A COMPETITIVE RECONFIGURATION APPROACH TO AUTONOMOUS FAULT HANDLING USING GENETIC ALGORITHMS

by

KENING ZHANG
B.S. Xidian University, 1998
M.S. University of Central Florida, 2004

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

Summer Term
2008

Major Professor: Ronald F. DeMara

ABSTRACT

In this dissertation, a novel self-repair approach based on Consensus Based Evaluation (CBE) for autonomous repair of SRAM-based Field Programmable Gate Arrays (FPGAs) is developed, evaluated, and refined. An initial population of functionally identical (same input-output behavior), yet physically distinct (alternative design or place-and-route realization) FPGA configurations is produced at design time. During run-time, the CBE approach ranks these alternative configurations after evaluating their discrepancy relative to the consensus formed by the population. Through runtime competition, faults in the logical resources become occluded from the visibility of subsequent FPGA operations. Meanwhile, offspring formed through crossover and mutation of faulty and viable configurations are selected at a controlled re-introduction rate for evaluation and refurbishment. Refurbishments are evolved in-situ, with online real-time input-based performance evaluation, enhancing system availability and sustainability, creating an Organic Embedded System (OES).

A fault tolerance model called N Modular Redundancy with Standby (NMRSB) is developed which combines the two popular fault tolerance techniques of NMR and Standby fault tolerance in order to facilitate the CBE approach. This dissertation develops two of instances of the NMRSB system – Triple Modular Redundancy with Standby (TMRSB) and Duplex with Standby (DSB). A hypothetical Xilinx Virtex-II Pro FPGA model demonstrates their viability for various applications including a 3-bit x 3-bit multiplier, and the MCNC91 benchmark circuits. Experiments conducted on the model

evaluate the performance of three new genetic operators and demonstrate progress towards a completely self-contained single-chip implementation so that the FPGA can refurbish itself without requiring a PC host to execute the Genetic Algorithm.

This dissertation presents results from the simulations of multiple applications with a CBE model implemented in the C++ programming language. Starting with an initial population of 20 and 30 viable configurations for TMRSB and DSB respectively, a single stuck-at fault is introduced in the logic resources. Fault refurbishment experiments are conducted under supervision of CBE using a fitness state evaluation function based on competing outputs, fitness adjustment, and different level threshold. The device remains online throughout the process by which a complete repair is realized with Hamming Distance and Bitweight voting schemes. The results indicate a Hamming Distance TMRSB approach can prevent the most pervasive fault impacts and realize complete refurbishment. Experimental results also show that the Autonomic Layer demonstrates 100% faulty component isolation for both Functional Elements (FEs) and Autonomous Elements (AEs) with randomly injected single and multiple faults. Using logic circuits from the MCNC-91 benchmark set, availability during repair phases averaged 75.05%, 82.21%, and 65.21% for the z4ml, cm85a, and cm138a circuits respectively under stated conditions. In addition to simulation, the proposed OES architecture synthesized from HDL was prototyped on a Xilinx Virtex II Pro FPGA device supporting partial reconfiguration to demonstrate the feasibility for intrinsic regeneration of the selected circuit.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1. Introduction to Genetic Algorithms

In computer science, Evolutionary Computation (EC) [1] is a subfield of Artificial Intelligence (AI) [2] that involves combinatorial optimization problems which uses iterative progress, such as growth or development in a population using guided random search to achieve the desired end. Two developed techniques involved in EC which are based on identical principles, but different biology behaviors are Evolutionary Algorithms (EAs) [3] and Swarm Intelligence (SI) [4]. They have been heavily researched and implemented in different problem solutions which start from limited available information about uncertain environment and eventually develop an approximated informative solution based on interaction of the population solutions themselves. EA emphasizes population-based meta-heuristic optimization approach which is composed of Genetic Algorithms (GA) [5], Evolutionary Programming (EP) [6], Evolutionary Strategy (ES) [7], Genetic Programming (GP) [8-10] and Learning Classifier System (LCS) [11], while SI is more based around the study of collective behavior in decentralized system which composed of Ant Colony Optimization (ACO) [12] and Particle Swarm Optimization (PSO) [8]. This dissertation concentrates on developing EA-based approach for fault-handling methods.

Genetic Algorithms (GAs) [1] are the most popular EA technique inspired by biological mechanisms of evolution used in finding exact or approximated solutions to either search problems or optimization problems originated by John H. Holland and his colleagues at the University of Michigan in the 1970s. A computer simulation, a population of individuals,

each of which represents a potential solution to the problem, explores and exploits the search space in response to the environment of the individuals undergoing adaptation. An individual is encoded by various computer manipulatable structures, and the typical structure is a binary string although the best representations are determined by the problem being solved. Problem representation is one of the key decisions to be made when applying a GA because it may affect the adaptation process in terms of shape of the solution space that a GA searches through as well as solution complexity and precision. Furthermore, a measurement of the performance of the population named the *Fitness Function* is implemented to select the candidate for the next generation for further operation. Figure 1 below shows a conventional GA flow.

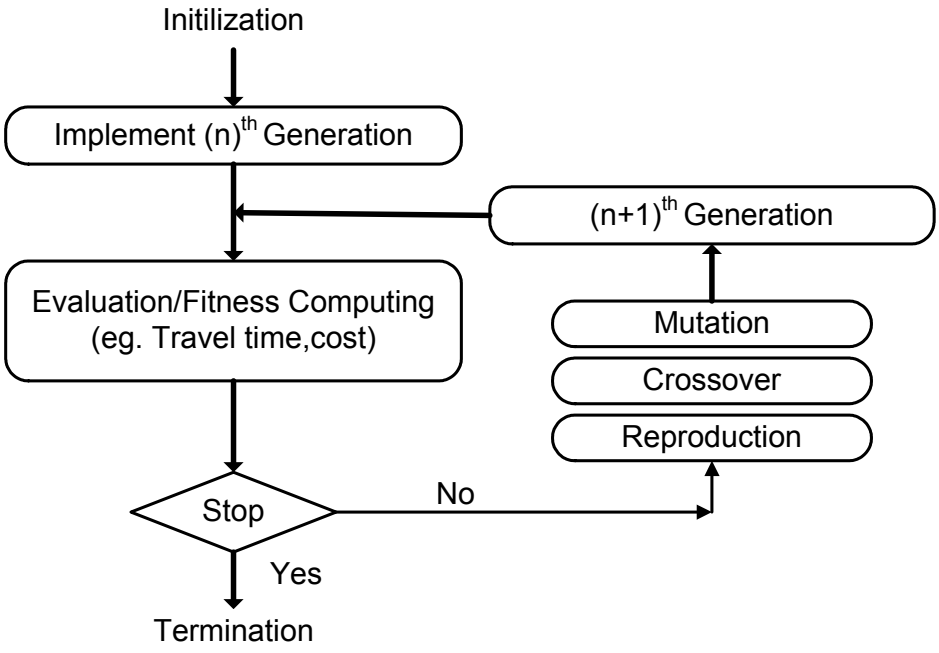

Figure 1: Genetic Algorithm Process

The GA repeats the above steps in Figure 1 iteratively in order to create better designs. The procedure begins with the initialization of the *individuals* in the population. An evaluation mechanism for the assessing the suitability of each individual design in the population is utilized called a *fitness function*. A fitness function computes how well a particular design performs in terms of some specific metrics. Different operators like *mutation* and *crossover* will be used for new offspring generations until the stop condition can be achieved for termination the process. The application field of GA is focused on the optimization and search problems which appear in biogenetics, computer science, engineering, economics, chemistry, manufacturing, mathematics, and physics [3, 8, 10, 13-17]. Evolvable Hardware is one of emerging application fields which emphasizes digital circuit design and fault tolerance based on reprogrammable devices.

1.2. Using Evolvable Hardware to Increase Reliability

Reliable embedded computing systems are vital to every sector of our economy and daily personal lives. Embedded systems using *Field Programmable Gate Arrays (*FPGAs*)* are frequently relied upon in mission-critical applications like deep space explore missions where the safety of human life and material assets are at risk. The recent availability of large multi-million gate-equivalent FPGAs provides the necessary resources facilitates the feasibility of using *Genetic Algorithms* (GAs) on these devices. GAs are used to evolve diverse and competitive solutions for a variety of problems, ranging from the general to the specific, by occluding the faults in the device at runtime. The reason GAs can be an appropriate adaptive mechanism for embedded systems are that they can adjust the solution quality without external control or

supervision. GAs can also adapt and respond to many unforeseen fluctuations in the operating environment.

*Evolvable Hardware (*EHW*)* [10, 14, 18] combines the benefits of reconfigurable hardware with GAs to offer efficient solutions to fault-related problems. Conventionally, EHW can be classified into two categories depending on the method of simulation. In the *Extrinsic Evolutionary* method, the physical condition of real circuits is simulated and a user defined genotype is used to evolve each individual outside of the real chip. Finally, the best-fit individual is selected and programmed into the real chip. On the other hand, in the *Intrinsic Evolutionary* method, the fitness is evaluated at run-time by using the phenotypes of the individuals directly in the real chip.

Depending on the application, EHW can be classified into two categories. One perspective is an alternative to traditional, specification-based manual circuit design techniques. In the other perspective, EHW is online device capability for autonomous reconfiguration. This dissertation will view EHW along the lines of the second approach. A fundamental difference of these two views is the former places the emphasis on the design phase and the latter emphasizes the execution, or run-time phase. The evolutionary design approach has several advantages as compared to the manual approach. For example, no a-priori knowledge is required on the specific domain, and the availability of a wider search space may help exploration of novel solutions.

With benefits of the EHW introduced in the previous section, still, there exist some substantial obstacles to overcome before there is wide utilization of this approach. First of all, the limited

number of optimal/suboptimal potential solutions within a large the gigantic search space always lead to excessive evolution time even under parallel search with multiple individuals in a population. This may not satisfy the cost-effective and efficiency of most problem solving criteria. For example, suppose there is a 2-bit adder composed of 10 gates and each gate can be implemented using 4 different functionalities (AND, OR, XOR, NOT). Without considering interconnection, there are 4^{10} possible ways and whenever one more gate is added to current design, that will increase 4-fold the possible designs over the previous designs. Instead of starting from scratch, some heuristic approaches have to be used as auxiliary tools to assistance exploring and exploiting the search space. To address this problem, a diverse population is used to supply candidate solutions initially as described below. Previous work did not investigate the benefit of diversity.

Secondly, each candidate problem is going to use specific application dependent fitness function to evaluate the new generated individuals for further evolution. Until now, no one has been proposed any universal fitness functions which can fit even similar classes of applications. However, without a versatile fitness function, it is difficult to assess how adaptive it will be for a GAs concept. Since a special fitness function must be dedicated for certain application in order to get accurate evaluation, knowing all of the circuit functionalities becomes a prerequisite system constraint which definitely decreases the feasibility of the GA utilization. To address this problem, this dissertation develops a standardized fitness assessment scheme based on discrepancy behavior suitable for any combinational logic circuit.

Lastly, most current EHW research is focused on digital circuit design which use randomly generated configurations as seeds which are evolved further with new offspring individuals in

subsequent generations. Starting from scratch is one possible way for small design, but not appropriate for design scalability since current FPGA device have multi-million gate capacities. Unlike conventional fault tolerance technology such as Triple Module Redundancy (TMR) [19] which uses majority information to maintain the current output for the system, there is no previously proposed idea to utilize the majority of the operational information contained in the population to maintain the system performance.

Consensus Based Evaluation (CBE) for autonomous repair of SRAM-based Field Programmable Gate Arrays (FPGAs) proposed in the dissertation is first implemented as a system using a general population consensus information to replace the specific fitness function based on a pre-designed population of functionally identical (same input-output behavior), yet physically distinct (alternative design or place-and-route realization) FPGA configurations. Instead of exploring the entire search space for the solution, any surviving individuals under a fault condition will be used as starting point for evolution. Furthermore, even if there are no individuals that survive unaffected, the population still can maintain acceptable system availability using partial working configuration which may not generate all possible correct outputs, yet maintain a useful amount of correct outputs.

For the CBE approach, the target applications are those mission-critical embedded systems which can utilize hundreds of field programmable devices with very limited capacity for spares. Another feature is that human intervention is infeasible for such applications in deep space or deep sea missions which required autonomy self-recovery as primary functionality. Also even they are mission-critical, using background evolution cycles, the system still provides attractive alternatives to device redundancy under graceful degradation except for catastrophic failures.

However, the system is not required to anticipate any specific environment it will experience and instead can dynamically adjust its configuration according to correspondent external conditions.

Two experimental applications are presented in this dissertation. One is using standard benchmark circuits implemented on an FPGA software model for system reliability analysis and the other on a prototype of an Organic Computing model. Both applications are using Xilinx Virtex-II Pro architecture model as hardware platform which introduced in next two sections and detailed in Chapter 4 and 5. The last section of this chapter presented the research objectives of this dissertation in more detail.

1.3. FPGA Architecture

1.3.1. Xilinx FPGA Architecture

The FPGA hypothetical model is inspired by Xilinx-style architecture [20-22].

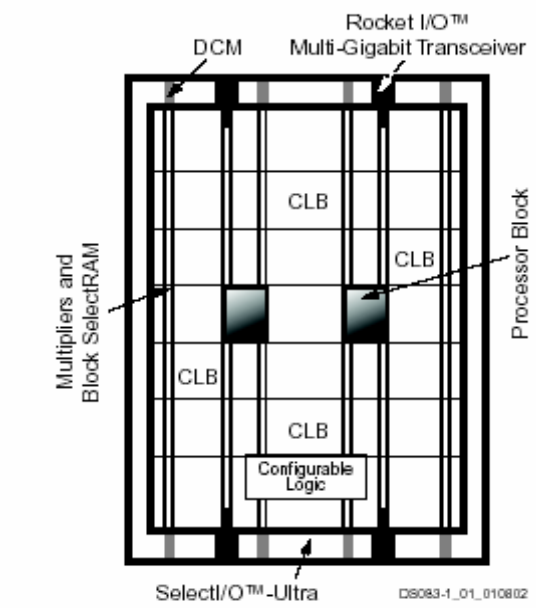

Figure 2: Xilinx Virtex-II Pro device Generic Architecture Overview

- Virtex-II Pro devices, as shown in Figure 2, are built on the Virtex-II FPGA architecture and are user-programmable gate arrays with various configurable elements and embedded cores optimized for high-density and high-performance system designs. The Virtex-II Pro family has the following features crucial to the design.

- Up to 22,592 Configurable Logic Blocks (CLBs) provide abundant reconfigurable recourses [21] with strong functional elements for combinatorial and synchronous logic, including basic storage elements (distributed RAM), MUX, fast carry chains, arithmetic logic, and BUFTs (3-state buffers).

- Up to four Incorporated embedded PPC405 cores in a single Virtex-II Pro device operate up to 400+ MHz with specially designed interface logic integrates the core with the surrounding CLBs, block RAMs, and general routing resources, which not only makes the implementation of autonomous system possible but also brings more flexibility and possibility to carry out complex reconfiguration application, such as GAs, in an even faster way by reducing off-chip I/O.

- A large amount of memory are available on-chip and on board, including the on-chip block RAM, on-board SDRAM SODIMM, Mobile SDRAM, Asynchronous SRAM and Flash, which provides huge extension for large calculation and reconfigurations.

The additional functionalities, such as Embedded 18-bit x 18-bit multipliers, Digital Clock Manager (DCM) blocks and multi-gigabit transceiver blocks, etc, [20], may greatly enhance programmable logic design and provide possible application extensions in the future as well.

1.3.2. Hypothetical FPGA Architecture

The hypothetical structure used in this dissertation is shown in Figure 3, which is similar to the architecture introduced in section 1.3.1. The feed-forward combinational logic digital circuit uses a rectangular array of nodes with two inputs and one output. Each node represents a Look-up Table (LUT) in the FGPA device, and a Configurable Logic Block (CLB) is composed of four LUTs. In the array, each CLB will be a row of the array and two LUTs are represented as four columns of the array. There are five dyadic functions -- OR, AND, XOR, NOR, NAND --

and one unary-function NOT, each of which can be assigned to an LUT. The LUTs in the CLB array will be indexed from 1 to *n*. This linear labeling enforces a feed-forward property in the combinational digital circuit for the array interconnection and routing.

Array routing is defined by the internal connectivity and the inputs/outputs of the array. Internal connectivity is specified by the connections between the array cells. The inputs of the cells can only be the outputs of cells with lower row numbers. Alternatively, the outputs of each cell are only allowed to be inputs of cells with higher row numbers.

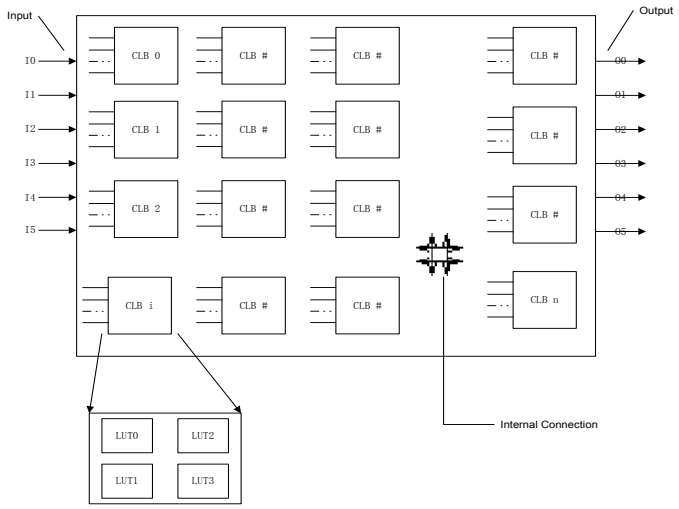

Figure 3: Genotype in a rectangular array cells

A phenotype is any observable characteristic of an organism, such as its morphology, development, biochemical or physiological properties, or behavior. They can also be represented as a linear string of integers as shown in Figure 4. This scheme is comprised of multiple CLB fields as well as array input and output fields. Array input-output fields are at the beginning and at the end of the entire configuration. Each CLB field is composed of a number of component

LUTs whose functionality and interconnection is specified. The first bit of the CLB field is the CLB number that indicates the relative order of the CLB in the entire configuration. Each LUT field within the CLBs is composed of a bit which reflects the functionality and bits which reflect the two inputs of the LUT. The array input and output sections both have six input bits and six output bits at the beginning and the end of the configuration.

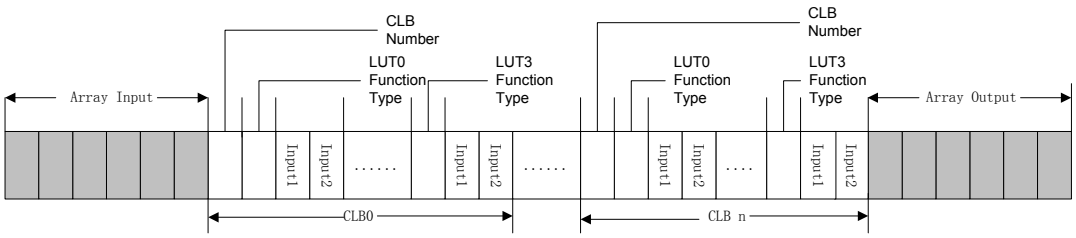

Figure 4: Genotype array representations

1.4. Organic Computing Concept

The realizations of systems that are capable of exhibiting such adaptive behaviors constitute the vision sought by *Organic Computing (OC)* [23]. OC *self-x* properties include self-configuration, self-reorganization, and self-healing which comprise the focus of this dissertation [23-26]. Ideally, these objectives are maintained in an autonomous fashion, yet sufficiently constrained to avoid undesirable emergent behaviors. In particular, OC systems rely on self-organization to respond to internal imbalances and changing environmental conditions using *an Observer/Controller* architecture [23, 27, 28].

To provide OC architectures with sufficient capability for exhibiting self-adaptive behavior, reconfigurable logic devices offer an attractive hardware platform. SRAM-based Field

Programmable Gate Arrays (FPGAs) logic devices can realize self-adaptation within their reconfigurable logic fabric using Evolvable Hardware techniques. Since evolution is employed, the *Observer/Controller* has the task of detecting internal/external errors and well as initiating reconfiguration when necessary.

A widely known generic OC platform called the Autonomous System-on-a-Chip (ASoC) architecture proposed in [29] is depicted in Figure 5. The ASoC platform consists of two layers: the *Functional* Layer and the *Autonomic Layer*. The Autonomic layer contains *Autonomic Elements (AEs)* that are responsible for correct operation of the corresponding *Functional Elements (FEs)* present on the Functional Layer. Every FE such as CPU, RAM, and Network Interface has a counterpart Monitor, Evaluator, and Actuator component within the Autonomic Layer. The Autonomic Layer also consists of an *Autonomic Supervisor (AS)* that has no counterpart on the Functional Layer. The AS is responsible for the correct functionality of all AEs on the Autonomic Layer.

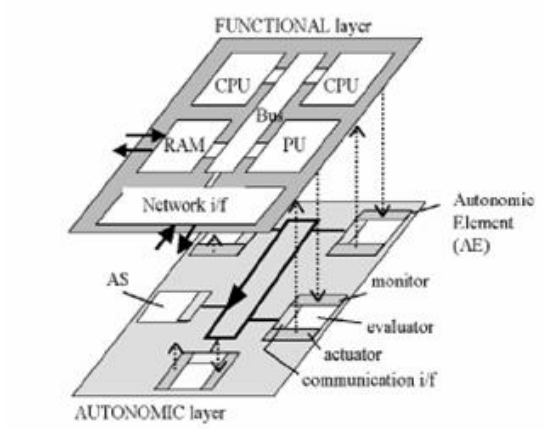

Figure 5: Dual-Layer ASoC platform from Lipsa *et al* [29]

The Dual-Layer ASoC design approach in Figure 5 is extended herein to provide fault coverage at both the Functional Layer and Autonomic Layer. This is achieved by assessing consensus among elements in a two-fold approach. Consensus is used first to realize failure detection. Once identified, consensus provides an organic method for fitness evaluation of competing alternatives during evolution providing a self-regulating approach to fault resolution. The measured performance is analyzed as an integrated OC system for self-configuration and self-healing. This demonstrates a generic OC architecture that can detect faults and refurbish itself while still providing a degraded level of valid throughput even during the online repair period.

1.5. Contribution of Dissertation

One point which comes through clearly from the previous discussion is that the fitness function is indispensable central composition of the GA process. It measurs the performance of different individual's structure and makes a trajectory through the possible solution search space under the successive GA operations. Conventionally, most GA applications such as function optimization and scheduling problems perceived their ultimate objectives prior to the design time. However, for the real time electronic device, the operation environment is full of unknown factors which may not be apparent to the designers. Even worse, the devices may be affected by a fluctuating operational environment during long execution times. Apparently, the static Fitness function cannot provide sufficient support for such applications. With FPGA devices in most mission-critical applications confront severe natural conditions, a new approach should be proposed and evaluated.

In response to the questions presented above, a consensus based Fitness evaluation approach is presented in this dissertation using population information and a new fault tolerance model which embedded both traditional TMR and Standby system and FPGA device reconfiguablity property in order to facilitate real-time competitive computing for autonomous regeneration of embedded reprogrammable model. An analytical software model is constructed to simulate the large-scale reconfigurable on-chip resources. Failures will be occluded by iteratively developed adaptive reconfiguration techniques in Extrinsic Evolvable Hardware. The most significant attributes and advantages are listed in Table 1.

Table 1: Attributes of proposed technique

Technique Terminology	Problem Domain	Attributes of Proposed Approach	Significant Contribution
Consensus Based Evaluation (CBE)	Genetic algorithm fitness function	Population-based evaluation which is driven by execution environment	First use of fitness function that depends on explicitly global population information and implicitly environmental information
Triple Modular Redundancy + Standby Model (TMRSB)	Reliability model	TMR with standby individuals in component-level	First proposed reliability model which takes advantage of the reconfiguration capacity in FPGA device
Evolutionary Organic Computing Architecture (OC)	Self-organization architecture	Autonomous fault-detection and self-recovery	New OC architecture with utilization of EHW idea
Specific Genetic Operators	Genetic operators	Genetic operators which can manipulate the configuration of SRAM-based FPGA	New operators which are specifically designed to facilitate SRAM-based FPGA genetic operation

This novel self-regeneration approach for embedded systems is based on CBE. Instead of using redundant spares to handle failures, it synthesizes fault-specific reconfigurations to regain lost functionality. Mechanisms of competitive computation are developed to achieve each Research Objective identified below.

1.5.1. Integrate Fault Detection, Isolation, Diagnosis, and Recovery phases

Real-time competitive computing approaches for autonomous regeneration of embedded reprogrammable model are developed and evaluated in this dissertation. An analytical software model is constructed to simulate the large-scale reconfigurable on-chip resources. Failures are occluded by adaptive reconfiguration techniques for Extrinsic Evolvable Hardware.

1.5.2. Realize Adaptable Quality of Service (QoS) Levels for Reliability

A novel self-adaptive population-based mechanism for all fault-handling stages in embedded reconfigurable devices is developed. This approach will *detect* faults by comparing outputs of competing configuration alternatives. By comparing discrepancies from alternative configurations, it is possible to *isolate* the failed physical resource. Remapping operators are then used to realize a failure-specific *refurbishment* during normal operations to make detailed physical failure mode *diagnosis* unnecessary. The refurbishment procedure will be realized using established mechanisms of *crossover*, *mutation*, and *deterministic guided search*. This objective will be realized with an experimental hardware-in-the loop research strategy.

1.5.3. Realize Online Device Refurbishment

By varying only an FEW parameters of the competition process, a wide range of reliability vs. overhead tradeoffs are obtained. Under the CBE technique, the costs of FPGA resource space

overhead, additional power consumption, and throughput delay incurred to support regeneration are *continuously variable*. Analytical modeling of these costs provides us not only a composite measure of system performance, but also feedback for adaptively reconfiguring FPGAs. Specifically, the *Evaluation Window Interval* and *Re-introduction Rate* can be updated according to recent discrepancy counts in order to maintain a suitable *Mean-Time-To-Repair* (MTTR) vs. *Mean-Time-Between-Failures* (MTBF) condition under adaptive and possibly hybrid control algorithms. To ensure system availability, parts need to be regenerated at a faster rate than they are failing such that a MTTR < MTBF condition is maintained. This objective of quantifying and optimizing the performance characteristics of the proposed CBE method will realize adaptive *Quality of Service (*QoS*)* levels for reliability via analytical modeling and advanced controls.

1.5.4. Proposed Self-Recovery Architecture

It is demonstrated, with the exception of catastrophic failures, how a device can be refurbished online without additional *function* or *resource* test vectors. This will be achieved by integrating competition and refurbishment wholly within the *FPGA's* normal data throughput processing flow. Because a fitness adjustment function is used that favors fault-free behavior, the *FPGA's* normal input data throughput stream can be used to evaluate fitness states while the device is under normal operation. The benefits of fitness adjustment functions based on *Binary Discrepancy* and *Hamming Distance* will be determined. This research will be conducted by developing a unified framework that involves techniques from three separate areas:

combinatorial analysis of the problem space and statistical analysis of fault occurrence versus population size.

CHAPTER 2: PREVIOUS WORK

2.1. Overview

This chapter will present a broad overview and survey of the techniques utilized in this dissertation including EHW applications under GAs, OC architectures, and fault tolerance strategies. The most advantage of above techniques is presented as well as the drawback in terms of their efficiency, overhead, and adaptive capacity. The successful EHW [14, 15, 18, 30] [16] applications are shown in Section 2.2 and the OC architecture [23] introduction follows and finally the TMR and Standby fault tolerance system are analyzed in terms of their own properties.

2.2. EHW Approaches to Increases Reliability

Several previous works describe other Regenerative Fault-Handling Techniques in EHW and how they attempt to actively restore mission-critical functionality in FPGA devices. They provide attractive alternatives to device redundancy for permanent degradation due to thermal fatigue, oxide breakdown, electromigration, and radiation-induced stuck-at-faults. Benefits of regeneration include fault recovery without the increased weight and size normally associated with spares. Also, failures need not be precisely diagnosed through external means, due to the intrinsic assessment of the remaining functionality on the device itself. Furthermore, a competitive regeneration approach enables failure-time synthesis of new FPGA configurations to handle failure scenarios that are unforeseen at design time.

Lohn, Larchev, and DeMara [14, 15, 31] develop an FPGA bit-string representation along with mutation and two-point crossover operators for actively refurbishing interconnection as well as logic resources. This related work demonstrated the complete regeneration of a Quadrature Decoder on a Xilinx SRAM-based Virtex XCV1000 FPGA. It shows that a stuck-at-fault on the input to a FPGA's Configurable Logic Block (CLB) can be occluded through reconfiguration. The Genetic Algorithm (GA) developed synthesizes a new alternative configuration using evolution in a population of 40 competing configurations after a few hundred generations. The GA is shown to recycle the damaged part as well. It was observed that partially-damaged CLBs were reassigned to new functions based on the residual functionality that could be utilized in the refurbished configuration. While achieving complete regeneration for modestly-sized circuits, refurbishment was performed offline and required exhaustive fitness test vectors.

Lach's deterministic approach segments the FPGA into static tiles at design time with a known functionality, some redundant resources, and a pre-designed alternate configuration. Spare tiles can be selected when needed, but their functionality is predetermined and thus limited. On the other hand, Roving STARS [18] is a resource-oriented dynamic online test approach that performs Built-in Self-Tests (BISTs) on roving sub-sections of the FPGA. Each portion is continually taken offline in succession and tested while its functionality moved to a new location. STARS' detection latency can be excessive since the tests must sweep through all resources. Also, STARS' power consumption and unavailability due to unnecessary reconfigurations when no faults have yet occurred can be prohibitive.

An alternative approach is taken by Keymeulen, Stoica, and Zebulem [30] using a design-time emphasis. They develop evolutionary techniques so that field programmable circuits are initially

designed to remain functional even in presence of various faults. Their population-based fault tolerant design method evolves circuits and then selects the most fault-insensitive individual. This method provides good resource coverage and passive runtime operation. The technique may be applicable for constructing a diverse initial population under our proposed CBE approach.

Table 2 addresses active *Fault Recovery* characteristics. Ideally, recovery would be performed with the residual functionality in faulty device remaining online whenever possible, but only STARS and CBE attempt this. Vigander's and Lohn's methods exhibit likelihood of recovery related to the FPGA's *design complexity*. In other words, they try to design an original repair where only a single failed configuration is available from which to learn from. Instead, CBE draws upon a diverse population to bias search towards regions of alternative configurations that are still operational. While the quality of recovery under evolutionary approaches cannot be guaranteed, static redundancy approaches like Lach's are either completely recovered or completely beyond recovery. STARS' quality of recovery is restricted by a fixed routing scheme that cannot adapt.

As listed in Table 2, several active recovery approaches support resource recycling, including the CBE. Under the CBE, the frequency of reconfiguration can be varied to tradeoff configuration overhead vs. recovery capability. With regards to pre-determined system recovery limits, only dynamic competitive approaches are truly restriction-free. While competitive and evolutionary recovery approaches have been demonstrated on small applications, the focus of this dissertation will be to extend the methods to larger, more useful circuits using improved techniques of the CBE with statistical, adaptive, and hybrid design methods of control.

In this dissertation, CBE utilizes an innovative *temporal voting* approach whereby the outputs of just two competing instances are compared. The presence or absence of a discrepancy is used to adjust the fitness statues of both individuals without rendering any judgment at that instant on which individual is actually faulty. The faulty, or later exonerated, configuration is determined over time when each individual is paired with other competing configurations under certain times. The competitive process is applied repeatedly to form a strong consensus across a diverse pool of alternatives. Under CBE, the FPGA's outputs are compared before they leave the chip so fault detection occurs on the first erroneous output and detection latency is negligible. A unique advantage of this competition-based approach is that it also permits coverage for active elements of the fault comparator itself by embedding an instance of the fault checker in each configuration. Fault isolation in the TMR, Vigander, and Lach approaches are restricted to coarse predefined granularities. Meanwhile, STARS attempts to isolate resource faults at only the very finest granularity. Alternatively, as in Vigander's and Lohn/Larchev/DeMara's approach, CBE does not require fault isolation of a particular granularity in order to achieve refurbishment. Under CBE, transients reduce instantaneous fitness values, but their effects are automatically attenuated over time so that unnecessary refurbishment is not triggered given a properly-selected *Threshold.*

Table 2: Fault Recovery Characteristics of Selected Approaches

Approach	Online Recovery	Basis for Likelihood of Recovery	Quality of Recovery	Availability	Externally-supplied Elements	Potential for Faulty Resource Recycling	Pre-determined Recovery Limits
TMR	No	Not addressed					
Vigander	No	Design complexity	Non-deterministic	Non-deterministic	GA Controller, function test vectors	Yes	None
Keymeulen, Stoica, Zebulum	No	Depends on characteristics at design time	Non-deterministic	Not addressed	None at runtime	No	Depends on characteristics at design time
Lohn, Larchev, DeMara	No	Design complexity	Non-deterministic	Non-deterministic	GA Controller, function test vectors	Yes	None
Lach	No	Available spares	Either complete or none	Either complete or none	Device test vectors	No	Only one faulty CLB per tile
STARS	Yes	Available spares	Restricted by non-optimizable re-routing strategy	~93% for ORCA FPGA	Test Reconfiguration Controller + device test vectors	Yes	Free STARS available and fixed routing chokepoints
CBE	*Yes*	*Recovery complexity*	*Optimized by second-order fitness metric scheme*	*Adaptable*	*Optional external RAM. Fault coverage for this RAM is intrinsic when configuration loaded. No test vectors used.*	*Yes*	*None*

2.3. Self-X properties on Organic Architecture

Related works in the literature have explored techniques useful for development of an OC system from various theoretical and practical perspectives. A frequent focus among these has been the design of OC architectures and OC development methodologies for systems with potential for exhibiting increased reliability and sustainability. For example, in [32] a runtime reliability

evaluation of ASoC architectures was addressed. The objective was to design SoCs that can tolerate faults by introducing dynamic reliability, power management, and security tradeoffs, as well as adaptation to environmental changes and unpredictable failure scenarios. Under these conditions, a theoretical model for calculating error probability during run-time is presented. A related fault model in [26, 33] concentrated on transient and timing faults caused by ionizing radiation or variations at the technology or device level. The C-program simulations executed on Leon-2 processor code resulted in a penalty of two cycles for the detection and correction of an error in the processor's pipeline. Work has also been conducted on prototyping platforms capable of support OC architectures. For instance, the *Egret* system provides a platform for reconfigurable SoC's supporting applications such as OC [34]. The Design objectives of the Egret is to provide a platform that students can use to rapidly prototype new reconfigurable, embedded computing application and the second objective is to provide a straightforward path to commercialization of prototyped designs. The platform consists of modular functional elements that can be interconnected to design an embedded application for reconfigurable logic.

From the design methodology perspective, various previously-proven agent concepts were combined into a system-level design approach for OC development. This is presented in [27] which developed an adequate, model-driven software engineering methodology based on the Unified Modeling Language (UML) and Model Driven Architecture (MDA). The model was applied and tested on a manufacturing control system which exhibited various self-x properties. In [28], Observer/Controller architecture was developed to provide a generic template to develop OC systems. The template was used to implement the control of an urban traffic network.

While maintaining area/performance and power requirements, Avizienis [35] suggests integrating biology inspired concepts into the integrated circuit design process with the main objective being achievement of higher reliability. The immune system which was proposed continuously identifies and handles problems either internally or with the aid of external agents. In more a general study, identification of SoC system requirements for detecting faults and handling the faulty components is addressed in [26]. Fault tolerant error detection techniques are classified into three groups: hardware redundancy, information redundancy, and time redundancy. The three techniques and their combination are surveyed on Autonomous SoC design consisting of the two layers: the Functional Layer and Autonomic Layer. In this theoretical research framework, it is suggested that the Autonomic SoC would need a well-tailored AE layer which would cope with malfunctioning subcomponents. The simulation consists of a paradigm with priori knowledge about the system's behavior when an error occurs and examines setting a threshold for errors that can occur before the system goes into self-organizing mode [26].

In order for an autonomous system to invoke its self-healing mode, it must be able on its own to detect errors during run-time [36, 37]. Reconfiguration and detection techniques explored include *scrubbing* which is the continuous reconfiguration of the bitstream to refresh the stored configuration [38], Built-In-Self-Test (BIST) techniques [39], on-chip hardware test benches [40] and Triple Modular Redundancy (TMR) [41]. Decentralized approaches to Observer/Controller units can be preferable in the design of fault-detection and self-healing systems due to the fact that the observer/controller system itself might be faulty [36, 37], and this is one focus of the OES Architecture described in Chapter four.

For realization of the recovery phase, Genetic Algorithms (GAs) have been applied to FPGA devices in various approaches. In the cases of intrinsic hardware evolution, the GA is invoked to apply crossover and mutation on the FPGA bitstream to evolve a fault-specific repair in-situ on the device. A software-simulation study of this approach was presented in [42]. It also explored the use of voting systems that operate in parallel despite imperfect GA solutions to refurbishment of local permanent damage in the FPGA fabric. Results showed improvement in aggregate repair performance from several different incomplete repairs obtained by the GAs. In [28], an autonomous self-repair approach for SRAM-based FPGAs is developed based on Competitive Runtime Reconfigurability. This approach was applied to a FPGA-based multiplier design which demonstrated evolution of a complete repair for 3x3 multiplier from several stuck-at-faults within a few thousand iterations. Using conventional offline population based approaches, GAs were also explored in [31] and [14, 15] for evolutionary fault recovery in Virtex FPGAs using an external controller and an offline repair process.

Other examples of OC architectural approaches include an OC system developed for face recognition [24]. The system utilizes some characteristics of an OC system such as self-organization and robustness. Methods for recognition of an input face from variation of images based on learning from biological systems are discussed. Others have argued that neutrality is a necessity for optimal self-adaptation [43]. They emphasize the need to provide a unifying formalism to embed approaches to self-adaptation in evolutionary computation.

2.4. TMR and Standby System Application on Improving Reliability

The TMR approach, first proposed by Von Neumann [19], is shown in Figure 1. It was widely used in software fault tolerance [3] and reliable hardware [30] applications. The primary drawback of the TMR approach is resource overhead. The TMR design triples the area and power consumption of physical resources over a simplex design. Duplex systems with a hot standby component based on a process pair [14] paradigm for fault tolerance are widely implemented in Network Access Devices (NAD) [15] and other uninterruptible operational systems.

However, much of the superiority of TMR and Standby type systems hinges upon some critical components. The reliability (or lack or reliability) of the majority voter in TMR systems and the Standby system switch mechanism may be detrimental to the overall system reliability. There are other issues to consider including: the reliability of memory which stores the standby configurations, the capability of sensing improper operation to trigger a switch, or how the majority voter and the switch operation must maintain data consistency between the primary and backup components.

Several previous works on TMR systems for FPGAs are introduced in [10] [30] and [13]. In [10], the TMR system with voting technique is combined with bitstream scrubbing implemented in a Virtex FPGA device in order to mitigate Single Event Upset (SEU) effects [44]. The voting mechanism identifies the faulty configuration based on single failure assumption and reconfigures (scrubs) the device with an alternative bitstream. However, the reconfiguration has

to take place off-line and can only deal with a transient faults which can be restrictive for use during deep space missions.

Fault detection characteristics relevant to embedded FPGAs are presented in multiple approaches. A traditional approach to fault-handling such as *Triple Modular Redundancy* (TMR) utilizes a fixed pool of three identical device resources. Under TMR, only the majority vote of three outputs is propagated, realizing online fault handling with negligible detection latency. Vigander's [16] approach extends TMR-style voting to utilize faulty FPGAs that have been partially regenerated using evolutionary algorithms. He demonstrates that FPGA-based implementations of 4-bit x 4-bit multipliers can be automatically reconfigured to realize partial refurbishment. Yet since each partially refurbished multiplier is deficient with respect to only certain input pairs, a voting arrangement of partially refurbished parts exhibits complete regeneration of the lost functionality. TMR, Vigander's, and other n-plex spatial voting approaches can deliver real-time fault detection, but also increase power consumption n-fold during fault-free operation and insert a critical voting element into the reliability path.

A TMR application for the Virtex series of Xilinx FPGA is described in [45]. The Majority voter is implemented with tri-State buffers based on the Virtex bus structures. Different types of data structures such as Throughput Logic, State-Machine Logic and I/O Logic are illustrated in terms of a TMR technique. Some special features provided by the Virtex architecture are also mentioned.

Another analysis of the TMR with mitigation of SEU effects in the Xilinx FPGA device is [46]. A selective TMR architecture is implemented for sensitive portions of the circuit in order to

harden against the SEU effects. However, as the authors mentioned in the conclusion section, the result of STMR is based on the input signal probabilities and nature of the circuit and may only be beneficial to the circuit with input environments where the size of the SEU sensitive portion is smaller than the original one. Such an approach narrows down the application range and can not be viewed as useful for general utilization in different kinds of circuit design.

Furthermore, an analysis of the SEU effects in the TMR architecture in [47] shows that TMR may not be sufficient to harden a circuit. The results presented show most of the faults escape the TMR architecture. They proposed a smart floorplan for the placement and routing which may improve mitigation of SEU effects using TMR.

A VHDL design methodology for redundancy in combinatorial and sequential logic research is developed in [48]. A VHDL approach has been developed for automatic TMR insertion and demonstration in order to mitigate the SEU effects. Both module level mitigation and gate level mitigation are discussed.

All the above enumerated techniques or architectures based on electronic embedded system have their own advantage and restriction in terms of different applications and different system performance requirement. Based on current techniques and architectures, we proposed a new technique CBE approach in order to cover some of disadvantage of previous approaches such as constant fitness evaluation, online repair, and specific architecture-oriented GA operations through evaluation multiple benchmark circuits. Also a new proposed OC architecture is shown in this dissertation to utilize either the Lispa's layered OC concept [29] in conjunction with the CBE technique.

CHAPTER 3: TMR, STANDBY AND TMRSB SYSTEM

Despite continued improvements in reliability at the component level, fault tolerance strategies still retain an essential role for applications that require high reliability in environments with unpredictable adverse effects. Fault tolerance strategy utilizing redundant components have a variety of architectures that can be used to obtain higher system reliability. Many previous fault tolerance approaches such as Triple Modular Redundancy (TMR), Simplex/TMR and Standby systems were extensively covered in literature [19] [35] [49] [50] starting in the 1950s. In recent decades new types of electronic devices have become available, such as reconfigurable hardware that has allowed some inefficient strategies, which were never considered or implemented before, to become viable due to the unique characteristics of such devices.

Consider the variety of embedded computing environments which frequently occupy harsh and difficult-to-regulate surroundings with thermal, mechanical or acoustical stress. In addition, space or avionic applications may also face very high levels of radiation exposure. Higher reliability systems required for long duration missions have, in most cases, limited capabilities for interactive diagnosis, repair and onboard spares. These systems must count on system level fault tolerance strategies even though implemented with high reliability components.

Furthermore, along with the finer granularity of the electronic device, the measurement of the system/component reliability may not satisfy the evaluation of the current implementation scenario and restrict ad-hoc repair strategy as well. The concepts of residual functionality after fault and autonomous repair are receiving increasing affection beyond traditional fault tolerance

techniques. Addressing these new considerations may improve not only the system reliability, but can be achieved in parallel with the system throughput without human intervention.

3.1. Overview of Traditional Fault Tolerance Strategy

The TMR approach, first proposed by Von Neumann [19] is shown in Figure 6. It was widely used in software fault tolerance [49] and reliable computer architecture [35] and Evolvable Hardware design[50]. The utmost drawback of the TMR approach is resource overhead which will increase by 200% the area and power consumption of physical resources over a simplex design and introduce the extra voting components which introduce new the vulnerability of the system. This may be infeasible to a system with limited payload capacity such as space application.

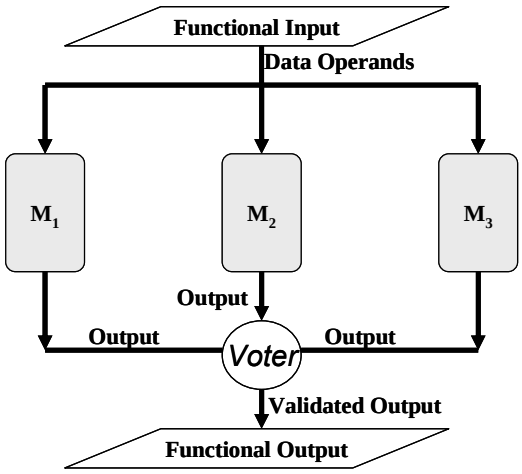

Figure 6: TMR System

The approach combines time and spatial redundancy by applying time redundancy to TMR systems. For the permanent fault, a reconfiguration will be implemented on either all of three

instances or just the failed module. For the transient fault, a data roll-back will be implemented by re-computing the task without replacement. However, in order to obtain accurate detection, the TMR system needs an extra vote which induces higher overhead. The Markov Chain model was utilized in this dissertation to analyze the system reliability and availability.

The conventional *N* modular Redundancy (NMR) [51] system provides a powerful approach of improving reliability and fault tolerance capacity of digital systems. *N* functional modules, $N=2m-1$ and $m>1$, implemented identically, are given concurrent computation tasks and utilize a majority voter on the output to obtain the final result whenever at least *m* modules are functioning correctly. Each module is identical in functionality, but fault independent and may have a different physical implementation or design in order to minimize fault impacts such as Common Mode Failure (CMF) [52]. The arbitrary fault can be masked by the majority voter without sudden performance degradation except in the case of catastrophic failure. Among NMR approaches, TMR [19] [41] has been one of the most popular fault-tolerance schemes using spatial redundancy in a practical system. In Figure 6, the three functionally identical modules *M1, M2, M3* are deployed in parallel and the outputs converge at the majority voter to obtain the validated output for the system.

Another fault tolerance strategy is a Standby System (SB) arrangement. A Standby Model refers to the case in which a primary component (or system) has one or more identical backup components in an "off" or "off-line" state. When the original active component fails, a switch mechanism selects one of the "Standby" backup components and makes it the new active component. The system continues to operate with execution effected only by switching overhead.

Duplex systems with a hot standby component based on process pair [53] paradigm for fault tolerance are widely implemented in Network Access Devices (NAD) [54], Web Server Systems (WSS) [55] and other uninterruptible operational systems. However, the "Hot" standby component will be active and have same fault probability as the current operating component even though the switch may have less impact on the system performance.

According to the backup component states, three varied types of standby system are defined. The "Hot" standby is keeping the primary and secondary (backup) components running simultaneously with the backup tracking the primary system in real time. This will allow a seamless switch when a fault in the primary component is detected. The "Cold" standby system is a method in which the secondary component is only called upon when the primary component fails. Between the "Hot" and "Warm" standby system, the "Warm" standby system will periodically mirror the primary component which means that there are times when both components do not contain the exact same data. As shown in Figure 7, the standby configuration can be in Hot, Cold, and Warm states depends on the specific system design.

However, much of the superiority of TMR and Standby type systems depends on some key components. The reliability (or lack or reliability) of the majority voter in TMR systems and the Standby system switch mechanism may be detrimental to the overall system reliability. There are other issues to consider like the reliability of memory which stores the standby configurations, the system power supply, the capability of sensing improper operation to trigger a switch, or how the majority voter and the switch operation must keep data integration between the primary and backup components.

Redundancy techniques are widely used in different applications. One example would be improving transmission rates of a communication system by expecting packet loss, duplicating, and reordering the corrupted data. Power plant stations and the power supply grid use redundant generators or power supply networks to continue to provide power in case of an emergency. It is also well known the reliability of digital system can be improved through the appropriate arrangement of additional components. High reliability and availability are particularly sought after in mission critical system.

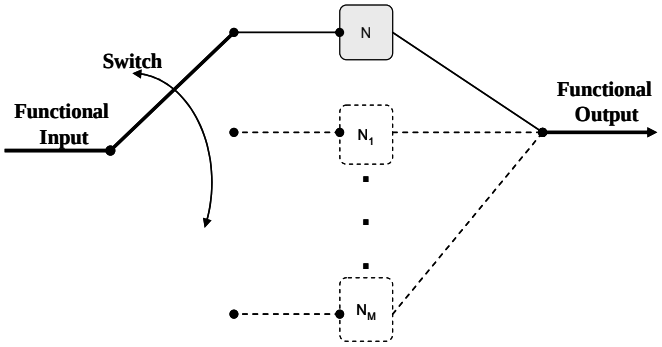

Figure 7: Standby System

3.1.1. Embedded Device Properties Influencing Redundancy Strategies

As the application scope of digital system have extended into science and engineering fields, a strong desire for operational fault tolerance has developed especially in mission-critical equipment. The particular requirement of fault tolerance and fault repair has to be compatible with the specific characteristics of a digital device in order to obtain the practical benefit. On the

other hand, new up-to-date devices which have unique characteristics can also be catalysts to develop new fault tolerance structures as is the case in this chapter.

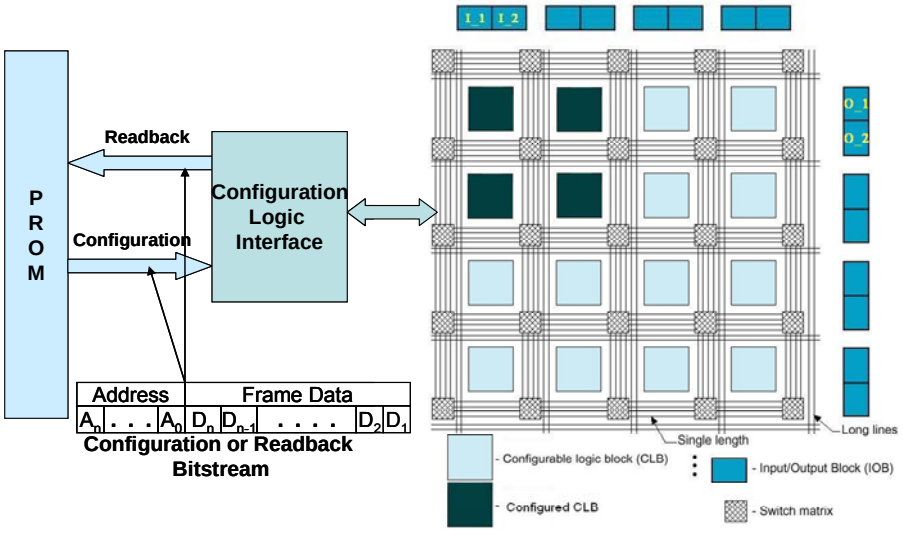

Figure 8: FPGA Configuration and Readback Mechanism

SRAM-based reprogrammable devices known as a Field Programmable Gate Arrays (FPGAs) are large multi-million gate-equivalent devices that employ these technologies extensively. Over 100 FPGA devices can be embedded in a mission-critical system. The FPGA configuration is stored in bitstream format in the PROM and loaded into or read back from the FPGA chip through Configuration Logic Interface shown in Figure 8. The different connections on the FPGA chip integrate the Configuration Logic Blocks (CLBs) or Look Up Tables (LUTs) to implement computation logic tasks.

Environmental challenges to reliability in space applications can be modeled as having a uniform failure rate exposure despite status and locations of device activity in the system. Therefore, the impact of device wear-out (active components vs. cold spares) is small relative to radiation exposures, which makes ambiguous the active vs. standby role in terms of reliability in the

various standby models. In other words the radiation effects far outweigh device aging effects and because both active and standby components are exposed to radiation equally their lifespan is primarily and equally determined by the effects of the radiation environment.

The pertinent reliability exposures for embedded FPGA's include hot carrier aging, ultra-thin gate oxide breakdown, and electromigration effects. FPGA's now utilize deep submicrometer (0.13 μm) CMOS technology. As geometries and supply voltages shrink and electric current densities raise, increasing interconnect failure rates caused by high current electromigration can be observed over long product deployments.

Several previous works on TMR system in the FPGA are introduced in [56]. In [56] [45],[46],[47] and [48], the TMR system with voting technique combine with bitstream scrubbing implemented in a Virtex FPGA device in order to mitigate *Single Event Upset* (SEU) effects. The voting mechanism identifies the faulty configuration based on single configuration failure assumption and reconfigures (scrubs) the device with an alternative bitstream. However, the reconfiguration has to take place off-line and can only deal with a transient fault which maybe inappropriate for a practical system.

A TMR logic generation control log for the Virtex series of Xilinx FPGA is described in [45]. The Majority voter is implemented with tri-State buffers based on the Virtex bus structures. Different types of data structures such as *Throughput Logic*, *State-Machine Logic* and *I/O Logic* are illustrated in terms of TMR technique. Some special features provided by the Virtex architecture are also mentioned. The attached example uses the XVRWARE synthesis library

which provides macros and synthesis for constructing TMR circuits in VHDL for the Virtex architecture.

Another analysis of the TMR with mitigation of SEU effects in the Xilinx FPGA device is [46]. A selective TMR architecture is implemented for sensitive portions of the circuit in order to harden against the SEU effects. However, as the authors mentioned in the conclusion, the result of *Selective TMR* (STMR) is based on the input signal probabilities and nature of the circuit and may only be beneficial to the circuit with input environments where the size of the SEU sensitive portion is smaller than the original one. Such an approach narrows down the application range and can not be viewed as a general approach in different kinds of circuit design.

Radiation-induced Single Event Upsets (SEUs) can produce soft failures in both the configuration memory itself and in the mapped circuit on the throughput data-path. In addition, changes induced to the configuration memory not only change the circuit memory but can change the functionality of the mapped circuit as well. Given the architecture of FPGAs, the two different types of failures can have equivalent effects. The result of a SEU that makes the device totally or partially lose functionality is generally defined as Single Event Functional Interrupt (SEFI) [44]. In order to accurately evaluate the SEU effect, a stuck-at fault model is used in this dissertation for simulating single and multiple failure scenarios.

FPGAs are the ideal platform for reliability models like NMR and SB. Their unlimited reprogrammable property makes the standby components switches feasible with low delay and overhead. Furthermore, the reprogrammability enables designers to consider the appropriate recovery mechanisms which can extend mission lifetime compared to the non-repair system.

After all, the millions gates capacity makes more physical resources reusable and provides more alternative space for rearranging the routing.

Autonomous repair of FPGAs is of particular interest in aerospace applications for both in-flight and Ground Support Equipment devices. Several advantages drive the FPGA as an appropriate platform for the spacecraft electronics. First of all, high flexibility in achieving multiple requirements such as high performance, low Non-Recurring Engineering (NRE) costs and fast turnaround allow systems to be made in a more efficient manner. Second, FPGA devices can be utilized in remote hard to maintain systems such as satellites and space probes and can allow for remote reconfiguration and repair without too much overhead while maintaining performance.

The emerging field of autonomous repair has essentially impacted deployable systems for deeper space exploration mission and other high availability, sustainability and serviceability application that need to survive and perform at optimal functionality during long duration in unknown, harsh and/or changing environment. Many techniques have been developed to generate the pre-complied alternative fault tolerance configurations and stored in memory or generate new fault tolerance configurations after a permanent fault is detected in order to reconfigure when a fault occurs.

Frequently, such systems have limited capacity for spares yet still have requirements for reliable operation over long lifetimes [50]. This dissertation approach in this chapter is to design and implement a hybrid system redundant architecture to handle a wide range of transient faults through automatic FPGA reconfiguration and also permanent failures though automatic selection

from a diverse set of standby components, which implement identical functionality, but may use different physical resources, and dynamic update of these alternative configurations.

3.2. System Reliability Analysis

3.2.1. Standby System

Consider the SB system configuration depicted in Figure 7. It contains $m+1$ identical component of which exactly one is active at any time and the remaining m components act as switchable spares. Up to m of these spares may provide feasible alternative standby configurations in order to extend the mission time.

A simple Standby system with only one component X_i (i=0, 1, 2…m, which include one active and m standby components) will be investigated in this case. The components are modeled with an exponential failure rate λ. Assuming that the de-energized components do not operate until a fault is detected on the active component, or otherwise dictated by the reloading schedule, the lifetime which is time to failure, Z, of such system can be characterized in term of the lifetime, X_i, of each individual configuration $Z = \sum_{i=0}^{m} X_i$.

Initially, assume the switch mechanism is completely reliable and all of the standby configurations are fault-free. To model the reliability of a standby redundancy system with a total $m+1$ independent configurations, we first identify the probability distribution by considering the case when $m=1$ where each component has an exponential distributed lifetime with parameter λ.

Let X_i and $X_{j\neq i}$ be random variables denoting the independent failure of each component. Assuming an exponential distribution given by the parameter λ, then pdf function is $f_{Xi}(t) = \lambda e^{-\lambda t}$, $t > 0$. Since $Z = X_i + X_j$, the density of the sum of two non-negative independent random variables is given by the convolution of the individual densities [57], we have:

$$\begin{aligned} f_Z(z) &= \int_0^Z \lambda e^{-\lambda t} \lambda e^{-\lambda(z-t)} dt \\ &= \lambda^2 e^{-\lambda t} \int_0^z dt \\ &= \lambda^2 z e^{-\lambda t} \quad , \qquad z > 0 \end{aligned} \tag{3.1}$$

Thus Z has a two-stage Erlang distribution [57] for the m=1 case and m-stage Erlang distribution, in general. Thus, for the m=1 case, the failure distribution function of Z is given by:

$$\begin{aligned} F(t) &= 1 - \sum_{k=0}^{m} \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad , t \geq 0, \lambda > 0, m = 1 \\ &= 1 - \left(1 - \lambda t e^{-\lambda t}\right) \\ &= \lambda t e^{-\lambda t} \end{aligned} \tag{3.2}$$

Then the m>1 reliability function is obtained by

$$\begin{aligned} R_{S\tan dby}(t) &= 1 - F(t) \\ &= \sum_{k=0}^{m} \frac{(\lambda t)^k}{k!} e^{-\lambda t} \\ &= e^{-\lambda t} + \sum_{k=1}^{m} \frac{(\lambda t)^k}{k!} e^{-\lambda t}, \ t \geq 0, \lambda > 0, m = 1,2,\ldots \end{aligned} \tag{3.3}$$

In Equation (3.3), $e^{-\lambda t}$ term represents the reliability of the initially-selected active component. The subsequent summation term in Equation (3.3) represents the probability that each standby component will provide a viable alternative. For example, suppose the initial active component fails and one of the standby components becomes energized to maintain the system availability. In this case, the summation of the reliabilities of all such replacements plus the initial component reliability determines the system reliability.

3.2.1.1.Imperfect Switching

Because the standby configurations are stored in non-volatile memory (e.g. EEPROM) and the circuits they describe are mapped into SRAM based FPGA architecture, we need to assume the standby individual failure status is unknown until they are selected for operation. Such a system is known to possess *standby redundancy* [54] in contrast to a system with *parallel redundancy*. In cold standby mode, the alternative configurations are in a power-off condition. In warm standby mode, they undergo periodic reloading and inspection.

There are varied distinct kinds of scenarios for the imperfect switching mechanism based on distinct standby strategies. For the cold standby system, the detection and switching function only works at time of failure and for the warm and hot standby system, the system is bound to have continual or periodic monitoring and detection.

However, the specific characteristics of the space application mentioned eliminate the variety on the different standby approaches. Two distinct scenarios should be considered in the FPGA case.

The first is unknown states of the standby configurations, the second is the imperfect switching case.

A few assumptions have to be made before further analysis since failure of the switch mechanism will cause the whole standby system cease operation permanently. Faults in an active configuration will simultaneously disable that configuration and trigger one switch.Each switch can cause a recovery from one or more failures. There are always enough fault-free standby configurations in the standby pool.

For the imperfect switching scenario, we introduce the term q as an observed success probability of switching to accommodate the reconfiguration process and u as the number of the successful switches before the switch failed. Prior to switch failure, all required switches were successful, and after switch failure, no switch function will work anymore. The probability that the entire system fails due to switching failure, in response to the component failure, can be model as a geometric random variable with probability mass function of $q^u(1-q)$.

Therefore, the reliability function of a standby system with an imperfect switch includes the influence of the probability q of each standby being successfully selected:

$$R_{StSw}(t) = e^{-\lambda t} + q^u \sum_{k=1}^{m} \frac{(\lambda t)^k}{k!} e^{-\lambda t}, \ t \geq 0, \lambda > 0, m = 1,2,\ldots \tag{3.4}$$

Thus, only after a failure in the initial active configuration is detected, can switching be implemented and the switch probability will add into the second term of the Equation (3) to obtain the Equation (3.4).

The number of the successful switches determines the system feasibility and, according to the assumption 3 above, u will always less than m, which will make m-u number of standby configurations without any impact on the Equation (3.4). Therefore, we can draw figure4 based on u=m to show the $R_{StSw}(t)$.

According to the Figure 9, the reliability of imperfect switching is not a linear increased with the number of the standby configurations. That is because the more configurations may bring more switching overhead when more fault occurred in the system. So in the later analysis, we use u=0.9 and m=4 as the optimization data set.

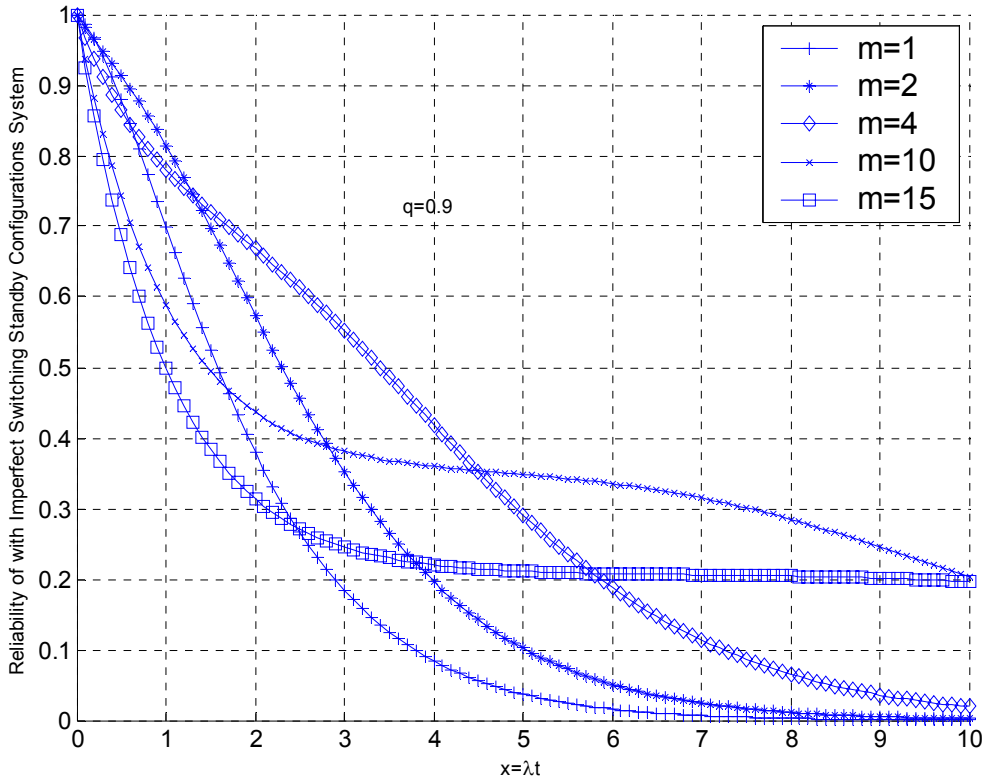

Figure 9: Reliability of Imperfect Switching Standby System

3.2.1.2.Unknown Configurations Status

Wherever the standby configurations are stored and whatever state they are in, radiation may cause the same affect on them same as on the active elements. Even with the perfect switching, a faulty standby configuration will generate an unexpected output. Faulty standby configurations will be detected when they are online and the switch mechanism will keep loading alternative backup configurations out of the standby pool until a fault-free one is running. When the fault-free configuration is loaded as active the one, the selection will be end until next fault occurs and impacts the current active one.

Because of the unknown status of standby configurations the probability that the system fails due to a switch to a standby configuration with a faulty configuration is follow the number of failures before the first success, supported on the set { 0, 1, 2, 3, ... }. It can be modeled as a geometric random variable with probability mass function of $p\ (1-p)^v$ in which v is the number of the failure selection trails ($v<m$) and p is the probability of success on each trial.

Assuming the survival rate p follows an exponential distribution and the selection process is a binomial distribution, and based on equation (3.4), the reliability for standby switching R_{StSw} is given by:

$$R_{StSw}(t) = e^{-\lambda t} + (1-p)^v \sum_{k=1}^{m} \frac{(\lambda t)^k}{k!} e^{-\lambda t},\ \ t \geq 0, \lambda > 0, v < m, m = 1,2,\ldots \qquad (3.5)$$

The number u of the standby configurations will yield to the dominator of the successful switch number in the Equation (3.5). Therefore, the Equation (3.5) becomes:

$$R_{StSw}(t) = e^{-\lambda t} + q^u (1-p)^u \sum_{k=1}^{u} \frac{(\lambda t)^k}{k!} e^{-\lambda t},\ \ t \geq 0, \lambda > 0, u = 1,2,\ldots \qquad (3.6)$$

As the number of standby configuration m is increased, will continue to decrease and converge to some constant value as depicted in Figure 10. The setting is set u=4, p=0.9, q=0.9. Figure 10 shows R_{StSw} as the time to failure, is increased for various values of m is increased. Once u is increased to a certain level, the improvement in system reliability levels off, implying that a sufficient pool of standby modules can provide adequate performance compared to using an infinite number of standby modules.

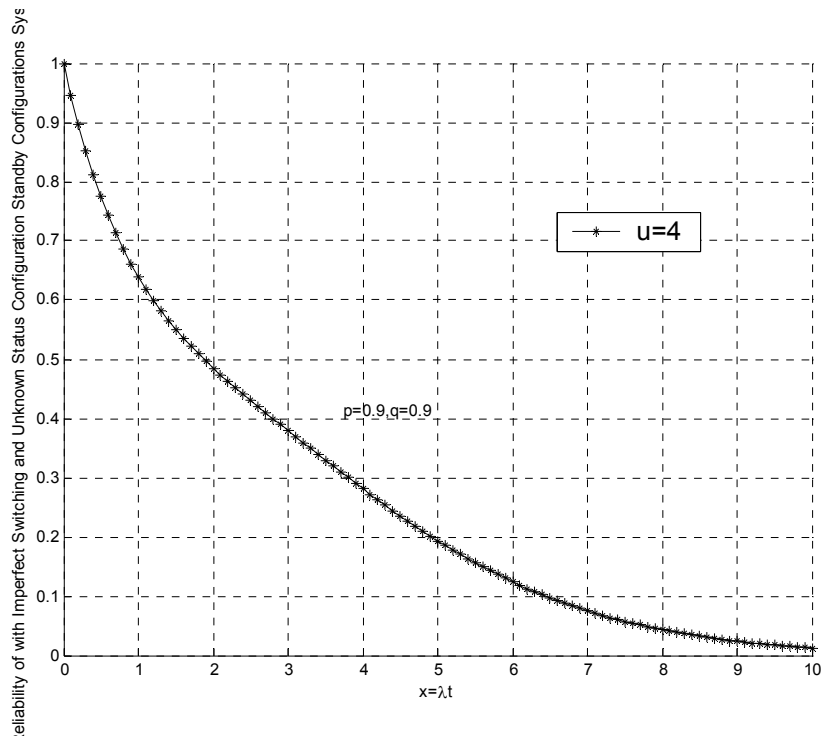

Figure 10: The Standby System with Imperfect switching and Unknown Configuration Status

3.2.2. NMR System

A general treatment of *NMR* system was developed starting in the 1950s [19]. Most of them assume a perfect voter in the system, and the reliability expression is based on binomial distribution given by:

$$R_{NMR} = \sum_{i=k}^{n} \binom{n}{i} p^{i} (1-p)^{n-i} \tag{3.7}$$

If each component follow an exponential distribution $p = e^{-\lambda t}$, then the

$$R_{NMR} = \sum_{i=k}^{n} \binom{n}{i} e^{-\lambda t i} \left(1 - e^{-\lambda t}\right)^{n-i} \tag{3.8}$$

In which R_{NMR} is equal to the system reliability R. Then, the Reliability of TMR system is $R_{TMR} = 3e^{-2\lambda t} - 2e^{-3\lambda t}$. In Figure 11, the different NMR system based on exponential distribution is presented. The cross point in figure is $\lambda t = 0.7$ which is obtain from let $3e^{-2\lambda t} - 2e^{-3\lambda t} = e^{-\lambda t}$.

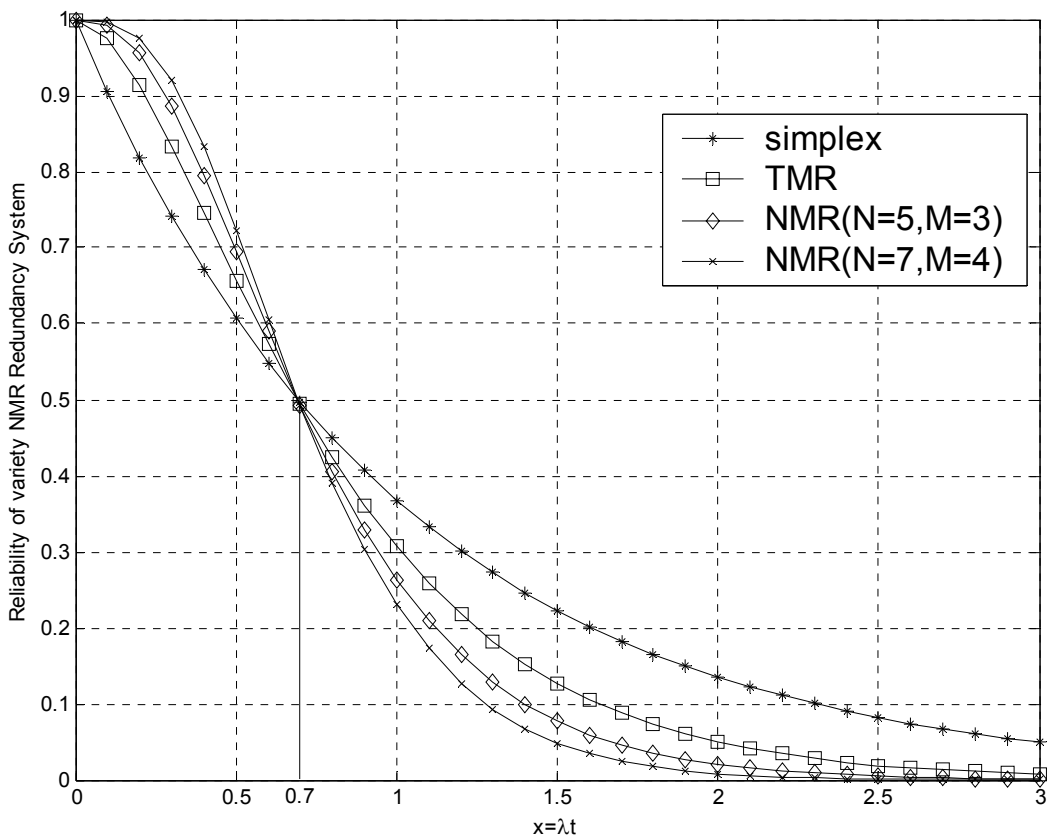

Figure 11: NMR System

3.2.3. Hybrid System

The TMRSD system in Figure 12 embeds the Standby system into the TMR framework in order to achieve the higher reliability and maintainability for the design. The system can be viewed as three functionally identical parallel subsystems with a majority voter, and each subsystem has *m-1* number of standby components. Components in this case are defined as functionally identical subsystems that utilize varied physical resources. To simplify the computation, we only consider the same number of standby components for TMR subsystems.

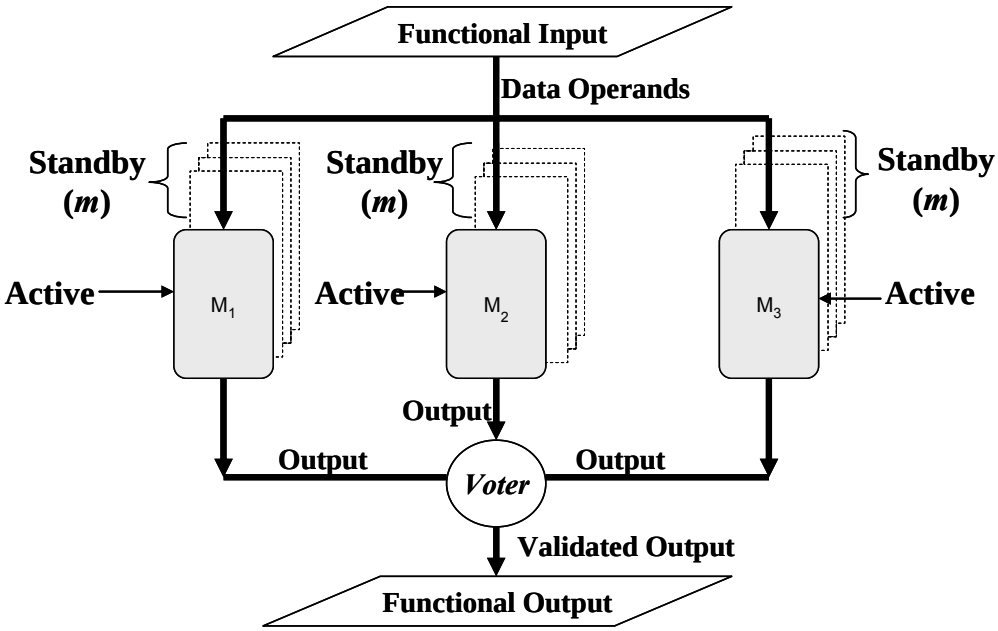

Figure 12: TMRSB System

$$\begin{cases} R = \sum_{i=k}^{n} \binom{n}{i} R_{S\tan dby}{}^{i} \left(1 - R_{S\tan dby}\right)^{n-i} \\ R_{StSw}(t) = e^{-\lambda t} + q^{m-1}(1-p)^{v} \sum_{k=1}^{m-1} \frac{(\lambda t)^{k}}{k!} e^{-\lambda t}, \quad \rho = 0.9,\ t \geq 0, \lambda > 0, m = 1,2,\ldots \end{cases} \tag{3.9}$$

The relabilities of different types of redundant systems are presented in Figure 13. Compare TMR vs Simplex and TMRSD vs Standby system, the similar comparison result are presented on the Figure 13. The TMRSD system improves the reliability only for the limited period time which can be utilized in short time mission.

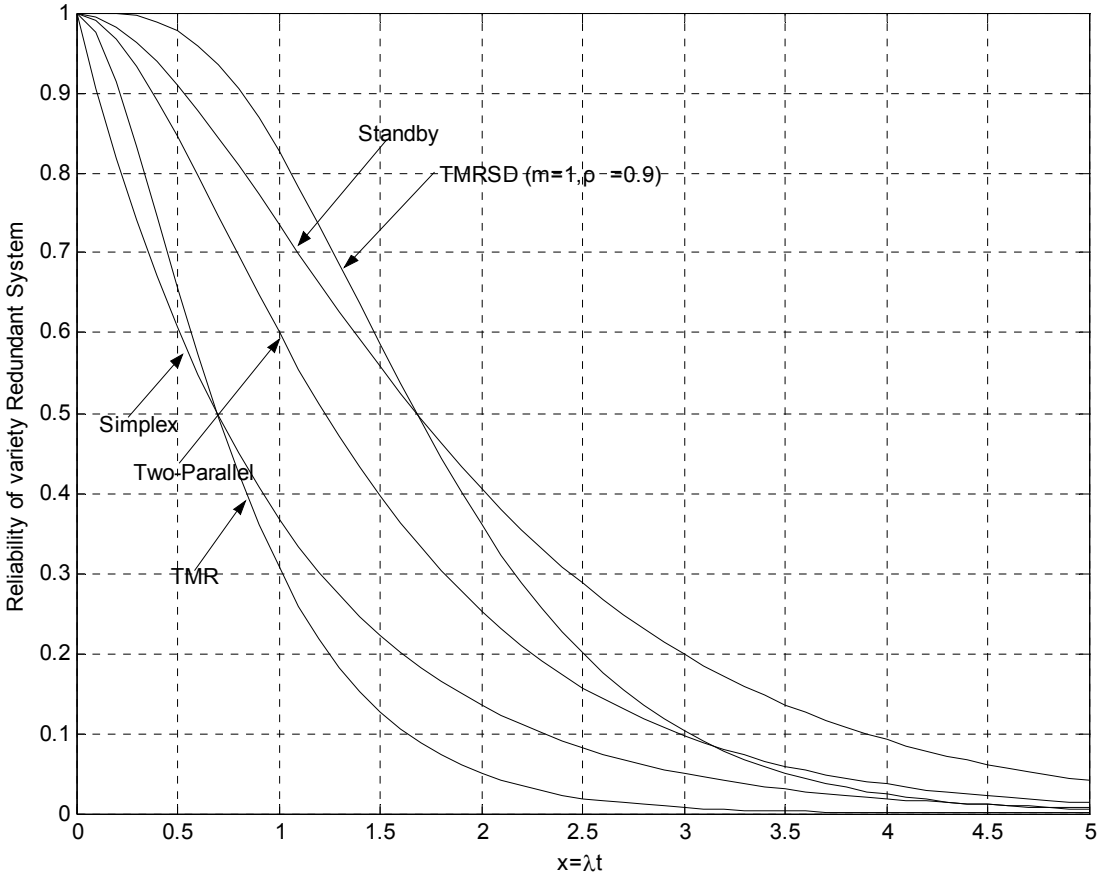

Figure 13: Comparison of Simplex,TRM,Two-Parallel-Redundancy,Standby,TMRSD Reliabilities

However, because $x=\lambda t$, when the λ is very small, the time t can be varied. This means the component reliability is essential factor of the system performance. Furthermore, this analysis shows that the system level reliability is based on the basically reliable components. In another word, the redundancy technique may not improve or even worse, the system reliability based on unreliable components.

Table 3 Performance Characteristics of FPGA-based Fault Tolerance technique

	Resource Utilization	**Power Consumption**	**Additional Latency**	**Failure Tolerance**
TMR	*3n*	3*n*+voter	voter	*1/3*
Standby	*n + S*	*n*+switch(m)	switch	*m/n*
Simplex	*n*	*n*	None	*n*
TMRSB	*3n+S*	3n+voter+switch(3m)	voter + switch	*3m/3n*

In Table 3, *n* represents the active resource set and *S* is the set of resources required to hold the standby configurations. So, for example, three active sets of resources are required for TMR configurations and one active resource plus *m* number of standbys held in *S* are required for standby. For the power consumption, TMR will require the power for the three sets of resources, *n*, plus consumption for the voter. For Standby the power requirements will be for the single active resource and possible m times switching if fault occurs. Different approaches may add different latency in term of the variety mechanism, for TMR the voter is vulnerable but critical path on the computation and cause the evitable latency, and the switch latency is a conditional latency based on the occurred fault numbers.

3.3. Simulation Result

BlockSim 6 offered by ReliaSoft was used in the dissertation. It allows you to analyze any process or product to obtain exact system reliability results (including system reliabilities, mean times, failure rates, etc.), to calculate the optimum scenario to meet system reliability goals and to obtain maintainability, availability and throughput results through discrete event simulation. BlockSim's blocks can be defined with the reliability characteristics of each component of the process or product. You can then configure these blocks into a reliability block diagram (RBD) that represents the reliability-wise configuration of the system and analyze the diagram in order to determine the reliability function (cumulative density function or cdf) of the entire system.

Another feature in BlockSim is use container to emulate the Standby scenario with Switch Probability on per request. In most cases, the reliability of a switch is to be included in the analysis the probability of the switch performing the action (i.e. switching) when requested to do

so. This is called "Switch Probability per Request" in BlockSim and is expressed as a static probability (e.g. 90%).

On the simulation, the exponential distribution is used in the experiment. According to the above discussion, we assign the same distribution on the both active and standby configurations. According to the Table 4, we can see the simulation results are corresponded to the section 3.2 analysis, even with the standby configurations number m increased, the system reliability may not improve. Meanwhile the system reliability will improve with the higher configuration reliability.

Table 4 Stadnby System Simulation Result

Simulation 50000hours	Perfect Switch		Imperfect Switch (90%)	
Standby # (m)	MTTF(hours)	System Reliability	MTTF(hours)	System Reliability
m=2	10000	78.89%	10000	51.54%
	20000	84.71%	20000	78.40%
	30000	93.29%	30000	87.18%
	40000	97.05%	40000	91.94%
m=3	10000	87.22%	10000	62.76%
	20000	93.97%	20000	85.18%
	30000	98,29%	30000	91.94%
	40000	99.56%	40000	94.36%
m=4	10000	83.20%	10000	69.76%
	20000	97.90%	20000	87.67%
	30000	99.80%	30000	92.71%
	40000	99.98%	40000	94.76%
m=10	10000	99.83%	10000	78.47%
	20000	99.97%	20000	89.10%
	30000	100.00%	30000	92.90%
	40000	100.00%	40000	94.87%
m=15	10000	99.92%	10000	78.60%
	20000	100.00%	20000	89.00%
	30000	100.00%	30000	92.90%
	40000	100.00%	40000	94.87%

The TMRSB approach is also simulated in the BlockSim and the result listed below in Table 5. The result shows TMRSB improve the reliability compare with the single standby system and the higher component reliability; the higher improvement can be achieved

Table 5 TMRSB Simulation Result

Simulation 50000hours	Imperfect Switch (90%)	
Standby # (m)	MTTR(hours)	System Reliability
m=2	10000	74.50%
	20000	96.71%
	30000	99.47%
	40000	99.80%
m=3	10000	86.30%
	20000	99.02%
	30000	99.68%
	40000	99.97%
m=4	10000	93.73%
	20000	99.45%
	30000	99.83%
	40000	99.99%

Based on the analysis and simulation, we can reach the conclusion that the TMRSB system can benefit the system reliability with lower storage overhead in the specific reconfiguration device. The reliability of standby system may not be linearly increased with the number of standby configurations. The reliability of the configurations both active and standby will be an essential factor on the reliability issue. The higher configuration reliability, the more reliability benefit is shown on system performance.

The following two chapters present how to utilize the TMRSB model to address the autonomous repair problem in EHW applications and OC systems. Chapter 4 introduces the CBE approach for EHW which normally do not have full self-repair capacity. Chapter 5 present the performance and measurement of the CBE approach using combinational logic circuits from the MCNC91 benchmark suite as an experimental sample. However, the inherent limitation of the hardware resource is going to be exhausted for a small circuit eventually and may not support the ultimate objective of space application which required sustainability a long mission. The OC architecture are presented in chapter 6 use multiple AE components with identical designs to self-regulate the system performance FE components which handle fault detection and repair. Therefore, there are no golden elements of the GA in the case of EHW.

CHAPTER 4: AUTONOMOUS REPAIR USING COMPETITIVE RUNTIME RECONFIGURATION

The proposed CBE scheme realizes regeneration by integrating all phases of fault handling within an evolutionary algorithm process flow. It employs population diversity information, partially online recovery of failed resources, and resource recycling with adaptable overheads. Two innovations are realized for self-adaptive EHW regeneration: elimination of additional test vectors and temporal assessment based on relative fitness assessment.

4.1. Detecting Faults using a Population of Alternatives

CBE detects and classifies faults using a *temporal voting* approach. In the Duplex mode, the outputs of two competing active L and R *half-configurations*, are compared to detect discrepancies. Alternative pairings are considered over time to provide the robust consensus described below. Each individual in the population is represented as a *configuration bitstream* [22] that defines the physical resources it uses and their interconnections when it is loaded onto the FPGA. An initial population of known-good individuals is created at design-time. These primordial configurations are functionally-identical, yet they utilize physically-distinct resources by having alternative design or place-and-route implementations. In the Duplex Mode, two of these competing half-configurations are instantiated on the reconfigurable FPGA device by downloading their configuration bit streams. This realizes a conventional *Concurrent Error Detection (CED)* [58] arrangement to detect at least any single resource fault with certainty. As in traditional CED approaches, comparison of the outputs of the two resident half-configurations

will produce either discrepant or matching outputs to indicate the presence or absence of faulty resources in the utilized FPGA hardware [50]. Maintaining exclusive resource utilization for half-configurations belonging to either half ensures that under a single fault assumption, the presence of a fault implies the fault-free nature of all the half-configurations designed for the other half. An additional advantage of using pre-designed configurations is that system downtime is reduced to a minimum as potentially viable alternatives are available. Also, the use of *L* and *R* half-configurations enables the use of runtime reconfiguration technology to reconfigure a portion of the device without taking other portions offline.

The CBE process is described below using Duplex Mode depicted in Figure 1. After the device is configured with the competing configurations, the same input vector is applied to both of the functionally-equivalent logic instances. Fault detection is accomplished when there is a disparity between the outputs of the active configurations, as ascertained by the discrepancy detector. The presence or absence of discrepancy is used to adjust the *Discrepancy Values* (*DV*s) of both individuals without rendering any judgment at that time as to which individual is actually faulty. Succeeding pairings of alternate combinations identify those individual(s) that utilize faulty physical resources through consensus formation. Meanwhile, the fault-free configurations become exonerated over time. This is because the *DV* of a faulty configuration always increases regardless of its pairing, yet the *DV* of fault-free half-configurations which are paired together are not increased. This *temporal* testing scheme enables the use of pseudo-exhaustive testing over a period of time without the reduced availability imposed by exhaustive testing.

4.2. CBE Approach

Competition among a diverse pool of individuals can generate robust information about their relative competence and reliability. In particular, the fitness states and health transitions of competing FPGA half-configurations during online operation are depicted in Figure 14: States in the Lifetime of the ith Half-Configuration. At any instant, each individual configuration is labeled with one of four states {*Pristine* (C_P), *Suspect* (C_S), *Under Repair* (C_U), *Refurbished* (C_R)} as governed by the transitions indicated by the numbered arcs in Figure 14: States in the Lifetime of the ith Half-Configuration. Initially, all of the individuals in the population begin in the *Pristine* state.

If output discrepancies are detected among the half-configurations in the FPGA then the competing *L* and *R* half-configurations undergo indicated health state transitions. A comparison can lead to one of two results, "$L=R$" or "$L \neq R$." When $L=R$ occurs, both individuals retain their *Pristine* state, as shown by transition event "*1*". However, when their outputs disagree, then transition "*2*" occurs whereby both of the configurations are demoted to the *Suspect* pool and their *DV* is increased. The determination of a configuration's fitness state for subsequent transitions is based on its cumulative *DV* relative to *DV* of the other individuals in the population evaluated over an *Evaluation Window*, denoted by *E*

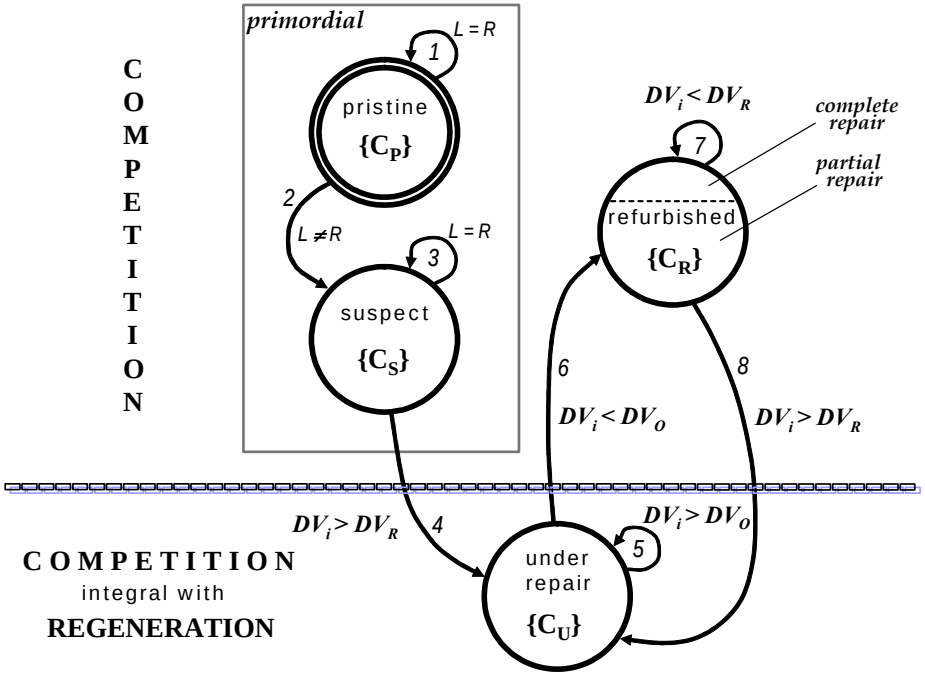

Figure 14: States in the Lifetime of the ith Half-Configuration

The period E defines a fixed number of evaluations at the end of which an individual's fitness state is updated depending on its observed discrepancy history. Only after an individual has undergone such testing is its fitness state updated. The *reintroduction rate,* denoted by λ_R, controls the rate at which individuals are rotated for instantiation on the FPGA. By varying λ_R, a tradeoff between the throughput and the rate of refurbishment can be obtained. In particular, the re-introduction rate denotes the probability that an instantiated functional configuration is replaced by another from the competing pool, regardless of whether it has completed its evaluation window, or exhibits a discrepancy. Higher throughput and availability can be ensured via a low reintroduction rate which will maintain individuals that perform well on the FPGA for the length of their evaluation window, at the cost of slower refurbishment of the individuals

undergoing refurbishment. Individuals that have been instantiated on the FPGA are replaced in one of three ways. They will be replaced when they articulate a discrepancy, when they have completed their evaluation window, or as dictated by the reintroduction rate.

The ith half-configuration is marked as *Under Repair* if its *DV* increases beyond the *repair threshold* denoted by DV_R as shown in transition *4* in Figure 14. DV_R is determined by the relative fitness of the operational elements among the population, i.e. those in the *Pristine*, *Suspect* and *Refurbished* states. After successive evolutionary refurbishment operations, if an *Under Repair* individual's *DV* returns to the range of the outlier threshold value DV_O as a consequence of transition *6*, then the configuration is *Refurbished*. Over a period of time, the *DV* of an individual could approach zero achieving complete regeneration. Without exhaustive testing however, it is not possible to completely distinguish partial regeneration from complete regeneration. Competing half-configurations remain *Refurbished* unless their *DV* rises above the *Repair threshold* DV_R, at which time they are again demoted to the *Under Repair* state. DV_O is lower than DV_R to ensure that only individuals with *DV* significantly lower than DV_R are recognized as refurbished enough to be operational.

4.3. Self-Adaptive Fitness Assessment using Outlier Identification

Instead of using an absolute fitness function with exhaustive testing, outlier identification is achieved using statistical techniques such as the hat matrix [59], H, where the diagonal elements Hii are used to identify the threshold to isolate faulty individuals as outliers. The hat matrix H defines the Least Squares projection matrix and is so named since it is denoted by a hat on the

column vector y=(y1,……,yn)t such that ŷ=H*y and ŷ is the LS prediction for y. The hat matrix H is defined as follows: consider that there are p explanatory variables and one response variable which will have n observations. The n-by-1 vector of responses is denoted by y=(y1,……,yn)t. The linear model states that y=X×θ+e, where θ is the vector of unknown parameters, e is the error vector and X is the n-by-p matrix:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_{11}} & \mathbf{x_{12}} & \cdots & \mathbf{x_{1p}} \\ \mathbf{x_{21}} & \mathbf{x_{22}} & \cdots & \mathbf{x_{2p}} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \mathbf{x_{n1}} & \mathbf{x_{n2}} & \cdots & \mathbf{x_{np}} \end{bmatrix}$$

Then, the ***H*** matrix is composed from ***X*** as follows: $\mathbf{H = X(X^tX)^{-1}X^t}$

The diagonal elements of H have a direct interpretation as the effect exerted by the ith observation on the expectation of response variable because they equal $\partial \hat{\mathbf{y}}_\mathbf{i} / \partial \mathbf{y}_\mathbf{i}$. The average value of the diagonal element Hii is *p/n* and it follows that $0 \le H_{ii} \le 1$ for all *i*. In the CBE approach, the DV of each individual can be viewed as one observation or one explanatory variable, and the observation interval can be set as the size of the entire population. Fortunately, since the X matrix consists of only one column in our application, we can see that the result of the XtX product is a single-element vector matrix, and its inverse can be computed using a straightforward computation. In general, the computation complexity of the H matrix approach is $2n^2+1$. In CBE, the threshold value is determined by an analysis of the diagonal elements *Hii* of the hat matrix generated from population statistics accumulated over an evaluation window.

In order to accelerate the identification of outliers, a Sliding Window, S, defines the period with which the global discrepancies consensus, to which all individual values are compared, is updated. Typically, S is selected to be an integer multiple of E such that S=q*E, where 1<q<|C| and |C| is the population size.

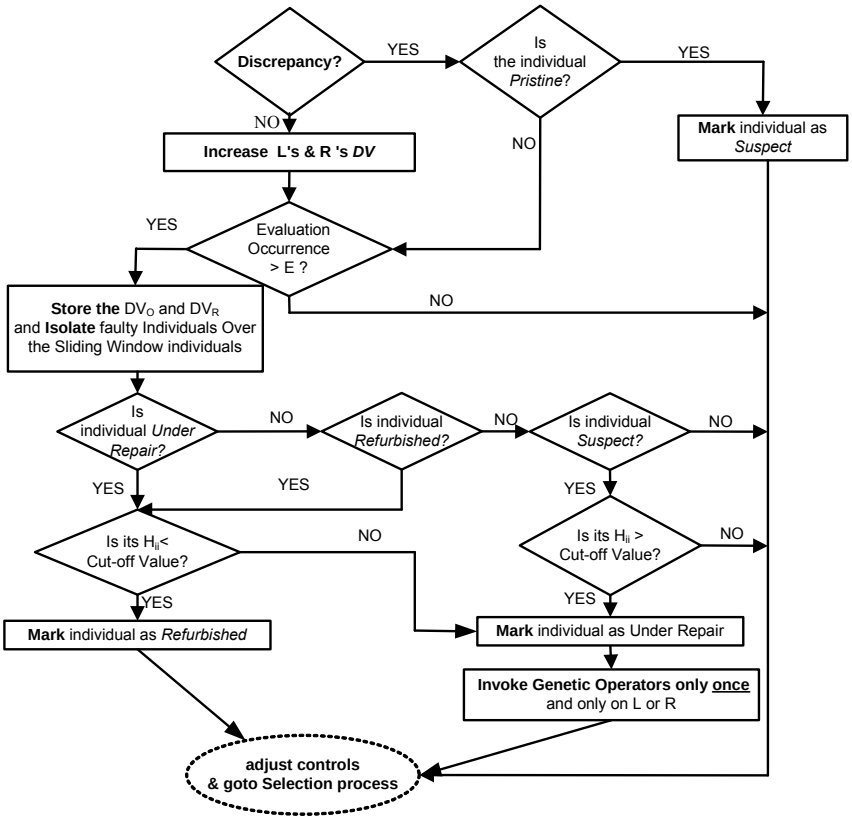

Figure 15: Fitness State Adjustment Process in the CBE Technique

Figure 15 depicts the *Fitness State Adjustment* process in CBE. Whenever a discrepancy is detected, the discrepancy values of the individuals involved are updated. The new discrepancy values are then compared to DV_R and DV_O to determine whether the individuals transition from one fitness state to another. Ideally, the repair and operational discrepancy values are updated

after a sliding window width of evaluations have been completed. Under ideal conditions, as soon as all the individuals in the population have completed at least E comparisons each, new values of these thresholds are obtained. Since it may be impractical to wait for all individuals to complete the requisite iterations, the sliding window width S reduces the latency involved in updating DV_R and DV_O by considering a subset of individuals instead of the entire population. The thresholds are updated as soon as a number of individuals, as defined by the sliding window width, have completed E iterations.

4.4. Achieving Device Refurbishment

Conventional GAs frequently use static fitness functions to search for pre-defined globally optimal criteria in analog [60] or digital [9] circuits. On the other hand, CBE uses a self-adaptive fitness measure that is based on consensus formation. This allows for adaptation throughout the process of solution construction involved with evolving a repair. If the realtime inputs are limited to a subset of the input space temporarily, then the relative fitness measure directs the GA towards creating individuals that perform best for this subset. However, there still remain other individuals in the population that perform optimally for other subsets. In the presence of viable alternative configurations, such *Recovery Complexity* of seeded search can be more tractable than *Design Complexity* using a blank slate..

Coarse-grained functional elements are recombined into candidate repairs using CBE's inter-module crossover operator. For crossover to occur such that offspring are guaranteed to utilize only mutually-exclusive physical resources within each L and R half configuration, a two-point crossover operation is carried out with another randomly selected *Pristine, Suspect* or

Refurbished individual belonging to the same *L* or *R* half, respectively. By enforcing speciation, breeding occurs exclusively in *L* or *R*, and non-interfering resource use is maintained. Crossover points are chosen along the boundaries of the FPGA's *Configuration Logic Blocks (CLBs)* so that intra-CLB crossover does not incur logic hazards. To encourage diversity and prevent stasis, an intra-modular input permutation operation performs alterations to logic cell functionality. The input permutation operator randomly changes the CLB's functionality or reconnects one of its inputs to a new randomly selected output. The *input permutation rate* defines the probability of changing the input connections and the logic functions of an LUT when the input permutation operator is applied.

CHAPTER 5: PERFORMANCE EVALUATION OF CBE APPROACH

The search-space complexity of a refurbishment problem is quantitatively compared to the complexity of the design problem using exhaustive analysis of the output space. Furthermore, refurbishment experiments were conducted using two classes of benchmark circuits. The first class consists of circuits where the fan-in exceeds fan-out and the second class includes two circuits where the converse applies. The performance of CBE in TMR and Duplex modes are analyzed for both kinds of circuits. In all experiments, performance is evaluated using two different schemes which are based on the bit-weight tabulation and the hamming-distance scoring of the observed outputs, respectively.

5.1. Circuit Representation and Benchmark Characteristics

The FPGA structure used in the following experiments is similar to that used by Miller and Thompson for GA-based arithmetic circuit design [9]. The feed-forward combinational logic circuit uses a rectangular array of nodes with four inputs and one output. Each node represents a Look-up Table (LUT) in the FGPA device, and a Configurable Logic Block (CLB) is composed of four LUTs. There are five dyadic operators OR, AND, XOR, NOR, NAND along with the unary operator NOT, from which a function may be composed within an LUT. The LUTs in the CLB array are indexed linearly from 1 to n. Array routing is defined by the internal connectivity and the inputs/outputs of the array. Internal connectivity is specified by the connections between the array cells. The inputs of the cells can only be the outputs of cells with lower row numbers.

Thus, the linear labeling and connection restrictions impose a feed-forward structure on the combinational circuit.

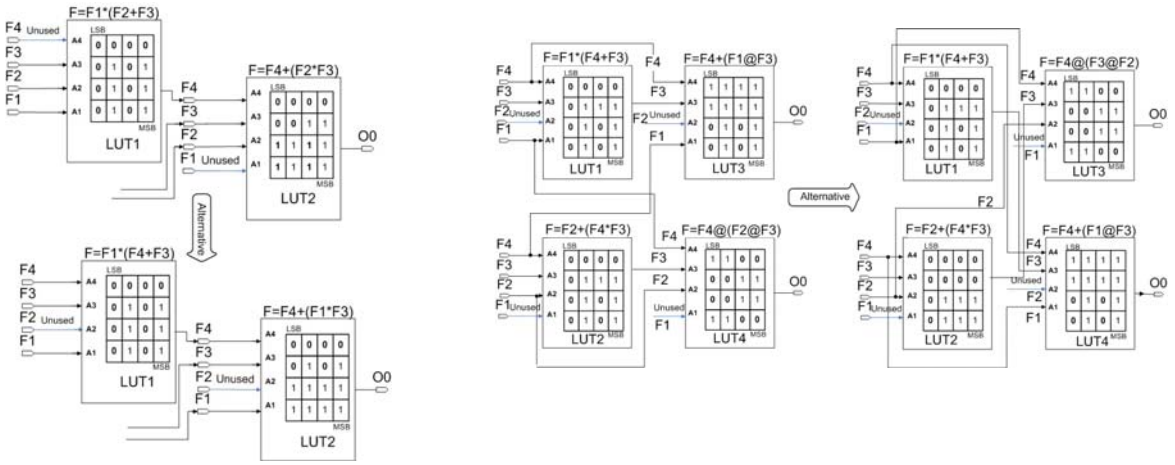

Figure 16: Generation of Alternate Configurations by –

a) Input Permutation (shown on left) and b) Cell Swapping (shown on right)

Each of the benchmark circuits was converted into a Verilog representation that preserved the described functionality. The design was then instantiated on the FPGA using Xilinx ISE version 9.1i. A diverse population of configurations was created from the single Xilinx tool synthesized design using *input permutation* and *cell swapping* operators. Figure 16 shows these operators, where F1 is the Least Significant Bit (LSB) of the input to an LUT and F4 is the Most Significant Bit (MSB). As shown in Figure 16a, input permutation leverages low-level redundancy by utilizing the unused inputs of LUTs to modify the input sequence of a single LUT as well as corresponding LUT functionality to maintain identical output behavior. The cell swapping operation, shown in Figure 16b, changes interconnection sequences among LUTs. The cell-swapping operation maintains the feed forward property and re-connects the LUTs to

preserve the functional logic. Together, these operations produce diverse circuits with different behavior under single or multiple physical resource failures. These circuit modification operators are also used later by the genetic algorithm to realize refurbished configurations during the repair process.

Benchmark circuits from the MCNC91 benchmark suite [61] were used to analyze CBE performance. Table 6 lists the characteristics of these circuits. As listed in Table 6, the *z4ml* and *cm85a* circuits have a fan-in greater than the fan-out, and the *cm138a* and *2x-decod* circuits have a fan-out greater than the fan-in value. To verify CBE performance on a circuit that utilizes more resources than the circuits provided by the MCNC91 suite, the *2x-decod* circuit was created by appending multiple copies of the *decod* benchmark circuit. The resulting *2x-decod* circuit utilizes approximately four times the LUTs used by the other circuits. The circuits were described using VHDL for synthesis on a Xilinx Virtex-II Pro VP7 FPGA to estimate the gate count and the number of LUTs used. The *input pin redundancy* is calculated as the ratio of the number of unused LUT input pins to the total number of LUT input pins. Table 6 also lists the percentage of aberrant outputs produced by each circuit under a single stuck-at fault for the entire output space, across all possible fault locations to indicate the demands of each refurbishment task. Results from experiments conducted on the MCNC91 circuits also provide insights into the relative merits of operating CBE in the Duplex and TMR modes, and the effect of the performance evaluation method used. To examine more demanding failure scenarios the following experiments consider multiple resource faults.

Table 6: Characteristics of Benchmark Circuits

Type of Circuit	Circuit	Functionality	No. of Inputs	No. of Outputs	Gate Count	LUT Count	Input Pin Redundancy (%)	Aberrant Outputs (%)
Fan-in > Fan-out	z4ml	2-bit adder	6	4	20	8	25	28.6
	cm85a	Logic	11	3	38	12	16.7	19.9
Fan-out > Fan-in	cm138a	Logic	6	8	17	10	22.5	6.6
	2x-decod	Decoder	10	32	44	40	25	3.7

5.2. Quantifying Search Space Complexity under Fault

In order to evaluate the effect of a single stuck-at fault at the inputs of a circuit, the *Correctness-Under-Fault (CUF)* search space characteristics for the various circuits are generated. The CUF characteristics for a circuit are obtained by inserting a single stuck-at fault at each of the inputs of the circuit, and then applying all possible input combinations to the instantiated circuit. The deviation of the observed output from the correct, expected output completely describes the response of the circuit to all possible stuck-at faults for its entire input space. Using this data, a three-dimensional representation of the refurbishment search space can be plotted as shown in Figure 17.

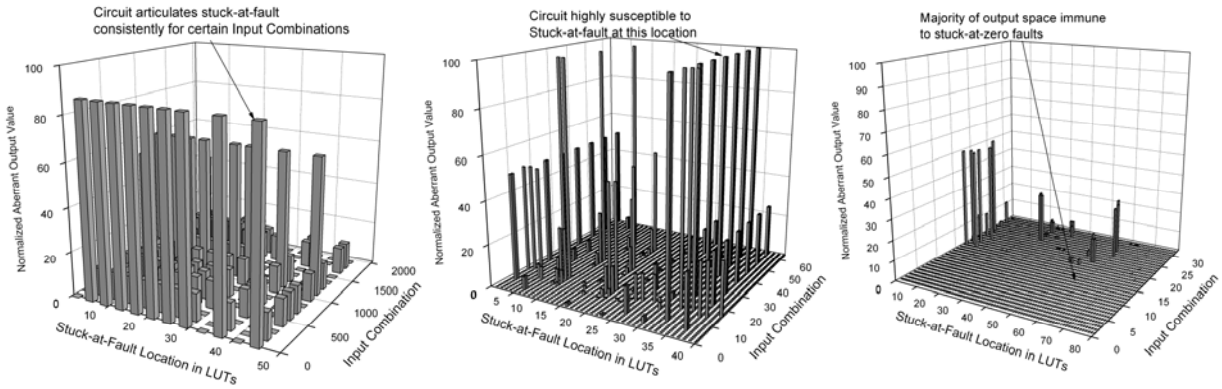

Figure 17: MCNC91 Benchmark Circuit Sensitivity to Stuck-at Faults

a) cm85a, b) cm138a and c) 2x-decod Circuit

The single stuck-at fault CUF search space of the benchmark circuits are shown in Figure 17, which show the Root Mean Squared discrepancy observed for all combinations of input and stuck-at-fault locations. Vertical bars depict representative aberrant outputs, with one sample taken from every 300 data points of the entire search space to enhance readability. In the above figures, the *x*-axis represents a particular stuck-at fault identified by the input pin at which the fault is introduced, and the *y*-axis represents the input combination applied to the circuit. The *z*=0 plane represents input combinations for which the output response of the circuit is ideal, in the presence of a stuck-at-fault. The percentage of aberrant outputs for the various circuits listed in Table 6 are obtained as the percentage of such points in the output space that are affected by the various stuck-at faults. The peaks and troughs in the 3-dimensional plot represent deviations from the expected output due to the presence of a fault. The search space may be sparse, as in Figure 19c, which represents the CUF space of the *2x-decod* circuit, or dense as in the case of the *cm85a* circuit shown in Figure 17a.

In the case of a refurbishment problem, the evolutionary algorithm is assisted a-priori by the presence of points in the search space where the deviation from the expected behavior is null, as represented by the set of points for which Normalized Aberrant Output is zero. For example, if a particular LUT input is unused, a stuck-at fault at this pin will not adversely affect the outputs of the circuit. This characteristic can be used by the cell-swapping and input permutation operator during the search for a refurbished configuration. In a design problem, the search for a solution starts from a population of arbitrary individuals which provide no such partial functionality. Yet, a refurbishment problem can leverage diversity of partially working spares.

5.3. Source of Redundancy in Digital Circuits

Under CBE, individuals are prioritized for refurbishment operations based on their discrepancies. In particular, individuals whose DV's deviate the most from the average DV of the population are given more opportunities to undergo refurbishment. This is implemented by reloading the individual under repair with a frequency exceeding that of individuals who have a higher relative fitness. Figure 18 shows the measured performance of an individual over 28 iterations during the repair process for the *z4ml* circuit. In this particular experiment, the reintroduction rate used was 20%, with both the cell-swapping rate and the input permutation rate set to 20%. As shown in Figure 18a, whenever the discrepancy of the individual rises above the average discrepancy of the population, the individual is reloaded onto the FPGA, as evidenced by Figure 18b. This can be clearly seen for the first and the next to last iterations shown in Figure 18a and Figure 18b. Conversely, when the individual discrepancy is equal to, or less than the average discrepancy of the population, the individual is not reloaded, or reloaded less than the average member of the

population. This ensures steady improvement in the average fitness of the population, while ensuring that individuals are prioritized for refurbishment operations based on their relative fitness arrived at by using a consensus-based evaluation method.

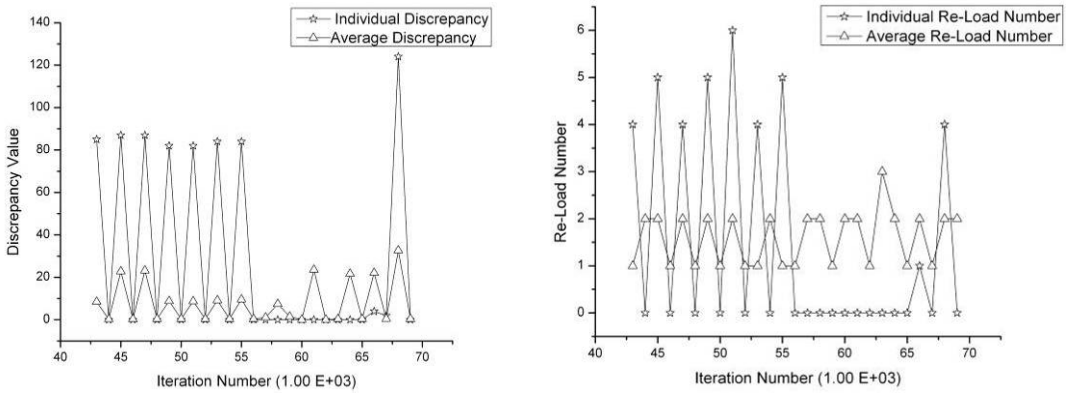

Figure 18: Prioritizing Individuals for Refurbishment
a) Discrepancy Values, and b) Number of Iterations the Individual is Reloaded

5.4. Initial Circuit Population Design

Figure 19 and Figure 20 show the performance of CBE under the Duplex and TMR modes when using bit-weights to calculate the fitness of individuals. Figure 19 shows the results of refurbishing circuits in a population of 20 individuals in the Duplex mode, with ten individuals each comprising the Left- and Right-half configuration populations. The Duplex experiment begins when a fault is inserted into two resources, one on the Left-half and one on the Right-half, which impact 18 of the 0.20 individuals in the population. In the TMR mode, a population of 30 individuals is used, with three resource faults distributed across each voting component affecting 27 out of the 30 individuals. However, as opposed to the Duplex mode, in the TMR mode,

outputs from three individuals are compared for the input vector applied to realize throughput, and the majority outcome is asserted as the output of the system.

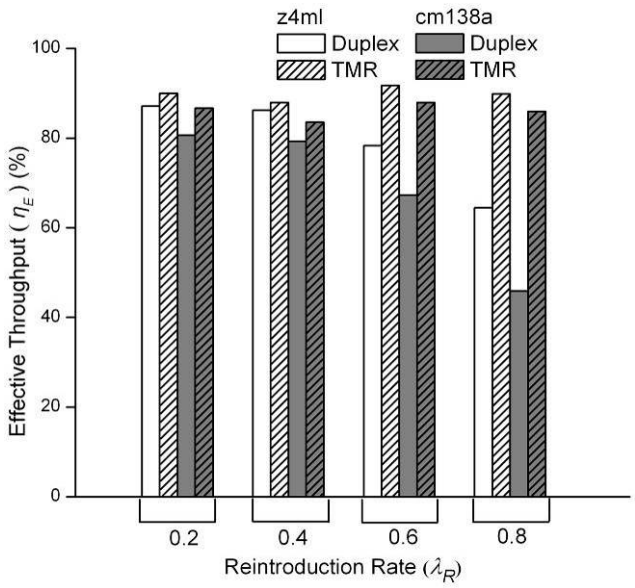

Figure 19: Effective Throughput η_E during Regeneration Under Duplex and TMR Modes of Operation

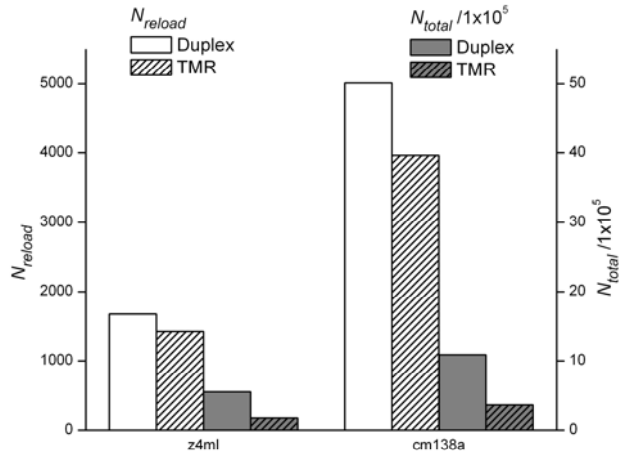

Figure 20: Comparison of Performance Characteristics under Duplex and TMR Modes

In all these experiments, the cell-swap rate and the input permutation operation rate were maintained at 80%. In Figure 20, performance metrics from the experiment refurbishing the population with re-introduction rate $\lambda_R = 0.4$ are presented, in order to compare the overheads of the two modes. Detailed results obtained from the implementation of the two modes are listed in Table11 which tabulates several parameters listed in Equation 5.1. The *effective throughput,* η_E is measured using the following relationship:

$$\eta_E = \frac{N_{total} - N_{evolution} - N_{reload} - N_{incorrect} - (N_{reload} \times \beta_{reload})}{N_{total}} \qquad (5.1)$$

where, $\boldsymbol{N_{total}}$ is the total number of iterations required to refurbish the population,

$\boldsymbol{N_{evolution}}$ is the number of iterations in which the genetic recovery operators are invoked,

$\boldsymbol{N_{reload}}$ is the number of iterations where the individuals currently evaluated are replaced by other members from the population,

$\boldsymbol{N_{incorrect}}$ is the number of iterations yielding discrepant outputs verified during the experiment to be incorrect,

$\boldsymbol{\beta_{reload}}$ is the *reload penalty*, which is the ratio of the time taken to reload a configurations and the time taken to compute the outputs for a single input.

Thus, η_E measures effective throughput during refurbishment by accounting for the number of iterations, and the time spent in refurbishment-related operations.

As shown in Figure 19, for low values of λ_R, $0.2 \leq \lambda_R \leq 0.4$, the effective throughput of CBE in the Duplex mode is only 2% to 6% lower than TMR mode. For example, with the *z4ml* benchmark circuit, from Table 7, CBE in TMR mode provides 2.9% higher effective throughput when compared to the Duplex mode. The difference in effective throughput is greater across different values of λ_R for the *cm138a* circuit. Performance varies depending on the fan-in / fan-out ratio of circuit as shown by the *z4ml* circuit, where fan-in > fan-out, and the *cm138a* circuit where fan-in is less than fan-out.

Table 7: CBE Performance under Duplex and TMR Modes for Two Different Circuits

Circuit	***Mode***	λ_R	$N_{evolution}$	$N_{incorrect}$	N_{reload}	N_{total}	η_E	***Fully Refurbished Individuals***
Z4ml	Duplex	0.2	144	3.9×10^4	1594	4.4×10^5	87.1	5
		0.4	166	5.7×10^4	1674	5.4×10^5	86.2	11
		0.6	133	5.3×10^4	1671	3.3×10^5	78.3	13
		0.8	131	5.7×10^4	1907	2.2×10^5	64.5	12
	TMR	0.2	132	3.9×10^3	1554	2.1×10^5	90.0	5
		0.4	150	5.9×10^3	1422	1.8×10^5	87.9	12
		0.6	125	1.5×10^3	1002	1.5×10^5	91.7	13
		0.8	121	2.3×10^3	1237	1.6×10^5	89.9	13
Cm138a	Duplex	0.2	187	1.1×10^5	4771	8.7×10^5	80.6	4
		0.4	231	1.7×10^5	5011	1.1×10^6	79.3	11
		0.6	165	1.6×10^5	5002	6.5×10^5	67.3	12
		0.8	161	1.7×10^5	5710	4.3×10^5	45.9	12
	TMR	0.2	1362	1.2×10^4	4229	4.3×10^5	86.6	5
		0.4	1398	1.8×10^4	3965	3.7×10^5	83.6	11
		0.6	1348	4.6×10^3	3125	3.2×10^5	88.0	13
		0.8	1340	6.8×10^3	3595	3.2×10^5	86.0	14

However for $\lambda_R \geq 0.6$, the difference in the effective throughput becomes pronounced in favor of the TMR mode. This occurs because a higher re-introduction rate replaces active configurations with configurations from the under repair pool more frequently. TMR throughput is less

adversely affected because it ensures throughput whenever any two of three configurations' outputs agree, giving $\binom{3}{2} = 3$ ways for agreement, as opposed to the Duplex mode where there is only one combination to realize agreement. In both Duplex and TMR modes, disagreements trigger reloading of configurations as well as re-computation of the outputs.

Figure 20 quantifies the time vs. space tradeoff during recovery when utilizing 50% fewer physical resources in Duplex mode as opposed to TMR. It shows the number of reloads and the total number of iterations required to refurbish the population for the *z4ml* and *cm138a* circuits when $\lambda_R = 0.4$. Under Duplex mode, up to 1.6 times as many reloads and 1.3 to 3 fold total iterations are required to achieve refurbishment of the population. This correlates with the lower effective throughput observed under the Duplex mode. From Table III, with higher values of λ_R, such as $\lambda_R = 0.8$, the increased number of reloads required for Duplex mode skews throughput in favor of TMR mode.

5.5. Effect of Reintroduction Rate on Refurbishment Performance

Table 8 lists the number of individuals that were fully refurbished from adverse effects of a single fault inserted into 18 out of 20 individuals under CBE in Duplex mode. A Refurbished individual might be partially or fully refurbished. An individual is fully refurbished if and only if its output response to the entire set of possible input vectors implements the correct truth-table in its entirety. The fitness of the individuals was evaluated using a bit-weight scoring scheme. The stopping criterion for all refurbishment experiments was the condition wherein none of the individuals remain in the Under-Repair pool. Nonetheless, the effectiveness of the refurbishment

can also be measured by exhaustively testing each individual under all possible input combinations. Such exhaustive testing is not required for CBE to refurbish individuals; it was conducted only to evaluate performance at the end of a refurbishment cycle.

Table 8: Number of Fully Refurbished Individuals vs. Effect of Reintroduction Rate (λ_R) for Four Circuits

Reintroduction rate (λ_R)	**Circuit**	**Fully Refurbished Individuals**
20	z4ml	8
	cm85a	6
	cm138a	5
	2x-decod	12
40	z4ml	11
	cm85a	12
	cm138a	12
	2x-decod	14

Table 12 indicates that as λ_R increases from 0.2 to 0.4, the number of individuals that are fully refurbished in the population rises, irrespective of the circuit used. The improvement depends on not just the fan-in to fan-out ratio, but also on the particular circuit. The *cm138a* circuit shows the best improvement – from three recovered individuals with the lower re-introduction rate to 10 fully refurbished individuals. In the *2x-decod* circuit, which is also a circuit with a fan-in greater than the fan-out, there is an improvement of only two additional fully refurbished individuals.

A higher reintroduction rate increases the probability that more individuals are evaluated, evolved, and therefore improved. This improvement occurs at the cost of the greater number of

re-computations and re-loads necessitated by individuals under repair which are instantiated on the FPGA for evaluation, leading to an increased number of discrepancies. If any individual in the population expresses very low fitness as expressed by a higher discrepancy count, the individual will be demoted to the Under Repair pool to be improved. This refurbishes individuals with low fitness, leading to a higher number of fully recovered individuals.

An additional insight provided by these results is that even though all individuals are not fully recovered, after successive evaluation, the individuals in the population were promoted from the Under Repair pool to the Refurbished pool by virtue of their fitness to inputs observed in practicality. In this manner, CBE emphasizes sustainability by improving the robustness of the entire population in the process of achieving complete recovery.

5.6. Comparing Discrepancy Scoring Schemes

Figure 21 and Figure 22 show the relative performance of two different discrepancy scoring schemes. In the Hamming distance method, the fitness of individual configurations was measured using the Hamming distance of the outputs produced by the competing individuals. The bit-weight scheme measures the arithmetic difference between outputs produced by the individuals. Experiments were conducted under the Duplex mode for the *cm85a* circuit and the *2x-decod* circuit. Results from the experiments, both of which were conducted with CBE in the Duplex Mode, are listed in Table 9.

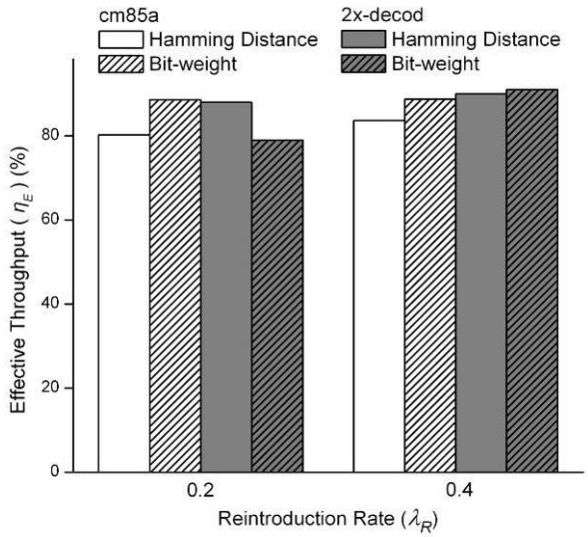

Figure 21: Effective Throughput with Hamming Distance and Bit-weight Schemes

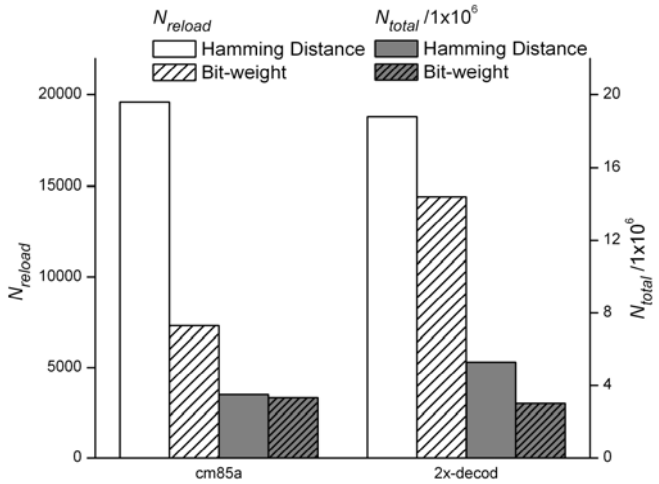

Figure 22: CBE Performance Characteristics with Hamming Distance and Bit-weight Schemes

As shown in Figure 21, the bit-weight evaluation scheme leads to higher effective throughput for the *cm85a* circuit for both values of λ_R, while for the *2x-decod* circuit, the hamming-distance based evaluation scheme seems to lead to a higher throughput. This is due to the fact that unlike the *cm85a* circuit, the fan-out of the *2x-decod* circuit is greater than the fan-in. Thus, a fault nearer the inputs of the circuit will affect a larger number of outputs for the *2x-*

decod circuit. Under these circumstances, the Hamming distance of the output from the ideal output will provide a much better indicator of the fitness of an individual configuration. From Figure 22, it can be seen that the Hamming-distance scheme reports a greater discrepancy value resulting in more refurbishment operations than the bit-weight scheme. As listed in Table V for either performance evaluation scheme, the effective throughput as well as the number of individuals that are fully refurbished for a constant λ_R do not vary significantly. From the results in Table 9, it is clear that refurbishment can benefit from the selection of an appropriate fitness-evaluation scheme for the target circuit.

Table 9: CBE Performance under Hamming Distance and Bit-weight Performance Evaluation Schemes

Circuit	***Performance Evaluation Scheme***	λ_R	$N_{evolution}$	$N_{incorrect}$	N_{reload}	N_{total}	η_E	***Fully Refurbished Individuals***
cm85a	Hamming Distance	0.2	1987	2.8×10^5	70387	8.0×10^6	87.5	5
		0.4	2120	3.6×10^5	19593	3.5×10^6	83.6	10
	Bit-weight	0.2	1913	3.3×10^5	7270	4.1×10^6	87.9	4
		0.4	1684	2.4×10^5	7300	3.3×10^6	88.7	10
2x-decod	Hamming Distance	0.2	13100	4.4×10^5	16676	5.1×10^6	88.0	11
		0.4	14420	3.2×10^5	18821	5.3×10^6	90.0	13
	Bit-weight	0.2	10115	5.9×10^5	13362	3.7×10^6	79.0	10
		0.4	12750	1.2×10^5	14429	3.0×10^6	91.0	12

5.7. Recovery from Pervasive Faults

The impact of simultaneous resource failures may completely deplete all viable spares from the dormant population. The worst case scenario occurs when all individuals in the N mutually exclusive resource pools allocated to each module are affected, creating a pervasive hardware failure. However, the residual functionality of each individual can be utilized by the CBE approach to fully refurbish one or more individuals. The CUF search space characteristics of the circuits demonstrate the viability of refurbishing individuals using the genetic operators. When affected by pervasive faults, the functionality of each of the diverse individuals remains partially intact. The less affected individuals will then be favored by CBE to remain on board longer and used to generate the consensus output. Conversely, the worst affected individuals will, by virtue of their discrepancy with the majority vote, be forced to undergo evolutionary repair to improve their performance.

The diverse failure behavior under a pervasive fault can be exploited to generate a completely functional individual even if all individuals in the population are faulty. Experiments conducted on the *2x-decod* circuit, which is the most resource-intensive of the benchmark circuits yield completely refurbished individuals. The Hamming distance based fitness metric produces a majority-indicative vote when the outputs of the three modules are compared on a bit-by-bit basis. In these experiments, all of the 30 individuals across the three modules are negatively affected by a single fault in the resources used by each of the TMR modules. In a sample experiment, CBE realizes three completely refurbished individuals after $N_{total} = 6 \times 10^5$ iterations with a reintroduction rate $\lambda_R = 0.4$. To realize refurbishment, the configurations were reloaded

N_{reload} = 1121 times, and a total of $N_{evolution}$ = 552 evolutionary operations were completed by CBE.

In all the experiments, the majority voted output produced by the three modules was asserted as the output. The throughput was observed to be maintained at 95% throughout the refurbishment experiment. High throughput is maintained during refurbishment because even partially-fit individuals can arrive at the correct result for many subsets of inputs encountered at runtime. For measuring throughput and evaluating the absolute fitness of the individuals, the outputs were verified against the truth table of the circuit. However, the correctness information provided by these comparisons was not made available to the refurbishment process. Of course, successful resolution of a pervasive fault still relies on having a population large enough and diverse enough to make recovery tractable by consensus.

CHAPTER 6: FAULT MONITORING AND RECOVERING USING ORGANIC COMPUTING APPROACH

6.1. Embedded Organic Computing Architecture

New trends in architecture and investigations for run-time adaptive systems have begun to explore the possibility of autonomous run-time reconfiguration for increased reliability and power awareness [35]. The Organic Embedded System (OES) architecture developed herein utilizes Evolvable Hardware [62] approaches based on a variety of genetic techniques.

6.1.1. Requirements and Architectural Overview

Requirements are summarized below for the ASoC-style architecture in Figure 23 which is partitioned into two logical layers. The functional layer houses the Intellectual Property (IP) core component or Functional Elements (FEs). FEs can be any functional element from general purpose CPUs, memories, on-chip busses, special purpose processing units or network interfaces. The Autonomic layer consists of Autonomic Elements (AEs) and an interconnect structure among the AEs. The following properties are inherent:

1. FEs and AEs both reside on *two distinct layers* with an interconnection structure between them.

2. The AEs and FEs can either be realized in *hardware*, *software*, or through hardware/software *co-design*,

3. The AE layer should supervise the functionality of the FE elements in the FE layer while requiring *no application-specific algorithms on the AE layer* to be developed to realize this fault-tolerant functionality.

4. The Observer/Controller architecture includes an AS element which had no counterpart to evaluate if the AS fault-free, so in the OES design we *address reducing the vulnerability of the AS* by emphasizing its simplicity as part of our approach.

As shown in Fig 29, the separate layers of the OC architecture implemented in the OES are mapped to alternating vertical columns of logic slices on the Xilinx Virtex II Pro FPGA device. This column-oriented structure permits the architecture to take advantage of Xilinx partial reconfiguration technology to manipulate the bitstreams of either the AEs or FEs configurations for the fault recovery.

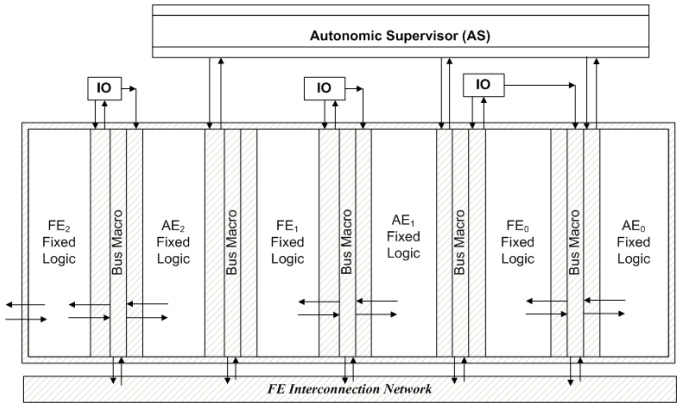

Figure 23: Column-oriented OES on Xilinx Virtex II Pro FPGA platform

Even a small size system composed of large numbers of various functionalities will need to occupy differing amounts of physical resources for each FE as well as require a different number

of I/O resources. Thus as shown in Figure 23, each FE is placed in single or multiple contiguous columns of the FPGA chip. The number of columns for each FE can be allocated as necessary according to the area requirements of the system being designed. Xilinx *bus macros* [21] are used to provide relocatable reconfigurable interfaces between FEs and AEs, AEs and the AS, and between FEs via a user-defined interconnection network module.

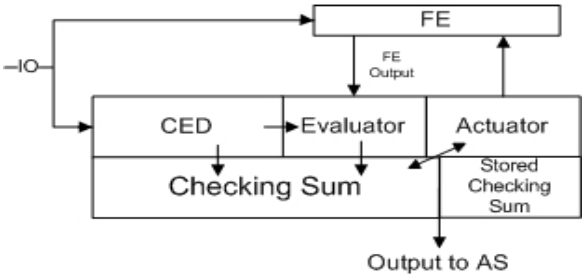

Figure 24: AE architecture in OES

Furthermore, controllability and maintainability demands can become substantial because of the overhead associated with scheduling, coordinating, and communication among the large number of interacting components. In order to evenly distribute this burden, the decentralization of the Observer/Controller components is proposed. In OES, the AEs reduce the demand for centralized controllability as shown in Figure 24. It consists of a *Concurrent Error Detection (CED)* [52] unit to collect and *Evaluate* outputs from 2 FEs, a *Checksum* for AE fault detection which are checked against *Stored Checksum* values and an *Actuator*. Each AE will monitor the operation of the corresponding FE component, evaluate the performance of the FE and render a local assessment on the failure status of FE. An important architectural property of the OES is that all AE components are identical in structure despite the fact that they monitor different types

of FEs. The homogeneous characteristics of the AE components deliver a uniform-behavior property which is leveraged to realize a consensus-based evaluation fault-detection methodology.

The AE layer will constrain the fault impact under consensus-based control mechanisms in a fashion that can improve system autonomy level while not needing application-specific information about the FEs nor extensive details of their functional behavior. Even though the AE components will add an additional layer to the design, this will ease modification difficulties inherent with current commercial IP cores while reducing the failure impact as results show in Section 4.3.

In addition to the AE and FE layers, the OES architecture also contains an AS. The AS implements the consensus mechanism to evaluate the behavior of all the AEs in the system and distinguish the abnormal individuals whose behavior may be distinguished from the rest of the members in the AE population. GA operators are implemented here to achieve fault recovery. All other factors being equal, the likelihood of local permanent fault of any component is proportional to the device area required for its realization. The AS is kept as simple as possible to reduce its complexity and reduce its likelihood of experiencing a fault proportionally.

6.1.2. System Operation

The OES architecture supports several operational phases of interaction between the FEs, AEs, and AS. The initial state of all components is fault-free. Figure 25 shows a diagram of the flow of operations in the OES architecture as described below.

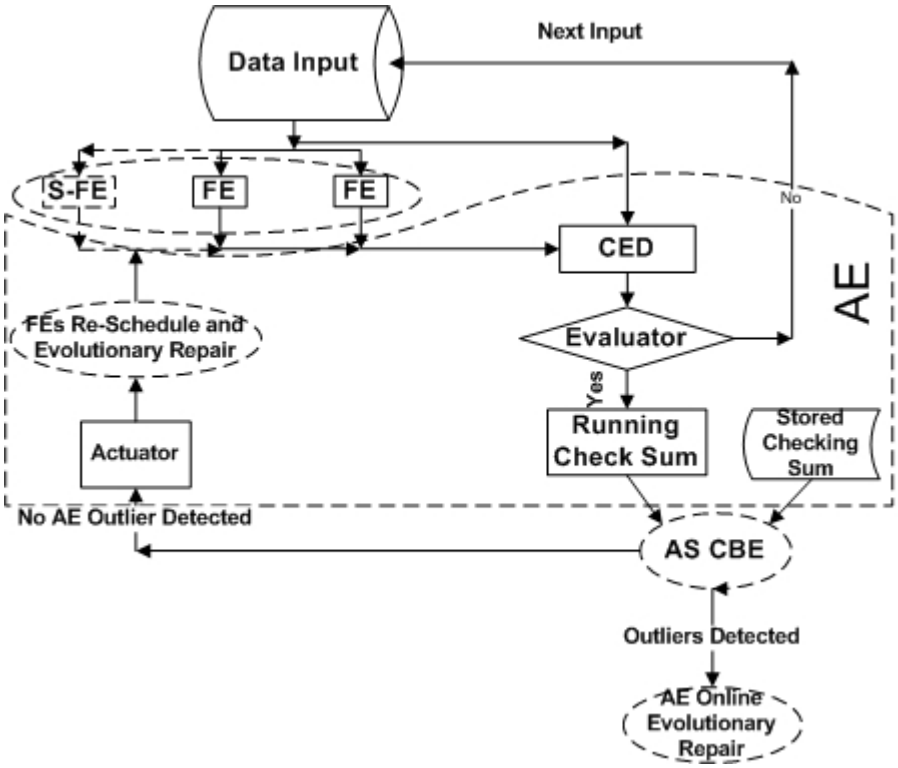

Figure 25: OES Integrated FE and AE Failure Detection Procedure

6.1.2.1.System Initialization Phase

FE Initialization step

Three functionally identical FE configurations labeled *FE*, *FE*, and *S-FE* are instantiated on different physical locations. Initially, only the two FEs are active and the S-FE acts a cold spare FE. The FEs supply the output for each set of inputs applied in parallel in a Concurrent Error Detection configuration to the AE for the fault detection.

Compute Checksum step

Each AE contains a Checksum Component which uses the stored outputs of the AE along with the small finite number of possible input combinations to the Evaluator and Actuator to populate the Check Sum Lookup Table (CS-LUT) in the AE. This feature in the AE will be utilized to detect if the current AE is faulty in a consensus-based approach. For the benchmark circuit selected a carry and sum, the CS-LUT required a 16-entry x 4-bit memory.

6.1.2.2.FE Fault Detection/Recovery and AE monitoring Phase

As depicted in Fig 31, at runtime the inputs destined to the FE are applied to both active ones under a CED strategy. After allowing for FE inputs propagation time through the AE, the expected output will be supplied to AE-CED for the fault detection. The output of the FE is then compared in the AE-CED module and any discrepancy between the two values will indicate that a fault has occurred either of one the FE or the AE-CED itself. Further detection will be required to distinguish which of the two is faulty.

If the AE component is identified as innocent then the fault which occurred in this output will be discarded and control will branch to a fault identification phase which will wakeup the cold standby FE and construct a temporary TMR system which can articulate the faulty FE under the new supplied external input. Furthermore, as described in Section 6.2, the actuator will initiate a repair cycle which may require automatic evolutionary repair of the identified faulty FE which will be set as standby-under-repair and the AE-CED will return to receive the remaining two active FEs' inputs. The decision-making procedure causes at least one throughput-delay penalty.

The AE supports two exclusive modes: FE monitor mode as described above and AE self-repair mode described in Section 6.2.2. Whenever the AS identifies that an AE is faulty then the AE will relinquish observation of its FE and focus on its own self-repair. Under FE monitor state, AE will keep observing the FE behavior and issue control instructions through the actuator.

The recovery procedure entails the use of alternative designs for the AE that have identical functionality but distinct physical resources. GA operation will manipulate the representation of the AE bitstream and evaluate each new generated offspring until the fault is occluded. This evolution may be time-consuming and halt the faulty FE operation, yet it is entirely automatic repair without any human intervention.

6.1.2.3.AE Fault Detection Phase

Three possible faulty scenarios may occur inside the AE:

- A fault may exist in the CED, Actuator, or Evaluator,
- A fault may exist in Check Sum component, or
- A fault may exist in the Stored CS-LUT.

All three scenarios are detected under the proposed approach. To detect if the CED, actuator, or evaluator are faulty we apply the outputs of the three components to the checksum circuit while simultaneously the inputs of the three components are applied to a parallel search circuit that will locate the input combination and its corresponding output in the CS-LUT. By the time the inputs propagate through the checksum circuit, the output from CS-LUT will be available, the two values are then compared and any discrepancy will detect a fault. The second and third scenarios

will also generate a discrepancy between the Checksum component and Stored Checksum component.

Furthermore, the dissertation reveals that the design would operate even under multiple faults as long as multiple faults generate the same faulty behaviors among different sub-components of the AE which is impossible in this design because each sub-component is implemented with distinct logic/arithmetic functionality. Nonetheless, we have observed in experiments that GA mutation operator described in Section 4.3 applied to AE unit and using cell swap can sometimes self-heal the AE unit even if more than one of its components is faulty.

6.1.3. CBE evaluation process and AE fault recovery Phase on the AS

A *Consensus-Based Evaluation (CBE)* approach is utilized for assessing the performance of individuals based on broad consensus of the AE population instead of a conventional fitness function defined for GAs. Adoption of CBE enables information contained in the population to not only enrich the evolutionary process, but also support fault detection and isolation. The AS component will collect all of AEs outputs and distinguish the abnormal individuals from the population instead of using traditional threshold, the population information will assist the outlier identification as well as fault recovery.

The automatic fault recovery utilizes the homogeneous characteristic of the AE components; each fault impact on any AE can mirror the health of the AE configuration which may reveal some inherent fault immunity property. Even though each AE occupies different physical

locations, they are implemented using identical logic functionality which can be used to overcome physical failure as explained in Sections 6.2.

6.2. Evolutionary Process FE and AE

The evolutionary process generates improved bitstreams which can be used to configure the logic fabric within a pre-defined genotype to phenotype mapping [63]. The phenotype is defined as the FE or AE circuit manifestation of a particular genotype. The physical realization is based on the specific configuration bitstream which is generated by the Xilinx synthesis tools and is readable by the FPGAs in that device family. In order to reflect the identical logic functionality, the logical chromosome of the AE will be uniform despite the physical configuration.

6.2.1. Genotype Definition

Genotype changes during evolution must adhere to the Xilinx-defined format of the bitstream. Even though not all bitstream information can be manipulated, there is still adequate evolutionary potential in the key fields of the bitstream. To prevent undesirable conditions that may damage the FPGA such as a mutation which might tie together two logic outputs inadvertently, a logical genotype is used for evolution. The proposed logical genotype in this chapter is an LUT vector which contains logic and physical ordering information plus the configuration I/O information as shown in Figure 26. The LUT is the basic building block of the genotype and contains both logic ordering numbers (Logic #) and physical ordering numbers (Col # and Row #) which identify both physical location and the functionality sequence of the LUT. Each LUT has 4 single-bit input lines in Xilinx FPGA architectures and each input line

contains the 2-tuple (Col # and Row #). The functionality of the LUT describes the logic function which is implemented and the content of the LUT stores the 24=16 bits which are the actual content of the LUT in the hardware.

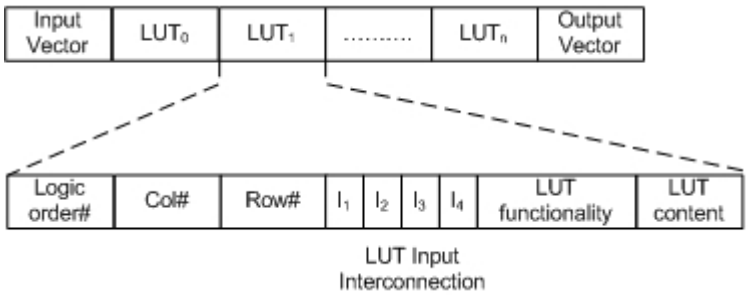

Figure 26: Genotype Chromosomes of GA Operation

Based on the genotype, three genetic operators are developed in this dissertation for manipulation, each of which emphasize a different aspect of information for the configuration and fault recovery process. The operators are implemented in the software simulator and in the FPGA prototype as described in Section 6.3.

The basic principle of evolutionary recovery approach advocated is on maintaining the integrity of the functionality of the configurations throughout evolutionary process. Instead of exploring completely random search space, the proposed approach will move outwards from the original design space by trying permutations of the existing logic and interconnection for occluding the physical failure. The reason is that feasible repairs may be expected to require less computational complexity than realizing a completely new design. Simulation and experimental results have borne this out this relationship between repair and design complexity [42].

6.2.2. Genetic Operations

6.2.2.1.Mutation Operation

The mutation operator is modified in order to fit the FPGA architecture which varied with traditionally defined mutation. Instead of the inverse binary bit approach, the objects of mutated are input interconnection of LUTs. The mutation will rearrange the input interconnection to each input pin of LUTs in order to search the potential unused resources for occulted the fault impact resource. In this way, the functionalities of LUTs are undistorted and explored in the search space.

Figure 27 and Figure 28 show how mutation works on both genotype and phenotype in the proposed GA design. Both figures show that after the permutation of input pins of the LUT, the new interconnection may use some inherent redundancy resource existing in the original design which is the result of the logic synthesis. The mutation also modifies the content of the LUT because of input changing. As shown in Figure 27, the original functionality is F = F1 • (F3＋F4) and input F2 is unassigned by the synthesis tool. The mutation operator will change the input arrangement to F4 as unused input and the function changed to F = F1 • (F3＋F2) and the shadow on the Before F2 and After F4 stand for the rearrangement of input lines as well as the LUT content update according to changed functionality. From Figure 28, you can see the detail update in both input lines and content of LUT according to the shadow show on each component. This operator will provide some opportunity for fault correction strategy for either input stuck-at fault or LUT content stuck-at fault. The process can be implemented without human interference and indispensable for the evolutionary procedure.

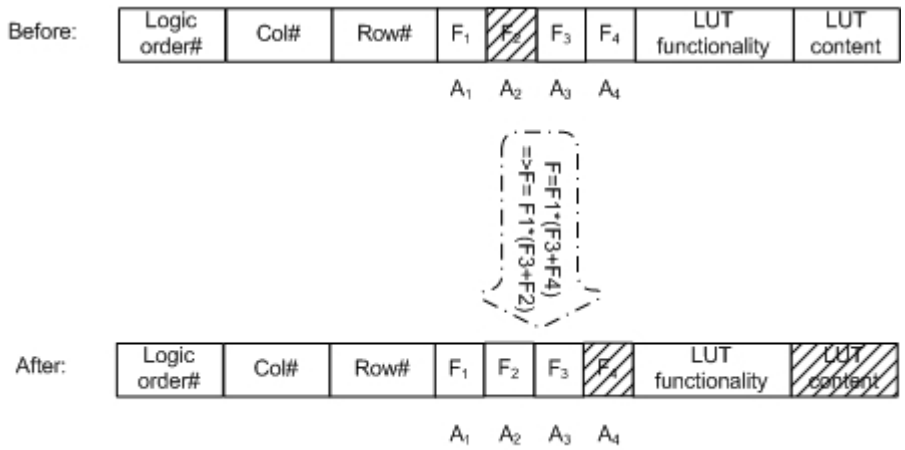

Figure 27: Mutation on the Genotype Chromosomes

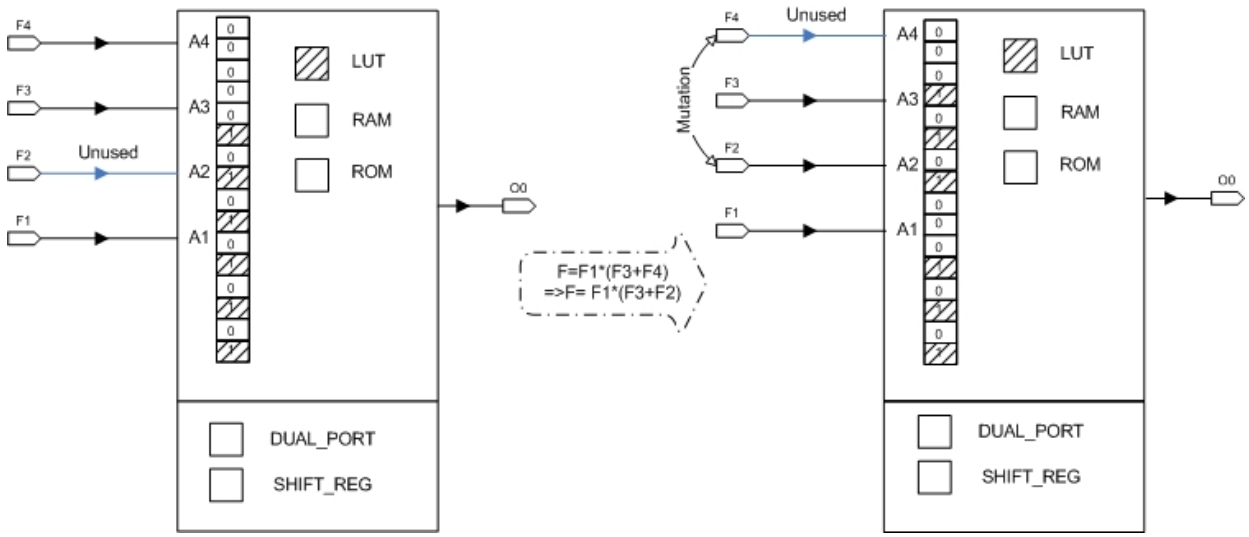

Figure 28: Mutation on the Phenotype

6.2.2.2.Cell-Swap Operation

The Cell-Swap operator is swaps two distinct LUTs' blocks and meanwhile maintaining correct the logic order and functionalities in the genotype. The swap will exchange all the LUT input

interconnections, LUT content and physical 2-tuple (Col#, Row#) as well as the logic sequence. As shown in Figure 29 and Figure 30, two LUTs swap all the information except the LUT sequence information which is fixed correspondent to hardware location. After swapping, the two LUTs will implemented the different functionality and have different input lines as the shadow in the figures. In this way, some fully occupied LUT may swap to some partially occupied LUT and find some alternative physical resource to recover from the fault impact. Another update issue in the configuration which should be considered but not shown in the figures is the output line update according to the swapping. Since the logic sequence now located in different LUT, the interconnection of output vector should also get current 2-tuple (Col #, Row #) to keep the integrated functionality of the entire configuration.

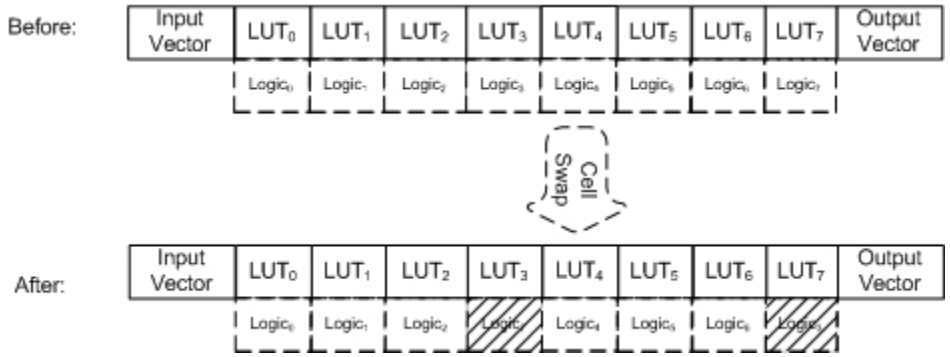

Figure 29: Cell-Swap operation on Genotype chromosomes

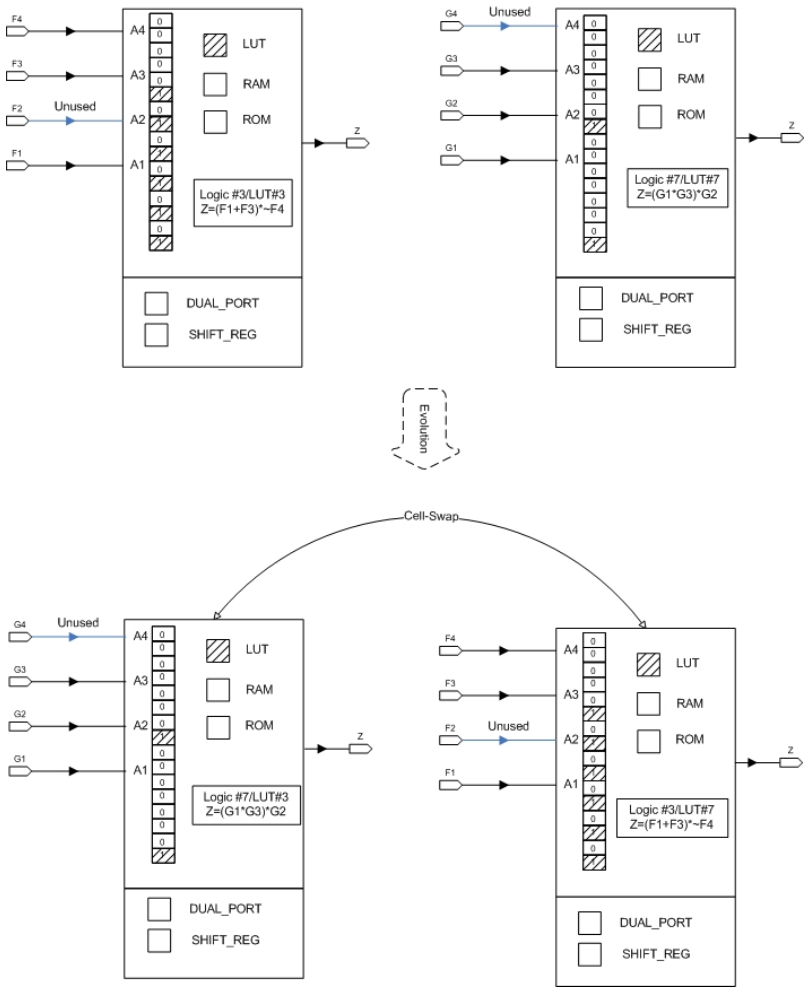

Figure 30: Cell-Swap operation on Phenotype chromosomes

6.2.2.3.Partial Match Crossover Operation

Partial Match Crossover (PMX) is proposed by [5] and maintains the crossover information as well as order information. In our design, the logic orders of each LUT are fixed and thus limit the possible search space of the initial design. Under PMX, two configurations are aligned, and a crossover site is picked uniformly at random along the boundary of the LUTs in genotype. This

crossover point defines a mating section that is used to affect a cross through LUT-by-LUT exchange operations.

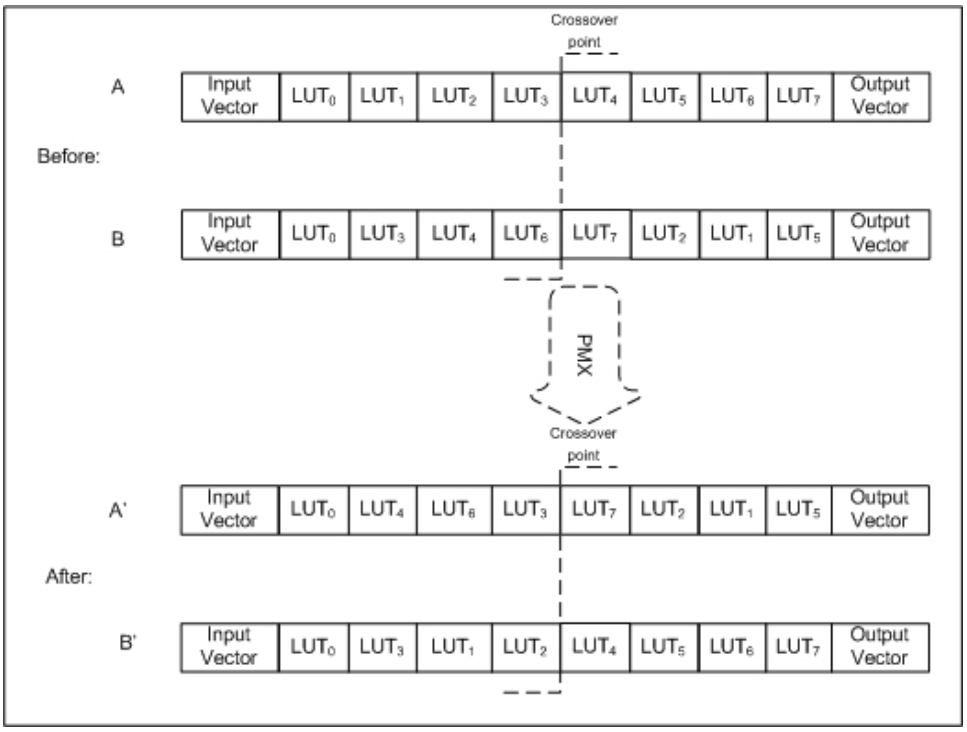

Figure 31: PMX operation on Genotype chromosomes

Figure 31 shows the crossover point that occurs in the position 4 of the LUT vector where PMX is implemented by position-wise exchange. The first step is to map configuration B to configuration A by exchanging the following aligned LUTs {(4,7),(5,2),(6,1),(7,5)}. This results in both configuration having duplicate elements and similar replacement mutation reoccurs to clean such correct functionality behavior. Applying PMX results in two new configurations A' and B'.

6.2.3. Consensus Based Evaluation (CBE)

An innovation of the OES architecture over conventional fault detection, diagnosis, and recovery strategies for the fault detection and fault recovery, Consensus Based Evaluation (CBE) approach was developed for fault detection and a GA approach was applied for fault repair in order to design an embedded system that exhibits some of the self-x properties essential for OC designs.

The GA used in the FPGA aforementioned in section 4.2 present some successful applications and demonstrates the benefit of both GA and reconfigurable device. The entities of GA used in this dissertation are analogical with the FPGA architecture but simpler than the real bitstream file. In other words, we only encode the information that can be manipulated in the bitstream to our genotype and apply specifically designed GA operators the bitstream.

The difference between repair and design is the difference in search space. The evolution repair strategy presented does not damage any functionality of the configurations. Actually the evolution results in some manipulated offspring of parents. Even if all of the configurations are fault-free, faulty physical resources may inhibit the configurations from generating the expected output. Therefore, the objective of the evolution is to obtain some specific configuration which works around faulty physical resource.

6.3. Experiment Configuration

6.3.1. FE and AE Failure Coverage

In the experiments, coverage and resolution of faults in the FE and faults in the AE are evaluated. The FE fault-handling experiments inject a stuck-at-zero or stuck-at-one fault at one of the FE's LUT input pins and the resolution process proceeds as described in Section 6.1. The AE fault-handling experiment utilizes a CBE-based approach to detect the faulty AE in the population. Once the fault is detected, the AS generates a new population for identified AEs, reconfigures them on the logic fabric sequentially in order to evaluate their correctness. After all the configurations are evaluated, CBE keeps detecting faults in that AE under repair, until the number of newly created configuration evaluations reach E_w. During the AE repair, the FE will reside on the chip and generate output even under fault impact conditions. The AE units are said to be functionally identical yet physically distinct due to the fact that they all contain the same functional elements with a constraint of identical number of I/O pins. This implies that as long as the AE is designed for the largest output word-width output by any FE, then all of the FEs can differ in function and even differ in output word-width by just tying any unused input pins of the AE to ground without any loss of generality.

6.3.2. Single vs. Multiple Fault Coverage

In order to determine the fault handling mechanism in the proposed system, two different fault models *Single-Failure Model* and *Multiple-Failure Model* are introduced. If *Single-Failure Model* is applied to the proposed system then the fault will be located either in a FE or an AE

component, but not simultaneously. Therefore the analysis of the evolutionary recovery operation will only focus on the faulty component without considering the other component's status. Whenever the AE component is undergoing an observable fault impact, the system will lose the monitoring functionality of the corresponding FE component. However, under *Single-Failure Model*, the FE component will be fault-free and maintain data throughput without error during that time period.

Alternatively, if the FE component is under the impact of the fault then the AE component notifies the AS that the wrong output came from the output of the FE component. Even when the FE component is under fault impact, the cold spare can provide a ready replacement for reconfiguration. Under the FE fault case there is no unavailability once the switch to TMR to identify the failed resource is completed. The failed FE can be repaired in the background via the GA as a refurbished CED mode has been restored.

For a single FE fault, the system availability, A_{SF}, is given by Equation (6.2). Let the number of correct behaviors of the FE that have been observed during the evolutionary recovery phase be denoted by F_c while the number of errant or discrepant behaviors of the FE is denoted by F_e. The quantity *1* represents the number of faulty outputs, i.e. exactly one output required to detect the fault during the original CED configuration. The coefficient *2* is the number of the reconfigurations required, i.e. one from CED to TMR, and one back from TMR to CED. The quantity β represents reconfiguration time expressed in the same time units as the computation time units, yielding:

$$A_{SF} = \begin{cases} 100\% & \text{if AE under single fault} \\ \dfrac{F_c}{F_c + 1 + 2\beta} & \text{if FE under single fault} \end{cases} \tag{6.2}$$

The next scenario represents the *Multiple-Failure Model*. If multiple faults occur in only either the FE component or the AE component exclusively, it yields the same behavior as the *Single-Failure Model* case because no matter how many faults that occur, the unit-under-test is the entire FE or AE itself during the fault detection step. On the other hand, if the AE and FE are under the impact of faults simultaneously, then the system will keep the FE online executing correct FE behavior if possible by introducing the S-FE.

If deployment of the S-FE is unsuccessful then group information will be used to repair the AE first. As long as the AE returns to normal, the FEs will be recruited into repair process. So under this strategy system still can maintain graceful degradation capability under the multiple fault impacts. The quantities F_c , F_e, and β in the second line of equation (6.3) have the same definition as equation (6.2), and the quantities F_{c1} and F_{e1} stands for the correct and faulty output number of the FE during the AE repair period, the (F_{c2}, F_{e2}) stands for the correct and faulty output number during the FE repair period and the n is number of reconfigurations of the FE. Hence, the system availability under multiple faults, A_{MF}, is given by:

$$A_{MF} = \begin{cases} 100\% & \textit{if AE under multiple faults} \\ \dfrac{F_c}{F_c + 1 + 2 \cdot \beta} & \textit{if FE under multiple faults} \\ \dfrac{F_{c1} + F_{c2}}{F_{c1} + F_{e1} + F_{c2} + F_{e2} + n \cdot \beta} & \textit{if AE and FE both under multiple faults} \end{cases} \tag{6.3}$$

6.3.3. Hardware Prototype

The case study example shown in Fig. 38 was implemented on the Xilinx Virtex II Pro as proof of concept to accompany other software simulations performed and presented in Section 5. Only a small number of resources are utilized for the AE and FE. The OES architecture in this case study consisting of a Full Adder FE unit with all of the elements in the AE Unit is realized using HDL implementation on the Xilinx Virtex II Pro FPGA using the GNAT library along with the MRRA framework and JTAG reconfiguration interface.

In Figure 32, the FE and AE units are shown in dashed boxes. The CS-LUT is shown in the dotted box. The Evaluator consists of XOR gates to check for any discrepancy between the FE units. There are three FE units of which only two are active during runtime, the third FE is a standby, i.e. S-FE, and will only become activated once a discrepancy is detected on the FE elements. Once a discrepancy is detected, the switching logic shown within the red box (contents not shown) will be used to activate the standby FE. TMR will be used in this case for the Evaluation Window during which Genetic Operators will be used to repair the faulty FE individual. Once evolution achieves repair, the repaired FE will now becomes the S-FE. This process is instantiated each time a FE discrepancy is detected.

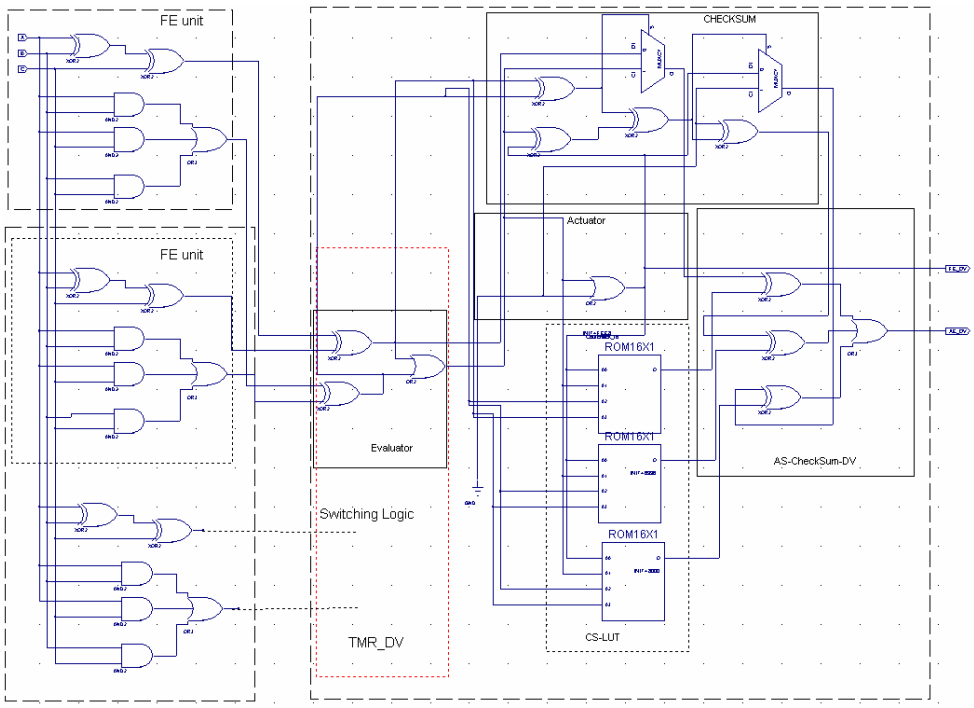

Figure 32: Gate Level Design of OES (Case study)

Notice that the inputs of the FE unit are connected to all FE units including the standby FE. Discrepancy in the two FE elements is detected using XOR gates fed to an OR gate. The output of the evaluator is fed to the Actuator that uses an XOR gate to send a signal labeled FE *Discrepancy Value (FE_DV)* that will initiate GA operators on the FE unit once a discrepancy is detected. The outputs of the two XORs checking the two outputs of the two FE elements along with the Evaluator output and Actuator output are all fed to a checksum unit consisting of 4-to-2 compressor tree. In this particular case study only one 4-to-2 compressor is needed. To check for any discrepancy between the Checksum element and the CS-LUT during runtime, a circuit similar to the evaluator circuit is used. The outputs of the Checksum element and CS-LUT are fed to XORs and the output of the XORs are fed to an OR gate. The output of the OR gate named

AE Discrepancy Value (AE_DV) will determine if a discrepancy is found between the two elements at runtime. The AE_DV is fed to the AS unit (not shown in figure) where it will be used along with CBE to confirm that the AE is in fact faulty and will cause the AS unit to initiate GA operators on the faulty AE element. Figure 33 produced by Xilinx ISE shows the physical layout of the design shown in Figure 32 on a Xilinx FPGA Virtex-II Pro.

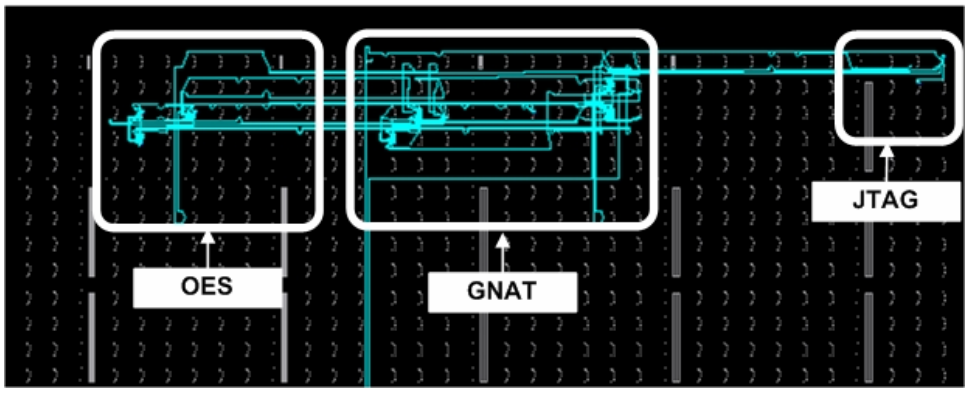

Figure 33: Physical Layout of OES system on FPGA with GNAT/JTAG shown

6.4. Result and Analysis

Three circuits were evaluated using the OES architecture, all of which are specified in Table 1 from the MCNC-91 benchmark [61]. The experiments implemented test the fault repair process on both the FE and AE components simultaneously. As previously discussed in Section 6.2, this will result in cascade of repair of both components via a single test scenario. In the FE fault recovery process, only Mutation and Cell-Swap operators are applied and the unit evolved to the fault-free state without utilization of population information. A fault was also induced on an input of LUT within the AE unit. During the AE fault recovery process, all three genetic

operators along with CBE were applied to evolve the AE unit to a refurbished status. The use of CBE along with GA operators proved to provide a large benefit in terms of number of repair iterations compared to conventional offline GA-based design-from-scratch-approaches [14, 15, 31]. Each experiment was executed for 10 runs on each circuit. The GA parameters were set as Mutation Rate=0.5, Cell-Swap rate=0.5 and Crossover rate=0.5 in all of the runs. The population size for AE is five and FE there is 2 active and one spare. The GA tournament selection rate was selected to be 2.

Table 10: MCNC-91 Benchmark Circuits Evaluated on OES Architecture

Circuit Name	Circuit Function	Inputs	Outputs	Approximate Gates
z4ml	2-bit Add	7	4	20
cm85a	logic	11	3	38
cm138a	Logic	6	8	17

The evolution repair strategy results in some manipulated offspring of parents. Even if all of the configurations are fault-free, faulty physical resources may inhibit the configurations from generating the unexpected output. Therefore, the objective of the evolution is to obtain some specific configuration which works around the faulty physical resource to eventually occlude it.

Figure 34 shows the fitness obtained for the cm85a circuit when a stuck-at-zero fault was injected at 48 different locations. Specifically the circuit was synthesized using Xilinx ISE to occupy 12 LUTs that each of which had 4 inputs, yielding 48 exclusive failure locations. The fitness of each cm85a circuit under each test scenario ranges from zero outputs correct up through a maximum of 2^{11}=2048 outputs correct because cm85a has 11 inputs.

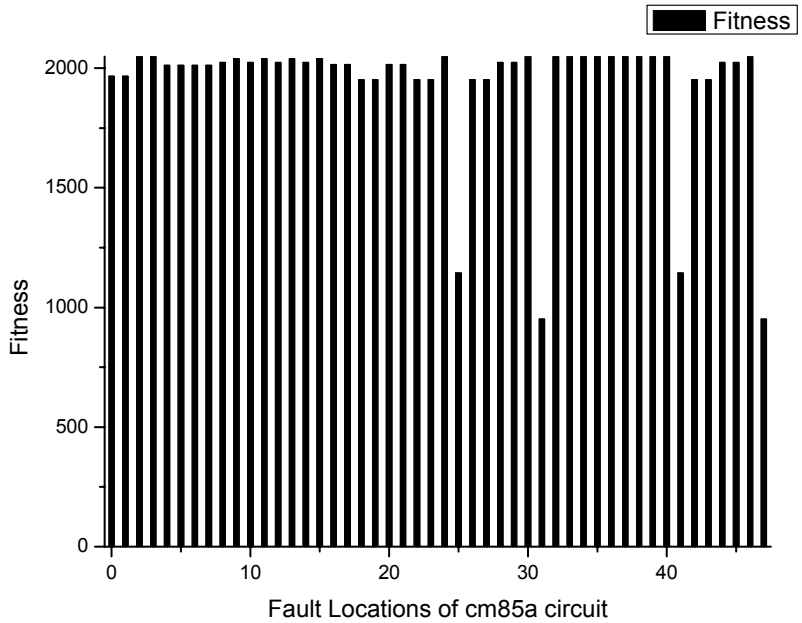

Figure 34: Fitness as a function of 12 LUTs with 48 fault locations tested

The three circuits' experimental results are listed in Table 11 through Table 13 which lists the system Availability during the repair phase. It is important to note that the system Availability outside the repair phase is by definition 100%. The column *n* denotes the measured number of reconfigurations for either the AE or the FE during the repair process during each test run. In last three columns, we assume the β is equal to 10^3, 10^4, and 10^5 respectively, to anticipate the system performance under different reconfiguration to computation ratios. The result presented shows that even spanning 3 orders of magnitudes of difference, the system performance can still be acceptable for some certain circumstances. When β=1000, the average system availability is 75.05% for the z4ml circuit and 82.21% for the cm85a and 65.21% for cm138a, all three may not exhibit graceful degradation but can keep partial functionality under the fault impact. The values of the redundancy for both FE (R_{FE}) and AE (R_{AE}) are calculated based on the ratio of unused LUT inputs over the total number of LUT inputs used on both AE and FE designs, respectively.

Table 11: z4ml Circuit Experiment Results

Circuit *z4ml Run*	R_{AE}=14.1% R_{FE}=25%	n	Fault-Free AE output		Fault-Impact AE output		System* Availability During Rep. $\beta = 10^3$	System* Availability During Rep. $\beta = 10^4$	System* Availability During Rep. $\beta = 10^5$
			F_{c1}	F_{e1}	F_{c2}	F_{e2}			
1	AE	15	20856		2003		80.45%	50.00%	8.97%
	FE	2	19979	2858	22				
2	AE	7	9403		917		72.99%	30.85%	4.24%
	FE	2	8914	1302	10				
3	AE	17	24899		2215		81.48%	54.20%	10.43%
	FE	2	23664	3380	8				
4	AE	11	14586		1702		78.10%	41.59%	6.59%
	FE	2	14234	1992	8				
5	AE	11	15474		1375		78.53%	42.47%	6.81%
	FE	2	14764	2036	2				
6	AE	3	3991		278		59.41%	15.58%	1.81%
	FE	2	3685	521	6				
7	AE	7	9612		767		73.10%	30.87%	4.25%
	FE	2	8929	1287	4				
8	AE	5	6880		444		68.78%	24.07%	3.06%
	FE	2	6334	877	7				
9	AE	17	23201		2084		81.01%	52.46%	9.80%
	FE	2	22068	3173	2				
10	AE	9	12622		1866		76.68%	38.65%	5.88%
	FE	2	12592	1831	6				
Average System Availability During Rep.							75.05%	60.54%	6.18%

* = system Availability outside the repair phase always equal to 100%.

Table 12: cm85a Circuit Experiment Results

Circuit cm85a Run	R_{AE}=14.1% R_{FE}=16.67%	n	Fault-Free AE output		Fault-Impact AE output		System* Availability During Rep. $\beta = 10^3$	System* Availability During Rep. $\beta = 10^4$	System* Availability During Rep. $\beta = 10^5$
			F_{c1}	F_{e1}	F_{c2}	F_{e2}			
1	AE	11	19479		1531		87.81%	44.95%	7.64%
	FE	2	16526	301	50	1			
2	AE	21	31371		3616		92.13%	58.65%	12.66%
	FE	2	28966	482	91	1			
3	AE	7	13092		1044		83.93%	35.90%	5.34%
	FE	2	11257	161	33	1			
4	AE	7	11202		1145		84.53%	37.06%	5.60%
	FE	2	11845	174	36	1			
5	AE	25	33405		2919		93.45%	64.11%	15.49%
	FE	2	36714	574	40	1			
6	AE	1	3724		96		53.91%	10.66%	1.18%
	FE	2	2358	45	35	1			
7	AE	11	16228		1341		87.90%	45.00%	7.65%
	FE	2	16543	284	52	1			
8	AE	7	10824		1127		83.66%	35.80%	5.33%
	FE	2	11219	199	47	1			
9	AE	3	4821		367		69.66%	19.20%	2.33%
	FE	2	4730	77	41	1			
10	AE	13	14438		2190		85.18%	39.78%	6.29%
	FE	2	13390	337	46	1			
Average System Availability During Rep.							82.21%	39.11%	6.95%

- = system Availability outside the repair phase always equal to 100%.

Table 13: cm138a Circuit Experiment Result

Circuit cm138a Run	R_{AE}=14.1% R_{FE}=20%	n	Fault-Free AE output		Fault-Impact AE output		System* Availability During Rep. $\beta=10^3$	System* Availability During Rep. $\beta=10^4$	System* Availability During Rep. $\beta=10^5$
			F_{c1}	F_{e1}	F_{c2}	F_{e2}			
1	AE	7	11696		1488		84.44%	37.07%	5.61%
	FE	2	11828	191	65	1			
2	AE	7	10071		759		66.23%	27.56%	4.03%
	FE	2	8484	2333	15	1			
3	AE	5	8296		754		64.07%	24.33%	3.38%
	FE	2	7057	1957	2	1			
4	AE	3	4624		450		54.87%	15.10%	1.83%
	FE	2	3724	1083	25	1			
5	AE	1	1849		94		37.31%	6.54%	0.71%
	FE	2	1404	398	24	1			
6	AE	7	11060		962		66.68%	29.20%	4.41%
	FE	2	9347	2672	3	1			
7	AE	23	31366		3067		73.75%	49.30%	11.42%
	FE	2	26732	7524	33	1			
8	AE	15	21769		1906		71.50%	41.89%	8.15%
	FE	2	18180	5258	28	1			
9	AE	9	12945		916		68.16%	31.91%	5.05%
	FE	2	10778	3044	20	1			
10	AE	7	9409		963		65.12%	26.36%	3.79%
	FE	2	7947	2269	25	1			
Average System Availability During Rep.							65.21%	28.93%	4.84%

* = system Availability outside the repair phase always equal to 100%.

Figure 35 shows the evolutionary process for the cm85a circuit which has 11 inputs and 3 outputs and a maximum fitness of 2^{11}=2048. During the repair process, only mutation and cell-swap operators are implemented because there is only single instance of FE under repair. Even if a random walk around the search space is achieved, then the results in the Fig.41 show that most of the time the circuit's fitness is above 1900 out of 2048 even during the existence of the fault

within the circuit resources. The reason of this phenomenon is either the inserted fault only impacts the circuit to a minor degree or because fan-in exceeds fan-out in this circuit. To better explain the concept behind Figure 35 and why the fitness and evolution behavior differs from a conventional Genetic Algorithms which improves monotonically over time, consider that in the OC case, the unit being evolved is always predefined at design time. What the GA in our system does is to explore a limited search space near that existing design to identify alternate physical resources to bypass a faulty input or faulty LUTs. Since the functionality of the unit is already predefined then the search space is limited to identifying distinct physical resources for occluding the fault to restore functionality. Hence, the GA is not exploring new designs from scratch, but restoring the lost functionality of the failed design. This is also demonstrated in Figure 35 where the stuck-at-zero fault is applied to all possible inputs of the FE, yet the fitness of the unit was on average above 1900 out of a maximum fitness of 2048. Hence, the slightly increasing non-monotonic curves in Figure 35 and Figure 36 can be observed.

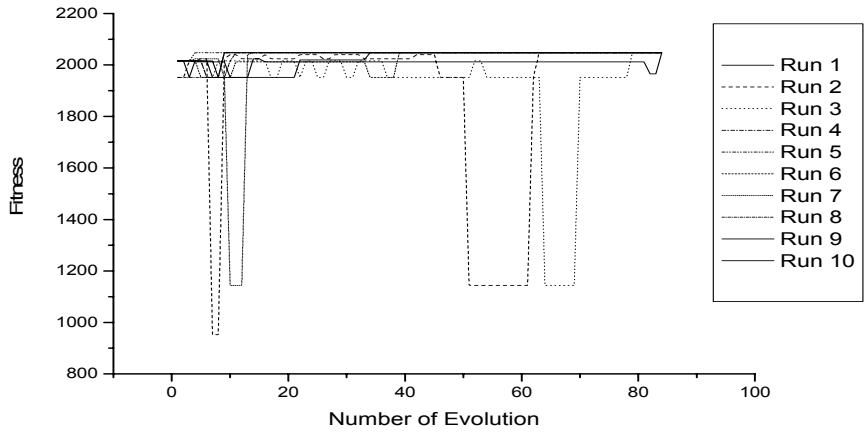

Figure 35: cm85a FE Evolutionary Recovery without CBE

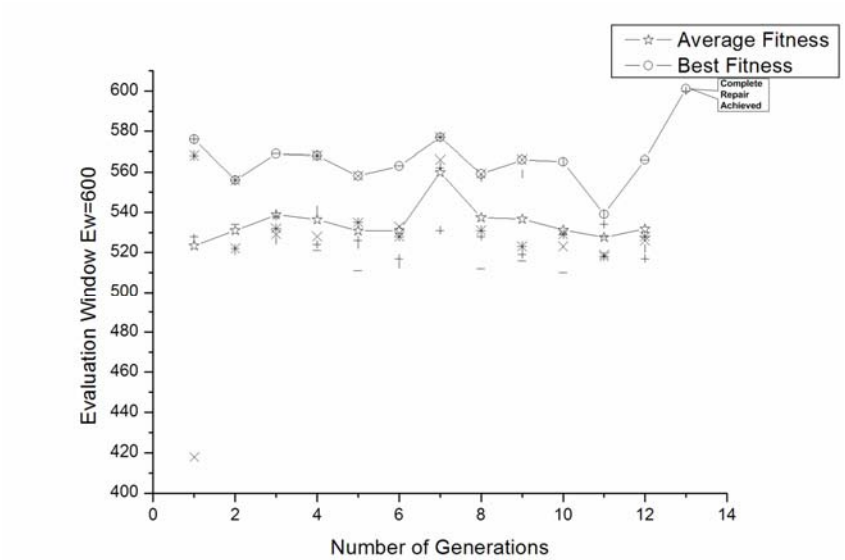

Figure 36: AE Evolutionary Repair for cm85a Circuit using CBE

Also in Figure 36, the cm85a AE evolutionary process is shown. The population information helps repair the circuit. The difference of this evolutionary process with the traditional GA is that the configurations are all correct, but the physical fault in the hardware resources that they utilize produces the unexpected behavior of the circuit. Instead of generating a new design, the evolution process only permutates the current design using different input line combination or different logic that occupies different physical resources. Because of the inherent redundancy which is generated by the synthesis tool, there is always a chance for new permutation occluding the physical failure. Therefore, the repair process is not time-consuming like the traditional GA process because of small search space.

When comparing Figure 35 and Figure 36, we clearly see the different benefit of the CBE-GA approach in comparison with just the random GA operation approach, respectively. It should be noted here that we are not applying the CBE-GA operator algorithm on the FE because in our design we are only utilizing GA operator to achieve fault-tolerant FEs. The population

information assists the evolution not only during the repair process and for future repairs as well. In the future, for any faults occurring in distinct physical locations but present in the same functional unit, the repair the configuration can used as a repair reference during the crossover which may help the reparation. Only the mutation and cell-swap operation explore the search space and while maintaining most of the time a graceful degradation property for circuit operation. However, such characteristics may not generally apply for all kinds of circuits. It may only manifest on certain types of circuit and for certain types of fault inserted, however it does apply for several different circuit types in the standard MCNC-91 benchmark.

CHAPTER 7: CONCLUSION

7.1. OVERVIEW

The original motivation of CBE approach combined conventional fault tolerance techniques and newly emerging reconfigurable devices to obtain improved system reliability and availability simultaneously. This dissertation is the first successful attempt to propose, design, implement and evaluate such components and architectures. Even though TMR/Standby approaches have been used separately, their benefits can be combined. Also because the characteristics of the FPGA, the reutilized physical resources under fault impact and partial functionality become feasible candidate solutions to increase mission lifetime. Furthermore, the EH approach enhances CBE self-repair capacity make it more suitable for real applications.

Several advantages of the proposed CBE approach are presented in this dissertation. The conventional TMR, Standby and dynamic TMRSB systems are limited by the hardware platform which may not have automatic reconfiguration capacity and exhibit overhead with the standby components. Without reconfiguration capacity, the standby components must not only use mutually exclusive hardware resources from failed components, but also add specific extra switch components which will decrease the system reliability. However, with the FPGA device, all the previous obstacles can be considered as trivial factors because of inherent reconfigurable characteristic of FPGAs. The bitstream files occupy less than several hundred Kilobyte storage space which will configure a multimillion gate sized FPGA device to provide superior performance over extra switching components. Overall inherent properties of the FPGA device provide the TMRSB system a renewed opportunity as a fault tolerance technique.

7.2. Evolvable Hardware and CBE

Furthermore, partial online EH regeneration essentially defines a problem that is different from offline EH design. A population of working designs can facilitate restoration of functionality by providing diverse alternates since the alternative configurations are only occupy small area of the memory. Conventional fitness evaluation associates a rigid individual-centric fitness measure defined at design-time. CBE uses instead, a self-adapting population-centric assessment method at run-time. Population-centric assessment methods such as CBE can provide an additional degree of adaptability and autonomy. CBE relies on the consensus observed among a group of individuals to evolve and adapt fitness criteria for individual members, thus providing graceful degradation. By utilizing outlier detection techniques that work temporally without the need for exhaustive testing, CBE provides a fault tolerance technique that maximizes device throughput during the fault detection process.

Under CBE, the outlier detection mechanism worked as shown in Figure 18, the measured performance of an individual over 28 iterations during the repair process for the z4ml circuit. In this particular experiment, the reintroduction rate used was 0.20, with both the cell-swapping rate and the input permutation rate set to 20%. As shown in Figure 18a, whenever the discrepancy of the individual rises above the average discrepancy of the population, the individual is reloaded onto the FPGA, as evidenced by Figure 18b. Conversely, when the individual discrepancy is equal to, or less than the average discrepancy of the population, the individual is not reloaded, or reloaded less than the average member of the population. This shows the autonomous behavior capability of CBE.

While the pre-existing methods focus on creating a single fully-fit configuration, CBE extends this to maintaining a population of solutions that have a higher average fitness. This ensures that the adaptability of a population of viable alternatives to a variety of unanticipated faults. An additional benefit of maintaining a population of diverse partially-fit individuals is that when the inputs to the system are localized to a subset of the set of all possible inputs, even partially-fit individuals can assist in generating expected outputs, thereby improving the rate of viable throughput. The population size evaluated was 20 and 30 for TMR/DUPLEX MCNC91 benchmark experiment where each branch has 10 candidates. Considering most benchmark circuits are less than 100 equivalent gates, 10 should be appropriate number of alternative possible designs.

CBE improves on existing fault tolerance techniques for reconfigurable devices by providing an adaptive, evolutionary algorithm that leverages diversity inherent in a population of solutions to evolve solutions at runtime with minimal downtime. The system availability shown in Table 11, Table 12, and Table 13 are keeping the system executing even under the presence of a fault impact. Finally, an additional benefit of CBE is that fitness evaluation becomes independent of the application running on the FPGA enabling *model-free* assessment during evolutionary repair. For example, experiments for the multiplier in section 5.3 show CBE did not require any behavior model nor truth table for the fitness function which is superior for adaptive system.

Leveraging the property that even partially-fit individuals respond correctly to some subset of inputs, CBE is shown to maintain adaptable levels of system availability in the presence of defective configurations. This allows for graceful degradation using population characteristics without requiring a circuit-specific fitness function. Additionally, the proposed approach

requires no specially constructed test vectors, as the response of individuals to real-time inputs forms the basis for evaluation. In Table 9, the number of the iteration number for the repair all of the faulty individuals by 50%. This recasts the emphasis in EHW for repair from exhaustive testing to a focus on functionality based only on the relevant inputs which are encountered in the embedded application.

Rather than trying to anticipate operating conditions, CBE utilizes runtime information to adapt to the subset of possible faults which are actually present and being articulated. Even pervasive faults that may completely deplete all viable spares from the dormant population are shown to be recoverable, given adequate population size and diversity. We can see from Figure 19 and Figure 20, the population size 20 and 30 for Duplex and TMR models respectively, are sufficient to distinguish and isolate fault and repaired the faulty individual with operational throughput. This focus on Recovery Complexity emphasizes use of a diverse population of previously correct alternatives as compared to a single failed seed configuration. Current work includes the development of a self-contained System-on-Chip implementation of self-healing EHW using the Multi-Layer Runtime Reconfigurable architecture [64] as a partial reconfiguration framework for Xilinx SRAM-based FPGAs.

7.3. Organic Computing Architecture

Even though model free circuits design are implemented and evaluated by CBE, this research was taken further to construct an autonomous self-governing architecture in order to make the whole CBE proposed applicable. In this dissertation, we developed an OES architecture for sustainable performance of reconfigurable FPGA soft cores. The architecture was developed

using an OC observer/controller organization and regeneration with Genetic Operators. Other innovations included provision of availability during regeneration, outlier-driven repair assessment, and a uniform design for the AEs. The design objective of developing an architecture that exhibits self-adaptation and self-healing properties can be attained using such techniques for completely autonomic operation without human intervention. The OES architecture is capable of handling many single fault scenarios and several multiple fault scenarios.

Experimental results strongly supported our design objectives were met. Using logic circuits from the MCNC-91 benchmark set, we assume the β is equal to 10^3, 10^4, and 10^5 respectively, to anticipate the system performance under different reconfiguration to computation ratios. The result presented shows that even spanning 3 orders of magnitudes of difference, the system performance can still be acceptable for some certain circumstances. When β =1000, the average system availability is 75.05% for the z4ml circuit and 82.21% for the cm85a and 65.21% for cm138a, all three may not exhibit graceful degradation but can keep partial functionality under the fault impact. The synthesized OES architecture was prototyped on Xilinx Virtex II Pro FPGA device supporting partial reconfiguration to demonstrate the feasibility of the OES architecture for intrinsic regeneration of the selected circuit. Through application of genetic operators for mutation and crossover, the OES architecture successfully achieved full repair of faulty element in the presence of stuck-at-zero and stuck-at-one faults. This integrated the use of redundant LUTs inherited in the FPGAs design. This integrated approach provides an innovation in fault-handling capability not only for the FEs, but also for the AEs as well.

The CBE-based approach developed herein can outperform conventional GA-based approaches for self-healing due to its search in a smaller repair space as opposed to an unbounded design space. The CBE-based approach relies heavily on population information and thus can be applied to the AEs directly. The population information assists the evolution not only during the repair process, but also for future repairs of a different AE as well. In particular, for future faults occurring in distinct physical locations, but within units having the same functional behavior as a previously handled fault, then the repaired configuration can provide a repair reference during the crossover which may help the reparation. Only the mutation and cell-swap operation explore the search space while maintaining the majority of the time a graceful degradation property for circuit operation. However, such characteristics may not apply in general for all kinds of circuits and for certain types of fault behaviors. However, it does apply to several different circuit types in the standard MCNC-91 benchmark under single and multiple stuck-at faults.

7.4. Future Work

For future work, one area to investigate is to develop OES architecture space-based reconfigurable embedded architectures. The most problem currently confronted in FPGA based EH research is more platform support and more low level API support. In order to fully utilize the reconfiguration capacity of the FPGA, new embedded GA oriented architectures are demanded for those purposes. For examples, a complete working OES prototype on Xilinx Virtex 4 and Virtex 5 FPGA chip which supports more advanced online reconfigurations is considered for using more advanced GA operations. It's also possible explore other GA operators and develop more methodologies for fault-isolation and fault-correction. Another

obstacle is dynamic reload and recompile overhead where each newly evolved configuration must be recompiled for reconfiguration which impact the CBE idea being implemented because of the recompilation time and the reload time β also a factor of decreasing issue for further utilization of the CBE approach. Those questions are excellent the research topics for future development.

LIST OF REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* The MIT Press, 1992.

[2] D. M. Poole, Alan & Goebel, Randy, *Computational Intelligence: A Logical Approach*: Oxford University Press, 1998.

[3] T. Jones, "*Evolutionary Algorithms, Fitness Landscapes and Search*," in *Computer Science*. vol. Phd Albuquergue,NM: University of New Mexico, 1995.

[4] M. Phillip and K. V. Ganesh, "*Evolving Combinational Logic Circuits Using a Hybrid Quantum Evolution and Particle Swarm Inspired Algorithm*," in *Proceedings of the 2005 NASA/DoD Conference on Evolvable Hardware*, 2005.

[5] D. E. Goldenberg and R. Lingle, "*Alleles, Loci, and the traveling salesman problem*," in *Proceeding of the First International Conference on Genetic Algorithms and Their Application*, 1985, Pittsburgh, PA, pp. 154-159.1985.

[6] L. J. Fogel, Owens, A.J., Walsh, M.J., *Artificial Intelligence through Simulated Evolution*: John Wiley, 1966.

[7] H.-G. Beyer, "*The Theory of Evolution Strategies*," Springer, 2001.

[8] J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the Evolutionary Design of Digital Circuits -- Part I " *Journal of Genetic Programming and Evolvable Machines,* vol. 1, p. 28, 2000.

[9] J. F. Miller and P. Thomson, "*Cartesian Genetic Programming*," in *Genetic Programming, Proceedings of EuroGP'2000*, 2000, Edinburgh, British pp. 121--132, 2000.

[10] J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the Evolutionary Design of Digital Circuits -- Part II " *Journal of Genetic Programming and Evolvable Machines,* vol. 3, p. 30, 2000.

[11] W. a. S. Browne, D "*Kernel-based, ellipsoidal conditions in the real-valued XCS classifier system*," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, 2005, Washington DC, USA, pp. 1835-1842, 2005.

[12] M. D. T. "Stützle, *Ant Colony Optimization*," MIT Press, 2004.

[13] V. K. Vassilev and J. F. Miller, "*Embedding landscape neutrality to build a bridge from the conventional to a more ef-ficient three-bit multiplier circuit*," in *Proceedings of the 2nd Genetic and Evolutionary Computation Conference*, 2000, San Francisco, CA, 2000.

[14] J. D. Lohn, G. Larchev, and R. F. DeMara, "*A Genetic Representation for Evolutionary Fault Recovery in Virtex FPGAs*," in *Proceedings of the5th International Conference on Evolvable Systems (ICES)*, 2003, Trondheim, Norway. March 17-20, 2003.

[15] J. D. Lohn, G. Larchev, and R. F. DeMara, "*Evolutionary Fault Recovery in a Virtex FPGA Using a Representation That Incorporates Routing*," in *Proceedings of the17th International Parallel and Distributed Processing Symposium*, 2003, Nice, France, April 22-26, 2003.

[16] S. Vigander, "*Evolutionary Fault Repair of Electronics in Space Applications*," in *Computer and Information Science*. vol. Master Trondheim, Norway: Norwegian University of Science and Technology (NTNU), 2001, p. 50.February 28, 2001.

[17] K. Zhang, R. F. DeMara, and C. A. Sharma, "*Consensus-based Evaluation for Fault Isolation and On-line Evolutionary Regeneration*," in *Proceedings of the International Conference in Evolvable Systems (ICES'05)*, 2005, Barcelona, Spain, pp. 12-24.September 12 - 14, 2005.

[18] M. Abramovici, J. M. Emmert, and C. E. Stroud, "*Roving STARs: An Integrated Approach To On-Line Testing, Diagnosis, And Fault Tolerance*

For FPGAs In Adaptive Computing Systems," in *The Third NASA/DoD Workshop on Evolvable Hardware*, 2001, Long Beach, Cailfornia, pp. 73-92.July 12-14, 2001.

[19] J. v. Neumann, "*Probabilistic logics and synthesis of reliable organisms from unreliable components*," in *Automata Studies*, C. S. a. J. McCarthy, Ed.: Princeton University Press, 1956, pp. 43--98.

[20] I. Xilinx, "*Development System Reference Guide v3.5.1*," April, 2003.

[21] I. Xilinx, "*Virtex-II Pro Platform FPGA User Guide v204*," August 2004.

[22] I. Xilinx, "*Virtex-II Pro and Virtex-II Pro X platform FPGAs: Complete data sheet*," October. 2005.

[23] H. Schmeck, "*Organic Computing-A New vision for distributed Embedded Systems*," in *Proceedings of Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, 2005, Seattle, Washington, pp. 201-203.May 18-20, 2005.

[24] R. P. Würtz, "*Organic Computing methods for face recognition*," 2005.

[25] C. Müller-Schloer, "*Organic computing: on the feasibility of controlled emergence*," in *Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2004, Stockholm Sweden, pp. 2-5.2004.

[26] J. Z. A. Bouajila , W. Stechele , A. Herkersdorf , A. Bernauer , O. Bringmann , and W. Rosenstiel "*Organic Computing at the System on Chip Level*," in *Proceedings of the IFIP International Conference on Very Large Scale Integration of System on Chip (VLSI-SoC 2006)*, 2006, pp. 338-341.October 2006.

[27] H. Kasinger and B. Bauer, "*Combining Multi-agent-system Methodologies for Organic Computing Systems*," in *Proceedings of the 3rd International Workshop on Self-Adaptive and Autonomic Computing Systems (SAACS 05)*, 2005, Copenhagen, Denmark, pp. 160-164.August 2005.

[28] J. Branke, M. Mnif, C. Müller-Schloer, H. Prothmann, U. Richter, F. Rochner, and H. Schmeck, "*Organic Computing - Addressing Complexity by Controlled Self-organization*," in *Proceedings of ISoLA 2006*, 2006, Paphos, Cyprus, pp. 200-206.November 2006.

[29] G. Lipsa, A. Herkersdorf, W. Rosentiel, O. Bringmann, and W. Stechele, "*Towards a Framework and a Design Methodology for Autonomic SoC*," in *Proceedings of the Second International Conference on Autonomic Computing (ICAC'05)*, 2005, Washington, DC, pp. 391-392,June 2005.

[30] D. Keymeulen, R. S. Zebulu, Y. Jin, and A. Stoica, "Fault-Tolerant Evolvable Hardware using Field Programmable Transistor Arrays," *IEEE Transaction on Reliabilty,* vol. 49, p. 12, September 2000.

[31] J. D. Lohn and R. F. Demara, "*A Co-evolutionary Genetic Algorithm for Autonomous Fault-Handling in FPGAs*," in *Proceedings of the Sixth International Conference on Military and Aerospace Programmable Logic Devices (MAPLD-2002)*, 2002, Laurel, Maryland.September 10-12, 2002.

[32] O. B. A. Bernauer , W. Rosenstiel, A. Bouajila , W. Stechele , and A. Herkersdorf, "*An Architecture for Runtime Evaluation of SoC Reliability*," in *INFORMATIK 2006 - Informatik für Menschen,Lecture Notes in Informatics, Köllen Verlag*, 2006, Köllen Verlag, pp. 177-185,2006

[33] A. B. A. Bouajila, A. Herkersdorf ,W. Rosenstiel , O. Bringmann , and W. Stechele "*Error Detection Techniques Applicable in an Architecture Framework and Design Methodology for Autonomic SoCs*," in *1st IFIP International Conference on Biologically Inspired Cooperative Computing (BICC 2006)*, 2006, Boston, MA, pp. 107-113.August 2006.

[34] N. Bergmann and J. WILLIAMS, "*Egret: a platform for reconfigurable system-on-chip*," in *Proceedings of 2003 IEEE International Conference on Field-Programmable Technology (FPT)*, 2003, Tokyo, Japan, pp. 340-343, December 2003.

[35] A. Avizienis, "Toward systematic design of fault-tolerant systems," *Computer,* vol. 30, pp. 51-58, 1997.

[36] J. Becker and M. Hübner, "*Run-time Reconfigurability and other Future Trends*," in *Proceedings of the 19th Annual Symposium on Integrated Circuits and Systems*, 2006, Design Ouro Preto, MG, Brazil, pp. 9-11.August 2006.

[37] K. Paulsson, M. Hübner, and J. Becker, "*Strategies to On-line Failure Recovery in Self-Adaptive Systems based on Dynamic and Partial Reconfiguration*," in *Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, 2006, Istanbul, Turkey, pp. 288-291,June 2006.

[38] C. Carmichael, M. Caffrey, and A. Salazar, "*Correcting single-event upsets through Virtex partial configuration*," 2000.

[39] C. Stroud, J. Sunwoo, S. Garimella, and J. Harris, "*Built-In Self-Test for System-on-Chip: A Case Study. In Proceeding of the International Test Conference (ITC'04)*," in *Proceeding of the International Test Conference (ITC'04)*, Charlotte, NC, pp. 837-846, October 2004.

[40] K. Sekar, P. Sanchez, S. Dey, Y. Cheng, and L. Chen, "*Embedded Hardware and Software Self-Testing Methodologies for Processor Cores*," in *Proceeding of the 37th Conference on Design Automation (DAC'00)*, 2000, Los Angeles, Ca, pp. 625-630, 2000.

[41] F. L. Kastensmidt, L. Sterpone, L. Carro, and M. S. Reorda, "*On the optimal design of triple modular redundancy logic for SRAM-based FPGAs*," in *Proceedings of Design, Automation and Test in Europe, 2005*, 2005, Washington, DC, pp. 1290-1295 Vol. 2, 2005.

[42] C. J. Milliord, C. A. Sharma, and R. F. Demara, "*Dynamic Voting Schemes to Enhance Evolutionary Repair in Reconfigurable Logic Devices*," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig'05)*, 2005, Puebla City, Mexico, pp. 8.1.1-8.1.6.September 28 - 30, 2005.

[43] M. Toussaint and C. Igel, "Neutrality: A Necessity for Self-Adaptation," *Natural Computing: an International Journal,* vol. 2, pp. 117-132, June 2003.

[44] C. Carmichael, E. Fuller, P. Blain, and M. Caffrey, "*SEU Mitigation Techniqies for Virtex FPGAs in Space Applications*," in *The 2nd annual Military and Aerospace Applications of Programmable Devices and Technologies Conference*, 1999, Laurel, Maryland, pp. 24-32.September 28-30 1999.

[45] I. Xilinx, "*Triple Module Redundancy Design Techniques for Virtex FPGAs*," November 2001.

[46] P. K. R. Samudrala, J.; Katkoori, S., "Selective triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs," *IEEE Transactions on Nuclear Science,* vol. 51, p. 13, Oct 2004.

[47] L. Sterpone and M. Violante, "Analysis of the robustness of the TMR architecture in SRAM-based FPGAs," *Nuclear Science, IEEE Transactions on,* vol. 52, pp. 1545-1549, 2005.

[48] S. Habinc, "*Functional Triple Modular Redundancy (FTMR): VHDL Design Methodology for Redundancy in Combinatorial and Sequential Logic*," Gaisler Research, Sweden December 2002.

[49] S. S. Gokhale and M. R.-T. Lyu, "A Simulation Approach to Structure-Based Software Reliability Analysis," *IEEE Transactions on Software Engineering,* vol. 31, pp. 643-656, August 2005.

[50] R. F. DeMara and C. A. Sharma, "*Self-checking fault detection using discrepancy mirrors*," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, 2005, Las Vegas, Nevada, pp. 311-317.June 27 – 30, 2005.

[51] F. Lombardi, N. Park, M. Al-Hashimi, and H. H. Pu, "*Modeling the dependability of N-modular redundancy on demand under malicious agreement*," in *Proceedings of the Pacific Rim International Symposium on Dependable Computing*, 2001, Washington, DC, p. 68.December 17 - 19, 2001.

[52] J. Khakbaz and E. J. McCluskey, "Concurrent error detection and testing for large PLA's," *Electron Devices, IEEE Transactions on,* vol. 29, pp. 756-764, 1982.

[53] J. F. Bartlett, "*A nonstop kernel*," in *Proceedings of the Eighth ACM symposium on Operating systems Principles*, 1981, Pacific Grove, California, pp. 22-29.1981.

[54] S. Sharma, J. Chen, W. Li, K. Gopalan, and T.-c. Chiueh, "*Duplex: A Reusable Fault Tolerance Extension Framework for Network Access Devices*," in *Proceedings of 2003 International Conference on Dependable Systems and Networks (DSN 2003)*, San Francisco, CA, June 2003.

[55] A. I. J. Song, E. Levy, and D. Dias, "Architecture of a Web Server Accelerator," *Computer Networks (Amsterdam, Netherlands: 1999),* vol. 38, pp. 75-97, 2002.

[56] F. Lima, C. Carmichael, I. Fabula, R. Padovani, and R. Reis, "*A fault injection analysis of Virtex FPGA TMR design methodology*," in *Radiation and Its Effects on Components and Systems 6th European Conference*, 2001, San Jose, CA, pp. 275- 282.Sept 10-14, 2001.

[57] K. S. Trivedi, "*Probability and Statistics with Reliability, Queuing and Computer Science Applications*," 2nd ed, Wiley-Interscience, 1982, pp. 375-378.

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

[59] P. J. Rousseuw and A. M. Leroy, *Robust regression and outlier detection*, 1st ed. New York: Wiley & Sons, 1987.

[60] D. A. Gwaltney and M. I. Ferguson, "*Enabling the on-line intrinsic evolution of analog controllers*," in *Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on*, 2005, pp. 3-11, 2005.

[61] S. Yang, "*Logic Synthesis and Optimization Benchmarks, Version 3.0 Tech. Report, Microelectronics.Centre of North Carolina*," 1991.

[62] X. Yao and T. Higuchi, "Promises and Challenges of Evolvable Hardware," *IEEE Transaction on Systems,Man, and Cybernetics- Part C: Applications and Reviews,* vol. 29, pp. 87-97, 1999.

[63] P. C. Haddow and G. Tufte, "*Bridging the genotype-phenotype mapping for digital FPGAs*," in *Evolvable Hardware, 2001. Proceedings. The Third NASA/DoD Workshop on*, 2001, Long Beach, CA, USA, pp. 109-115, July 12-14, 2001.

[64] H. Tan and R. F. DeMara, "A Multi-layer Framework Supporting Autonomous Runtime Partial Reconfiguration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 16, issue 5, pp. 504-516, May 2008.