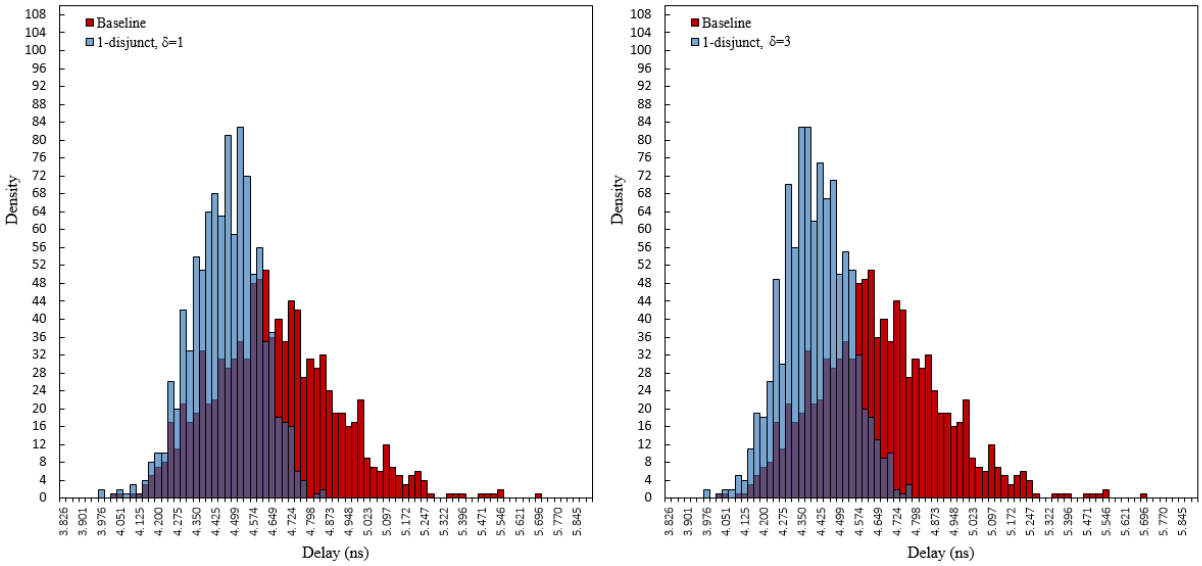


Figure 7.4: Density of delay of critical paths for adpcm benchmark.

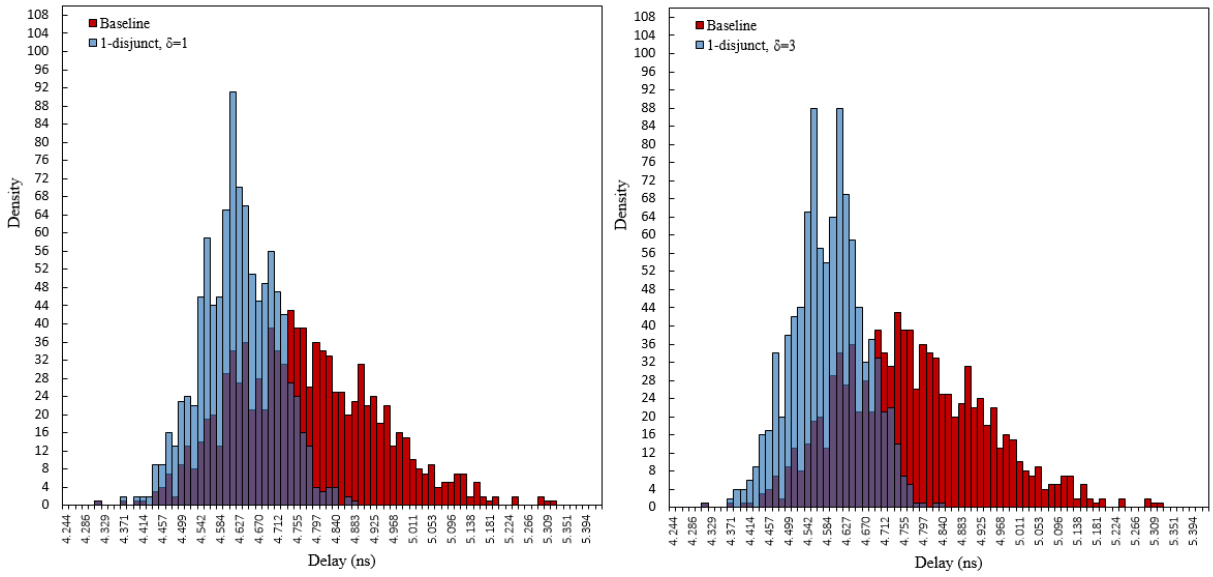


Figure 7.5: Density of delay of critical paths for s38417 benchmark.

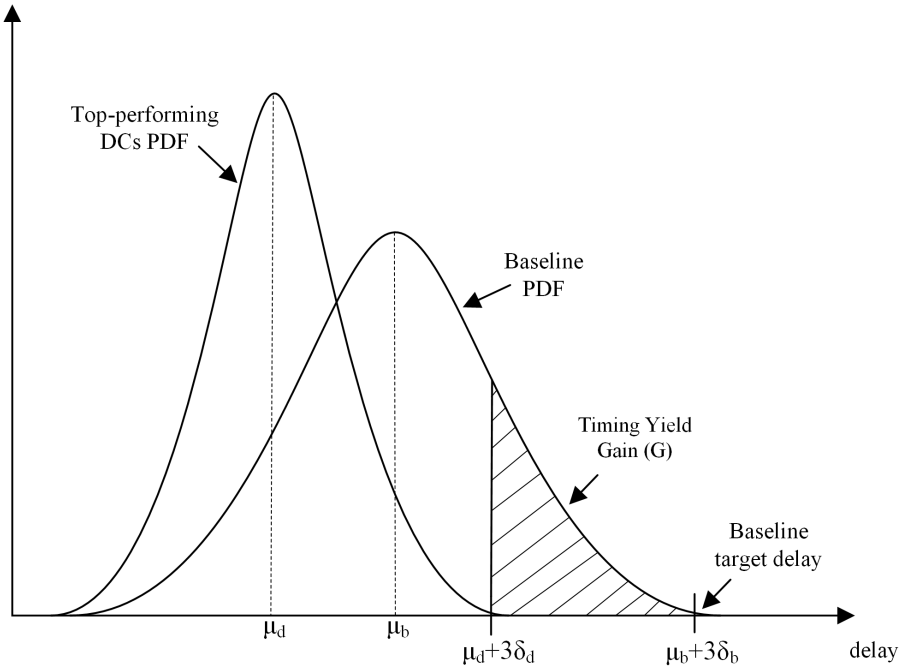


Figure 7.6: Probability density of delay of critical paths for baseline and design disjunction

Table 7.1: Delay of Critical Path for Baseline and Disjunct Designs

Total Variation	Baseline			1-disjunct, δ=1						1-disjunct, δ=3					
	μ	σ	μ+3*σ	g	μ	σ	μ+3*σ	G (%)	CP Reduction (%)	g	μ	σ	μ+3*σ	G (%)	CP Reduction (%)
AES															
25	4.557	0.176	5.085	24	4.345	0.081	4.589	42.703	9.749	40	4.326	0.074	4.549	51.777	10.538
15	4.384	0.106	4.700		4.246	0.062	4.431	32.641	5.736		4.239	0.057	4.409	40.268	6.193
5	4.235	0.041	4.357		4.153	0.035	4.259	27.460	2.239		4.153	0.035	4.258	28.540	2.269
s38417															
25	4.767	0.160	5.246	20	4.620	0.083	4.870	25.790	7.166	35	4.582	0.079	4.819	37.300	8.156
15	4.571	0.117	4.920		4.470	0.061	4.652	24.110	5.450		4.434	0.057	4.606	37.990	6.388
5	4.473	0.052	4.630		4.385	0.027	4.466	55.670	3.559		4.351	0.027	4.431	78.850	4.306
adpcm															
25	4.645	0.245	5.381	16	4.455	0.134	4.857	19.300	9.737	27	4.393	0.129	4.779	29.170	11.183
15	4.501	0.155	4.966		4.366	0.090	4.635	19.170	6.662		4.300	0.090	4.571	32.460	7.961
5	4.399	0.056	4.567		4.328	0.041	4.451	17.520	2.545		4.259	0.039	4.377	64.950	4.162

The gain in timing yield G can be quantified by computing the area under the PDF of delay distribution of baseline over the interval $[\mu_d + 3\sigma_d, \mu_b + 3\sigma_b]$ as illustrated in Figure 7.6. The gain G can be expressed as follows:

$$G = \int_{\mu_d+3\sigma_d}^{\mu_b+3\sigma_b} PDF_b \, dx \tag{7.5}$$

This quantity can also be defined in terms of the cumulative distribution function (CDF) of the baseline delay distribution as:

$$G = \int_{0}^{\mu_b+3\sigma_b} CDF_b \, dx - \int_{0}^{\mu_d+3\sigma_d} CDF_b \, dx \tag{7.6}$$

Table 7.1 reports G for each benchmark. The average gain in timing yield are 29.26%, 25.31%, and 33.55% for total variations of 25%, 15%, and 5%, respectively. The substantial gain in timing yield proves the effectiveness of the proposed design method to mitigate the impact of within-die variability. This gain in timing yield can also be enhanced by designing for $\delta > 1$. For $\delta = 3$, the gains are increased to 39.42%, 36.91%, and 57.45%, respectively.

7.5 Summary

In this work, we show that within-die variation on logic resources and interconnects can be mitigated through a set of disjunct configurations. These configurations are prepared at design-time using current commercial toolchain without the need for hardware and software modifications. Results for a set of benchmarks show average gains in timing yield of up to 39.42%, 36.91%, and 57.45% for total variations of 25%, 15%, and 5%, respectively. The enhanced timing yield is attained while achieving reductions in mean delay of 9.96% 6.85%,

and 3.58% for the same variability levels.

CHAPTER 8: CONCLUSION AND FUTURE WORK

8.1 Fast Online Diagnosis and Recovery using Design Disjunction

Current scaling trend of reconfigurable hardware and their improving design flexibility have fueled the continuous increase of their adoption in various applications. The reconfigurability feature can provide immense opportunities for designing effective FT platforms. Unfortunately, the design flexibility of reconfigurable hardware does not come at no cost. With current programmable logic and interconnect density, implementation time including place and route phases can take an order of minutes to hours using a state-of-the-art multi-processing machine [9]. Although, the complexity of execution time can be substantially decreased for incremental re-place and re-reroute tasks, it is still a difficult computational workload for embedded processing cores. Therefore, design-time FT approaches that minimize dependency on run-time invocation of design flow are more favorable.

In this research, a novel design-time technique for providing multi-fault isolation and recovery for reconfigurable hardware based on design disjunction and non-adaptive group testing has been presented. The research has featured three primary tasks:

The first task was to develop a parameterized construction method that defines how resources are distributed to achieve the described disjunctive property on implemented design configurations. The developed construction method based on the mosaic convergence algorithm can scale to thousands of resources which allows applicability to large designs and to allow a fine selection of any resource count and target design size. The defined resource distribution can be directly translated into a placement constraint file format supported by existing design tools.

The second task was to realize and validate the proposed FT approach. This includes defining and implementing the design steps that can be integrated into existing vendor CAD tools with minimal effort required to alter the conventional design flow. The proposed approach is validated and implemented on the latest Xilinx 7-sieres FPGA family. Experimental results on a set of diverse benchmarks have demonstrated f-diagnosability at the individual slice level with a minimum average isolation resolution of 96.4% (94.4%) for $f = 1$ ($f = 2$) without accounting for the impact of the low coverage of functional testing. An algebraic-based method was also introduced to further increase the fault isolation accuracy of proposed method to any level deemed adequate for successful recovery and for efficient and rapid repair. We also demonstrate the potential benefits of the proposed technique as a fault recovery. In particular, the proposed approach was theoretically and empirically demonstrated to provide multi-fault recoverability coverage at minimal time complexity.

The third task was to develop a framework for autonomous fault tolerance operations on an embedded reconfigurable hardware based on the proposed approach. A hardware implementation on a commercial Xilinx test and embedded board was considered to validate applicability of the proposed scheme to cover the second and third tasks.

The demonstrated tasks have shown that the proposed approach for fault tolerance can be integrated with other commonly used fault detection mechanisms such as TMR, DWC, parity-based, or any user-defined fault detection method at the system-level. The proposed FT scheme can also be combined with other fault tolerance approaches to ameliorate their fault recovery strategies.

8.2 Hypergraph-Cover Diversity for Maximally-Resilient Reconfigurable Systems

Although design disjunction can attain the reliability objectives through a combinatorial search for the optimal set of designs, the exponential time complexity of the search may hinder their applicability to very large systems. In this work, two novel deterministic algorithms based on graph and set theory are developed to address scalability concerns of the mosaic convergence algorithm. We show that set separability defined by hypergraph theory has great potential to create highly resilient designs at design-time for optimal low recovery overhead and energy saving. Results have demonstrated the potential of the proposed FT method to achieve 37.5% area saving and up to 66% reduction in power consumption compared to the frequently-used TMR scheme while providing a superior fault tolerance.

8.3 Mitigating the Impact of Process Variations via Disjunct Resource Utilization

Process variation has emerges as a major obstacle in the advance of manufacturing technology. The high impact of process variation can lead to a diminishing return from scaling devices. Traditionally, FPGAs have been at the frontier to adopt new scaling technology. This entails careful design considerations to avoid yield loss and maintain the expected advantages of smaller nodes. Although reconfigurability can be an obvious solution to mitigate the shortcoming related to both manufacturing and reliability, the complexity and cost of fine reconfiguration for a large density has been a challenge up until now. In this dissertation, the emphasis has been to address this problem by shifting the associated cost of design implementation to design-time. We extended our investigation of design disjunction to mitigating process variation through pre-defined set of diverse design which can be tested after target devices are manufactured to find the most variation tolerant. Results have shown

statistical significance to reduce mean delay and increase timing yield.

8.4 Future Work

Current implementation of design disjunction is limited to the structural grouping of hardware resources. Although this strategy simplifies how the method is realized using commercial toolchain, a better approach should include a functional grouping to increase the test coverage while providing equivalent isolation accuracy using a lower number of configurations. It is also remained to be investigated whether such an extension can be universally applicable to a large set of design topologies rather than being an application-dependent design-time solution.

The effect of leakage power in the final timing yield of the PV study in this work is not included. It is worthy to be investigated for an accurate prediction of how diversification in resource utilization can improve the overall timing yield.

LIST OF REFERENCES

[1] J. Williams, C. Massie, A. D. George, J. Richardson, K. Gosrani, and H. Lam, "Characterization of fixed and reconfigurable multi-core devices for application acceleration," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 3, no. 4, pp. 19:1–19:29, Nov. 2010.

[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 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'11)*, 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] 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.

[5] C. Carmichael and C. W. Tseng, "Correcting single-event upsets in Virtex-4 FPGA configuration memory," Xilinx, Application Note XAPP1088(v1.0), Oct. 2009.

[6] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb, "FPGA partial reconfiguration via configuration scrubbing," in *Proc. IEEE International Conference on Field Programmable Logic and Applications (FPL'09)*, Prague, Czech Republic, Aug./Sep. 2009, pp. 99–104.

[7] W. Zha, "Facilitating FPGA reconfiguration through low-level manipulation," Ph.D. dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA, Feb. 2014.

[8] 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.

[9] Xilinx, "Vivado design suite," White Paper WP416 (v1.1), Jun. 2012.

[10] M. Pignol, "COTS-based applications in space avionics," in *Proc. of Design, Automation and Test in Europe Conference (DATE'10)*, Dresden, Germany, Mar. 2010, pp. 1213–1219.

[11] R. Pellizzoni, P. Meredith, M. Caccamo, and G. Rosu, "Hardware runtime monitoring for dependable COTS-based real-time embedded systems," in *Proc. of IEEE 29th Real-Time Systems Symposium (RTSS'08)*, Barcelona, Spain, Nov./Dec. 2008, pp. 481–491.

[12] 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 International Test Conference (ITC'99)*, Atlantic City, NJ, Sep. 1999, pp. 973–982.

[13] 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.

[14] 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 International Test Conference (ITS'01)*, Baltimore, MD, Oct./Nov. 2001, pp. 924–931.

[15] K. Zhang, R. F. DeMara, and C. A. Sharma, "Consensus-based evaluation for fault isolation and on-line evolutionary regeneration," in *Evolvable Systems: From Biology to Hardware*. Berlin, Germany: Springer, 2005, pp. 12–24.

[16] S. Chakraverty, A. Agarwal, A. Agarwal, A. Kumar, and A. Sikri, "Design space exploration for high availability drFPGA based embedded systems," in *Proc. 1st International Conference on Advanced Machine Learning Technologies and Applications (AMLTA'12)*, Cairo, Egypt, Dec. 2012, pp. 234–243.

[17] B. Bollobás, *Modern graph theory*. Springer Science & Business Media, 1998, vol. 184.

[18] P. Frankl and Z. Föredi, "Union-free hypergraphs and probability theory," *European Journal of Combinatorics*, vol. 5, no. 2, pp. 127–131, 1984.

[19] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, Sep. 2003.

[20] T. Karnik and P. Hazucha, "Characterization of soft errors caused by single event upsets in CMOS processes," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 128–143, April 2004.

[21] Xilinx, "Partial reconfiguration user guide," Application Note UG702 (v14.5), Apr. 2013.

[22] Altera, "Increasing design functionality with partial and dynamic reconfiguration in 28-nm FPGAs," White Paper WP-01137-1.0, Jul. 2010.

[23] M. Berg, "Fault tolerance implementation within SRAM based FPGA design based upon the increased level of single event upset susceptibility," in *Proc. IEEE 12th International Symposium on On-Line Testing (IOLTS'06)*, Lake of Como, Italy, Jul. 2006, pp. 89–91.

[24] M. G. Parris, C. A. Sharma, and R. F. Demara, "Progress in autonomous fault recovery of field programmable gate arrays," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, p. 31, Oct. 2011.

[25] C. Stroud, E. Lee, S. Konala, and M. Abramovici, "Using ILA testing for BIST in FPGAs," in *Proc. IEEE International Test Conference (ITC'96)*, Washington, DC, Oct. 1996, pp. 68–75.

[26] A. Doumar and H. Ito, "Detecting, diagnosing, and tolerating faults in SRAM-based field programmable gate arrays: a survey," *IEEE Trans. VLSI 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] J. Emmert, C. Stroud, and M. Abramovici, "Online fault tolerance for FPGA logic blocks," *IEEE Trans. VLSI Syst.*, vol. 15, no. 2, pp. 216–226, Feb. 2007.

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

[30] X. Iturbe, A. Ebrahim, K. Benkrid, C. Hong, T. Arslan, J. Perez, D. Keymeulen, and M. Santambrogio, "R3TOS-based autonomous fault-tolerant systems," *IEEE Micro*, vol. 99, preprint, Jul. 2014, http://doi.ieeecomputersociety.org/10.1109/MM.2014.58.

[31] M. B. Tahoori, "High resolution application specific fault diagnosis of FPGAs," *IEEE Trans. VLSI Syst.*, vol. 19, no. 10, pp. 1775–1786, Oct. 2011.

[32] S. Mitra and E. McCluskey, "Which concurrent error detection scheme to choose ?" in *Proc. of IEEE Int. Test Conf. (ITC'00)*, Atlantic City, NJ, Oct. 2000, pp. 985–994.

[33] C. Bolchini, A. Miele, and C. Sandionigi, "Autonomous fault-tolerant systems onto SRAM-based FPGA platforms," *Journal of Electronic Testing*, vol. 29, no. 6, pp. 779–793, Nov. 2013.

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

[35] V. Hahanov, S. Galagan, V. Olchovoy, and A. Priymak, "Algebra-logical repair method for FPGA logic blocks," in *Proc. IEEE East-West Design & Test Symposium (EWDTS'10)*, St. Petersburg, Russia, Sep. 2010, pp. 482–487.

[36] 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 International Test Conference (ITC'13)*, Anaheim, CA, Sep. 2013, pp. 1–10.

[37] E. A. Stott, N. P. Sedcole, and P. Y. K. Cheung, "Fault tolerance and reliability in field-programmable gatearrays," *IET Computers & Digital Techniques*, vol. 4, no. 3, pp. 196–210, May 2010.

[38] 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.

[39] R. Dorfman, "The detection of defective members of large populations," *The Annals of Mathematical Statistics*, vol. 14, no. 4, pp. 436–440, Dec. 1943.

[40] 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.

[41] A. B. Kahng and S. Reda, "New and improved BIST diagnosis methods from combinatorial group testing theory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 3, pp. 533–543, Mar. 2006.

[42] A. Alzahrani and R. F. DeMara, "Non-adaptive sparse recovery and fault evasion using disjunct design configurations," in *Proc. ACM/SIGDA International Symposium on Field-programmable Gate Arrays (FPGA'14)*, Monterey, California, Feb. 2014, pp. 251–251.

[43] C. Bolchini and A. Miele, "Design space exploration for the design of reliable SRAM-based FPGA systems," in *Proc. IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems (DFTVS'08)*, Boston, MA, Oct. 2008, pp. 332–340.

[44] M. Cheraghchi, "Coding-theoretic methods for sparse recovery," in *Proc. IEEE 49th Annual Allerton Conference on Communication, Control and Computing (Allerton'11)*, Monticello, IL, Sep. 2011, pp. 909–916.

[45] A. J. Macula, "A simple construction of d-disjunct matrices with certain constant weights," *Discrete Mathematics*, vol. 162, no. 1-3, pp. 311–312, Dec. 1996.

[46] C. L. Chan, S. Jaggi, V. Saligrama, and S. Agnihotri, "Non-adaptive group testing: Explicit bounds and novel algorithms," in *Proc. of IEEE International Symposium on Information Theory (ISIT'12)*, Jul. 2012, pp. 1837–1841.

[47] E. Knill, W. J. Bruno, and D. C. Torney, "Non-adaptive group testing in the presence of errors," *Discrete Applied Mathematics*, vol. 88, no. 1, pp. 261–290, Nov. 1998.

[48] Xilinx, "Kc705 evaluation board for the kintex-7 fpga," User Guide UG810 (v1.5), Jul. 2014.

[49] 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.

[50] G. Varatkar and N. Shanbhag, "Error-resilient motion estimation architecture," *IEEE Trans. VLSI Syst.*, vol. 16, no. 10, pp. 1399–1412, Oct 2008.

[51] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*, 2nd ed. Berlin, Germany: Springer, Apr. 2011.

[52] C. Kohn, "Partial reconfiguration of a hardware accelerator on Zynq-7000 all programmable SoC devices," Xilinx, Application Note XAPP1088(v1.0), Jan. 2013.

[53] A. Alzahrani and R. DeMara, "Hypergraph-cover diversity for maximally-resilient reconfigurable systems," in *Proc. of IEEE 12th International Conference on Embedded Software and Systems (ICESS'15)*, New York, USA, Aug 2015, pp. 1–7.

[54] P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu, "An overview of reconfigurable hardware in embedded systems," *EURASIP Journal on Embedded Systems*, vol. 2006, no. 1, pp. 13–13, Jan. 2006.

[55] E. Stott, P. Sedcole, and P. Cheung, "Fault tolerance and reliability in field-programmable gate arrays," *IET Computers & Digital Techniques*, vol. 4, no. 3, pp. 196–210, May 2010.

[56] M. G. Parris, C. A. Sharma, and R. F. Demara, "Progress in autonomous fault recovery of field programmable gate arrays," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, p. 31, Oct. 2011.

[57] R. Libeskind-Hadas, N. Hasan, J. Cong, P. K. McKinley, and C. L. Liu, *Fault Covering Problems in Reconfigurable VLSI Systems.* Norwell, MA: Kluwer Academic Publishers, 1992.

[58] M. Garca-Valls, P. Uriol-Resuela, F. Ibez-Vzquez, and P. Basanta-Val, "Low complexity reconfiguration for real-time data-intensive service-oriented applications," *Future Generation Computer Systems*, vol. 37, pp. 191–200, Jul. 2014.

[59] A. L. Rosenberg, "A hypergraph model for fault-tolerant VLSI processor arrays," *IEEE Trans. Comput.*, vol. C-34, no. 6, pp. 578–584, Jun. 1985.

[60] K. Sugihara and T. Kikuno, "On fault tolerance of reconfigurable arrays using spare processors," in *Proc. of IEEE Pacific Rim International Symposium on Fault Tolerant Systems (PRFTS'91)*, Kawasaki, Japan, Sep. 1991, pp. 10–15.

[61] P. Erdös and S. Shelah, "On problems of moser and hanson," in *Graph theory and applications.* Springer, 1972, vol. 303, pp. 75–79.

[62] "Process for limiting orbital debris," NASA, Technical Standard NASA-STD-8719.14A, Dec. 2011. [Online]. Available: http://http://www.hq.nasa.gov/office/codeq/doctree/871914.pdf

[63] "KC705 evaluation board for the Kintex-7 FPGA," Xilinx, User Guide UG810(v1.6.1), Apr. 2015.

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

[65] M. Violante, C. Meinhardt, R. Reis, and M. Reorda, "A low-cost solution for deploying processor cores in harsh environments," *IEEE Trans. Ind. Electron.*, vol. 58, no. 7, pp. 2617–2626, Jul. 2011.

[66] A. Alzahrani and R. DeMara, "Process variation immunity of alternative 16nm HK/MG-based FPGA logic blocks," in *Proc. of IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS'15)*, Fort Collins, CO, Aug 2015, pp. 1–4.

[67] K. Kuhn, M. Giles, D. Becher, P. Kolar, A. Kornfeld, R. Kotlyar, S. Ma, A. Maheshwari, and S. Mudanai, "Process technology variation," *IEEE Trans. Electron Devices*, vol. 58, no. 8, pp. 2197–2208, Aug 2011.

[68] K. Kuhn, "Reducing variation in advanced logic technologies: Approaches to process and design for manufacturability of nanoscale CMOS," in *IEEE International Electron Devices Mtg*, Wash., DC, Dec. 2007, pp. 471–474.

[69] "Design for manufacturability technology requirements table," International Technology Roadmap for Semiconductors ITRS, Tech. Rep., 2011. [Online]. Available: http://www.itrs.net/ITRS~1999-2014~Mtgs,~Presentations~&~Links/2011ITRS/2011Tables/Design_2011Tables.xlsx

[70] C. Chiasson and V. Betz, "Should FPGAs abandon the pass-gate?" in *Proc. Intl. Conf. on Field Programmable Logic and Appl.*, Porto, Portugal, Sep. 2013, pp. 1–8.

[71] C. Chen, R. Parsa, N. Patil, S. Chong, K. Akarvardar, J. Provine, D. Lewis, J. Watt, R. T. Howe, H.-S. P. Wong, and S. Mitra, "Efficient fpgas using nanoelectromechanical relays," in *Proc. 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'10)*, Monterey, CA, Feb. 2010, pp. 273–282.

[72] A. DeHon and N. Mehta, "Exploiting partially defective LUTs: Why you don't need perfect fabrication," in *Proc. International Conference on Field-Programmable Technology (FPT'13)*, Kyoto, Japan, Dec. 2013, pp. 12–19.

[73] Y. Taur, D. Buchanan, W. Chen, D. Frank, K. Ismail, S.-H. Lo, G. Sai-Halasz, R. Viswanathan, H.-J. Wann, S. Wind, and H.-S. Wong, "CMOS scaling into the nanometer regime," *Proceedings of the IEEE*, vol. 85, no. 4, pp. 486–504, Apr. 1997.

[74] A. Kumar and M. Anis, "FPGA design for timing yield under process variations," *IEEE Trans. VLSI Syst.*, vol. 18, no. 3, pp. 423–435, Mar. 2010.

[75] H.-Y. Wong, L. Cheng, Y. Lin, and L. He, "FPGA device and architecture evaluation considering process variations," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD'05)*, San Jose, CA, Nov. 2005, pp. 19–24.

[76] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 12, pp. 1859–1880, Dec. 2005.

[77] T. Pi and P. J. Crotty, "FPGA lookup table with transmission gate structure for reliable low-voltage operation," Dec. 23 2003, US Patent 6,667,635.

[78] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Trans. VLSI Syst.*, vol. 12, no. 3, pp. 288–298, Mar. 2004.

[79] S. Nassif, K. Bernstein, D. Frank, A. Gattiker, W. Haensch, B. Ji, E. Nowak, D. Pearson, and N. Rohrer, "High performance CMOS variability in the 65nm regime and beyond," in *IEEE International Electron Devices Meeting*, Wash., DC, Dec. 2007, pp. 569–571.

[80] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proc. 40th Annual Design Automation Conference (DAC'03)*, Anaheim, CA, Jun. 2003, pp. 338–342.

[81] C. Zhuo, D. Sylvester, and D. Blaauw, "A statistical framework for post-fabrication oxide breakdown reliability prediction and management," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 4, pp. 630–643, Apr. 2013.

[82] A. Agarwal, V. Zolotov, and D. Blaauw, "Statistical clock skew analysis considering intradie-process variations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 8, pp. 1231–1242, Aug. 2004.

[83] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "VARIUS: A model of process variation and resulting timing errors for microarchitects," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 3–13, Feb 2008.

[84] S. Bhunia, S. Mukhopadhyay, and K. Roy, "Process variations and process-tolerant design," in *Proc. 20th International Conference on VLSI Design*, Bangalore, India, Jan. 2007, pp. 699–704.

[85] E. Foreman, P. Habitz, M. Cheng, and C. Visweswariah, "A novel method for reducing metal variation with statistical static timing analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 8, pp. 1293–1297, Aug 2012.

[86] A. Ceyhan and A. Naeemi, "Cu/Low-k interconnect technology design and benchmarking for future technology nodes," *IEEE Trans. Electron Devices*, vol. 60, no. 12, pp. 4041–4047, Dec 2013.

[87] Z. Guan, J. Wong, S. Chaudhuri, G. Constantinides, and P. Cheung, "A two-stage variation-aware placement method for FPGAS exploiting variation maps classification,"

in *Proc. 22nd International Conference on Field Programmable Logic and Applications (FPL'12)*, Oslo, Norway, Aug 2012, pp. 519–522.

[88] L. Cheng, J. Xiong, L. He, and M. Hutton, "FPGA performance optimization via chipwise placement considering process variations," in *Proc. International Conference on Field Programmable Logic and Applications (FPL'06)*, Madrid, Spain, Aug. 2006, pp. 1–6.

[89] S. SRINIVASAN and V. Narayanan, "Variation aware placement for FPGAs," in *Proc. IEEE Annual Symposium on Emerging VLSI Technologies and Architectures*, Karlsruhe, Germany, March 2006.

[90] G. Nabaa, N. Azizi, and F. Najm, "An adaptive FPGA architecture with process variation compensation and reduced leakage," in *Proc. 43rd ACM/IEEEDesign Automation Conference(DAC'06)*, San Francisco, CA, Jul. 2006, pp. 624–629.

[91] Y. Matsumoto, M. Hioki, T. Kawanami, H. Koike, T. Tsutsumi, T. Nakagawa, and T. Sekigawa, "Suppression of intrinsic delay variation in FPGAs using multiple configurations," *ACM Trans. Reconfigurable Technol. Syst.*, pp. 1–31, Mar. 2008.

[92] H. Chang and S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 9, pp. 1467–1482, Sep. 2005.

[93] B. Cline, K. Chopra, D. Blaauw, and Y. Cao, "Analysis and modeling of CD variation for statistical static timing," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD'06)*, San Jose, CA, Nov. 2006, pp. 60–66.

[94] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD'03)*, San Jose, CA, Nov. 2003, pp. 900–907.