A Combinatorial Group Testing Method for FPGA Fault Location

Carthik A. Sharma and Ronald F. DeMara
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
casharma@mail.ucf.edu

Abstract

Adaptive fault isolation methods based on discrepancy-enabled pairwise comparisons are developed for reconfigurable logic devices. By observing the discrepancy characteristics of multiple Concurrent Error Detection (CED) configurations, fault isolation is realized without requiring additional test vectors or data coding schemes. Hence the reprogrammability of Field Programmable Gate Arrays (FPGAs) is utilized to examine CED alternatives in succession. Results show that for a reprogrammable device with one million resources, where 50% of the resources are used on an average by the target application, fault isolation can be achieved in as few as 28 iterations. The effect of resource utilization, the number of competing candidate solutions, and the number of unit resources are analyzed and the performance of a halving-based algorithm for fault isolation are quantified.

1 Introduction

Efficient detection and isolation of faults within logic devices are fundamental issues in the design of dependable systems for mission-critical applications. While traditional approaches to these problems rely on unique instances of dedicated hardware elements and/or extensive testing involving exhaustive or pseudo-exhaustive test vectors, this paper develops a new technique based on *runtime reconfigurability* and *competitive dueling*.

In Field Programmable Gate Array (FPGA) devices, the available number of unit cells such as *Look Up Tables* (LUTs) can comprise many thousands of physical resources. In this paper, the difficult problem of rapidly identifying a failure among these resources is addressed by extending methods from the algorithmic work on *Combinatorial Group Testing* (CGT) [1]. These concepts are used to develop new adaptive methods that utilize only the FPGA's usual runtime inputs to identify and isolate faults. This maximizes the device's online availability even while fault isolation is in progress.

The proposed techniques are evaluated using analytical and experimental methods for target implementation on SRAM-based FPGAs. With production exceeding 100 million units per year, SRAM-based FPGA devices are frequently used in a wide range of embedded applications requiring high levels of reliability and availability. Reconfigurable devices, such as FPGAs, enable new fault handling techniques where the repair process can take place *online* when the hardware is in active use, or *offline* when the refurbishment occurs outside the dataflow of the normal computational throughput. The emphasis of this paper is on fast and reliable fast isolation online using an adaptive algorithm.

Runtime fault detection without using special test vectors is achieved in a Concurrent Error Detection (CED) [5] strategy by comparing the outputs of two identical functional circuits for discrepancies. The discrepancy detector provides information regarding whether or not the outputs of two competing configurations resident on the FPGA produce outputs which are in bitwise agreement with each other. Using only this information, an adaptive method for reconfiguring the FPGA's functional logic can enable fault isolation because alternative CED configurations use varying subsets of resources. When these instances of the functional elements are paired over time, the accumulated correctness behavior along with their resource utilization characteristics are used to isolate the physical resource fault.

Fast fault detection and isolation techniques are highly relevant to many embedded device applications, including remote sensing, applications in hazardous environments, and space missions. For instance, deep space satellites such as Stardust contain over 100 FPGA devices [2] while aerospace applications routinely employ FPGAs extensively for tasks ranging from launch control to signal processing. SRAM-based FPGAs are of significant importance due to their high density, unlimited reprogrammability, and growing use in mission-critical/safety-impacting applications.

Techniques that enable online fault detection, isolation, and refurbishment by reprogramming FPGAs once a failure occurs provide an attractive alternative to traditional redundancy-based techniques. Reconfiguration offers the potential for resolving permanent degradation due to radiation-induced stuck-at-faults, thermal fatigue, oxide breakdown, electromigration, and other failures. Potential benefits include recovery without the increased weight and size normally associated with distinct spares that are configured at design-time before a specific failure occurs. Also, failures need not be precisely diagnosed due to automatic evaluation of the FPGA's residual functionality while in-circuit. Fault location methods provide inputs to the repair mechanism which accelerate the repair process, and reduce the search space that consists of candidate solutions. The fault isolation technique identified in this paper is one such method for isolating faults with low latency.

A common limitation facing many fault detection schemes is that the failure detector itself may fail. A fault involving the checker may be undetectable or result in the corruption of otherwise valid outputs. Traditional approaches to fault-detection typically rely on coding-based schemes [3][4] or redundancy using a single *voter*, *comparator* or *error detector*[5][6]. *Triple Modular Redundancy* (TMR)[7][8] approaches rely on three parallel instances of the functional logic to compute the output in triplicate and a majority voting element to determine consensus and hence the asserted output. The proposed system uses the output of the detector element for fault isolation, in addition to detection. In a duplex redundant CED system, the problem of fault detection is simplified as the outputs of the two elements will be identical in the absence of at least a single fault..

2 Problem Definition

In order to better understand the problem at hand, consider an analogy termed the *Treasurer's Problem* which is related to the Counterfeit Coin Problem [1]. The Counterfeit Coin Problem is extended here by analogy to support arbitrary groupings of logic cells within FPGAs. In this Treasurer's Problem, legitimate coins are made of gold, with the face value of the coins being proportional to their weight. However, some counterfeit coins have other metals mixed in with the gold, and these counterfeit coins are to be identified and removed. The weight of an impure coin is different from the weight of pure coins of the same denomination. The treasurer must inspect large quantities of coins for authenticity. Most significantly, since the number of counterfeit instances is small relative to the total number of coins present, the treasurer does not weigh the coins individually. Instead the coins are in a vat, and the treasurer retreives coins from the vat to fill bags containing exactly 100 monetary units worth of coins. The number of coins in each bag may vary because of their multiple denominations, yet due to the property that their mass is proportional to their denomination then only two equally-valued legitimate bags will display equal weight.

Using a pan balance, the treasurer compares the weight of two bags at a time to determine whether they are equal weight or not. The coins from the bags may be returned to the vat after weighing, so that they can be filled in other bags later after shuffling. Given these pre-conditions, a number of questions arise about how the treasurer will identify any faulty coinage such as: How many weighings will the treasurer need to identify bags containing the impure coins? Can the impure coin be identified, if there was only one?

These questions are analagous to the problems addressed in this paper for identification of a faulty physical resource used by a functional arrangement of FPGA configurations. FPGA devices are composed of an array of logic resources such as LUTs that are utilized by functional configurations just as the coins are grouped into a bag for weighing. A digital design can be mapped onto the resources on an FPGA in several ways, just like a bag worth 100 monetary units can be filled with coins of different denominations in several different ways. When one of the resources used by a configuration is faulty, the output of the configuration in response to an input may be faulty. Identifying the faulty resource from among many fault free resources, without testing the resources individually is a challenging task. Exhaustive testing of the individual resources is time consuming which takes the device offline and reduces its availability. By analogy, if the coins were weighed and checked individually, the time required would be phenomenal to locate a single fault out of thousands of resources. Instead, we recast the problem of identifying the faulty resource into one of making choices for group comparison from among the given FPGA configurations.

A novel fault-handling scheme based on pairwise comparison of competing configurations that utilize resources from a common pool is proposed. The fault isolation scheme should isolate faults with minimal latency without using a specialized block design, or any special test inputs. It should also be robust against single-faults that could affect the resources in the device. The proposed approach does not require any additional test inputs; only the normal dataflow inputs that are applied. The functionally equivalent configurations used for operation are pre-designed to realize the required logic functions. The property of reconfigurability inherent in FPGAs is utilized to accelerate the fault isolation by intelligently shuffling the resources used by individual configurations.

The arrangement of competing configurations on an FPGA is as shown in Figure 1. The proposed approach to hardware regeneration operates by comparing the outputs of a pair of physically distinct but functionally identical logic configurations that are loaded onto an FPGA. These configurations are identified as *Functional Logic L* (for left half CED configuration) and *Functional Logic R* (for right half CED configuration). These are loaded from the offchip EEPROM or RAM. The normal data throughput inputs to the FPGA are applied to both configurations simultaneously. The outputs of the competing configurations are compared using a discrepancy detector, which is mirrored over the two competing halves as shown. The discrepancy mirror is self-testing as it is instantiated equally among the competing configurations and thus accounts for faults in the resources used to realize the detector itself. The discrepancy detector's output shows whether or not the outputs of the configurations match. If the outputs do not match, then one of the two competing configurations utilizes a faulty hardware resource. The Data Output is propagated outside the FPGA only when there is no discrepancy between Functional Logic L's output and Functional Logic R's output. Moreover, the results of the discrepancy detection serve as inputs to control the fault isolation algorithm. The detailed design of the self-testing discrepancy detector and verification of its operation are presented in [9].

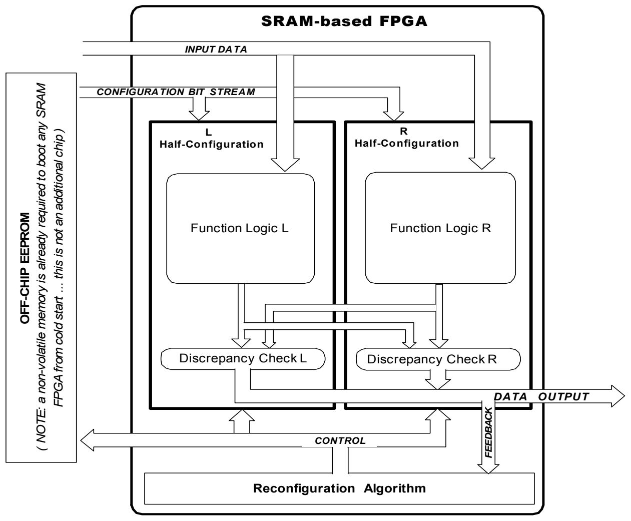

Figure 1: Overview of FPGA Operation for Repetitive Pairwise Evaluation

A mathematical representation of the proposed isolation scheme is developed. The underlying principle of operation is to use repetitive pairing of competing configurations. The properties of the proposed isolation mechanism are identified. Conditions that are conducive to expedite the fault isolation process are explored.

3 Resource Representation

The problem of locating faults using Iterative Pairing can be represented using the following mathematical model:

The set of all competing configurations is represented by **S**. Set $\mathbf{C_k}$ represents the resources utilized by configuration *k*. Each competing configuration k, $1 < k < |S|$ has a unique binary *Usage Matrix* $\mathbf{U_k}$, $1 < k < p$, with elements $\mathbf{U_k[i,j]}$, $1 < i < m, 1 < j \; n$, where *m* and *n* represent the rows and columns in the device layout respectively . Elements $U_k[i,j] = 1$ denote the usage of resource *(i, j)* by C_k. The *History Matrix* **H**, with elements **H[i,j]** $1 < i < m$, $1 < j < n$, is an integer matrix used to represent the relative fitness of individual resources.

A discrepant output from the discrepancy mirror for any pairwise comparison of two configurations will lead to a unit increment in the value of all **H[i,j]** where $\mathbf{U_k[i,j]} = 1$, for the two configurations. Initially $\mathbf{H[i,j]} = 0 \ni i,j$. Over a period of time, the value of **H[i,j]** will provide temporal information regarding the number of configurations that used resource *(i,j)* that also caused a discrepant output to be observed. Lower values of **H[i,j]** denote higher fitness for resource *(i,j)*. Thus, at any point of time, **H[i,j]** indicates the cumulative history of discrepancies articulated by group tests that utilized resource with coordinates *(i,j)*.

4 Fault Isolation By Discrepancy-Enabled Dueling

Fault isolation by discrepancy-enabled dueling utilizes information from the history matrix to accelerate the fault isolation process. The algorithm proceeds by maintaining a history of the discrepancies, and the fitness indices of individuals involved in the competitive dueling instances. An iteration of the isolation process is complete when the outputs of the pair of dueling configurations are evaluated for discrepancies and the history matrix is updated. In order to facilitate faster isolation, the functional configuration of the individuals will be updated, by swapping columns of resources used by the individuals. These can later be used to select configurations to be loaded onto the FPGA for carrying out computations. The fitness measure produced can also be used by the evolutionary repair algorithm at a later stage. The experiment is seeded with a population of competing individual configurations. Each individual uses a particular subset of the available resources. A cell is represented by two coordinates corresponding to the row and column that completely specify the location of the cell on the FPGA.

Assuming a single fault, such as a stuck-at-0 or a stuck-at-1 fault in the device, a subset of the population of competitors will be adversely affected. The effect of the fault will be observed as a discrepancy at the output of the discrepancy detector when an affected individual is paired with an unaffected individual. When a discrepancy is observed, the subset of resources used by the individual are assumed to be *suspect* with regard to faults. The elements in the history matrix corresponding to the coordinates of the resources used by the individual are incremented to note the occurence of the discrepancy. Likewise, when an individual does not exhibit a discrepancy upon evaluation, the elements in the history matrix corresponding to the resources used by the individual are decremented, thereby making these resources less suspect of being faulty. Over a period of time, through a repeated process of successive intersection being subsets of known faulty and fault-free resources, a consensus will emerge regarding the location of the single fault. The fault is isolated when there is a unique element in **H** with the maximum value among all elements. Finding this unique element is the terminating condition for one iteration of the isolation process. If the algorithm fails to converge upon a single element of **H** after a pre-defined number of iterations, then the existence or more than one fault is established, nullifying the single-fault assumption. As a corollary, if the system converges to a state where there are *d* elements with the maximum value in all **H[i,j]**, then *d* faults are said to have been identified. The coordinates of the element with the maximum value in the history matrix provides information regarding the location of the fault.

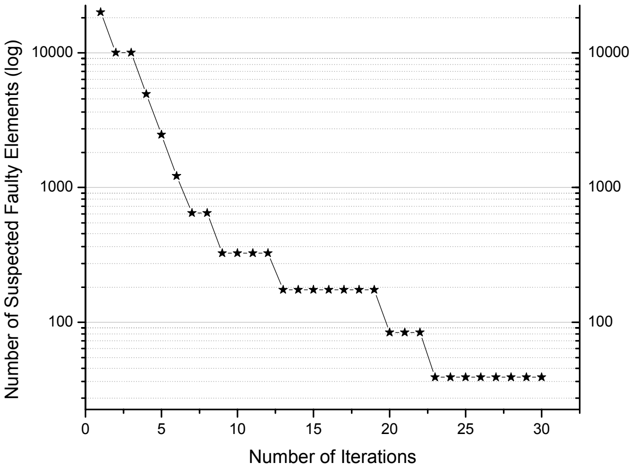

Figure 2: Successive Isolation as Input Iterations Increase

There are certain cases where the simple fault isolation scheme described above may fail to converge on a single faulty resource. A trivial case is when all the resources available on the FPGA are used by each configuration. If the application demands that all the resources be used, then isolation cannot occur through the process of successive intersection. Also, in cases where a very low number of resources are used by individual configurations, it is possible that none of the individuals utilize the faulty resource, leading to the state where no discrepancies will be observed. The most challenging case is when multiple individuals utilize the faulty resource. In this situation, the history matrix elements corresponding to the interection of the subset of resources used by these individuals will have no relative differences, and will all have the highest value. Successive intersections between the resource subsets will not lead to any further fault isolation. For example, with a resource utilization of 40% in a device with 40,000 unit resources, isolation proceeds as shown in Figure 2. The isolation cannot be completed, and after about 23 iterations, the number of suspected faulty elements stays a constant at 36. Any further isolation cannot occur since there is none of the intersections that may follow provide any additional isolation information. This neccessitates an algorithm based on group testing.

5 Dueling with Modified Halving

Combinatorial group testing algorithms relate to the problem of identifying individual defective members from a large population by conducting tests on *sub-groups* or *blocks* of elements. Group testing has been used in medical, chemical and electrical testing, coding, drug screening, pollution control, multiaccess channel management, and recently in data verification, clone library screening and blood testing. Though group testing algorithms are not directly applicable to the problem at hand, some of the principles can be used to facilitate fault isolation. CGT algorithms have also been applied to the problem *Built-In Self Test* (BIST) diagnosis [11], whereby methods such as *digging*, *multi-stage batching*, *doubling* and *jumping* are developed to reduce the number of test within BIST schemes. BIST however is a diagnosis method that relies upon exhaustive and comprehensive testing, often carried out offline.

To avoid the problem of not being able to proceed with isolation in certain cases where successive iterations do not provide isolation information, a dueling algorithm is proposed which tries to emulate *halving*. Halving is the process of successively reducing the size of the subgroup under test by half until, finally a test of a single element is required to identify the faulty element. This method cannot be directly applied to the problem since it is only possible to test groups of resources, and also, the groups of resources have to be of the same size – as specified by the target application's computational needs on the FPGA.

The proposed dueling algorithm works by swapping columns in the configurations of individual elements. When the fault isolation process approaches a state of stasis, some of the columns in the individuals are swapped. The number of columns to be swapped is determined by considering the number of resources currently suspected of being faulty. A number of columns equal to half of the remaining number of suspect elements are swapped with other columns in the same individual. This will introduce new information, as some of the suspected faulty elements used by the individual earlier will no longer be used, for example.

Swapping is restricted only to the columns to facilitate future implementation in FPGA hardware.

Dueling with column swapping always leads to an isolation of the fault, even in the difficult cases where the resource utilization is too high, or too low. As shown in Figure 3, isolation proceeds till a single faulty element is isolated under the same conditions under which the results shown in Figure 2, for dueling without swapping were obtained.

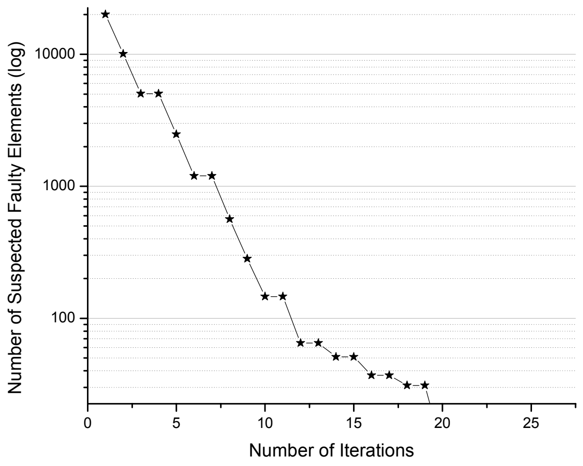

Figure 3: Isolation Progress when Halving is used

6 Fault Isolation using Halving-enabled Dueling

In order to analyze the behavior of the dueling algorithm with modified halving, further experiments were conducted to see the implications of various factors on the isolation process. In each of the following experiments, the population size specifies the number of competing individual configurations in the population. Resource utilization, expressed as a percentage signifies the amount of available resources used by an application implemented on the FPGA. The FPGA device is simulated by using a square matrix of order *n* where *n* denotes the number of rows and columns in the device

6.1 Effect of Number of Elements in S

The effect of the size of the isolation problem was evaluated by applying the proposed technique to simulated FPGAs of various array sizes. As shown in Figure 4, for an isolation problem where there are 100 rows and columns, or 10000 elements, only an average of 14.3 iterations are required to isolate a single fault. As the size of the array containing the fault increases, the increase in the required number of iterations is minimal. For example, for the difficult case where there is a single fault in 1 million resources, the algorithm requires only an average of 27.4 iterations to isolate the fault, showing that the algorithm scales well with the size of problem.

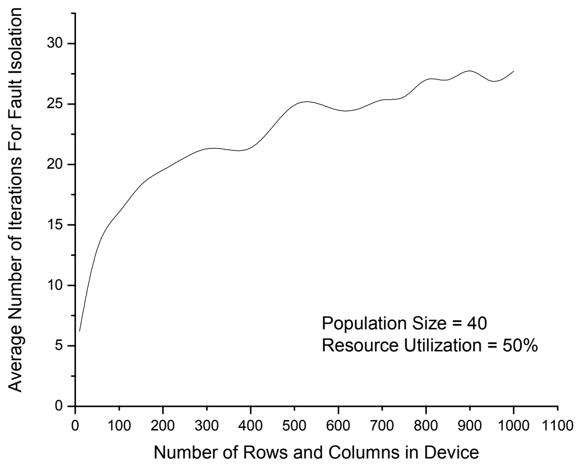

Figure 4: Isolation Performance as a Function of the Total Number of Elements

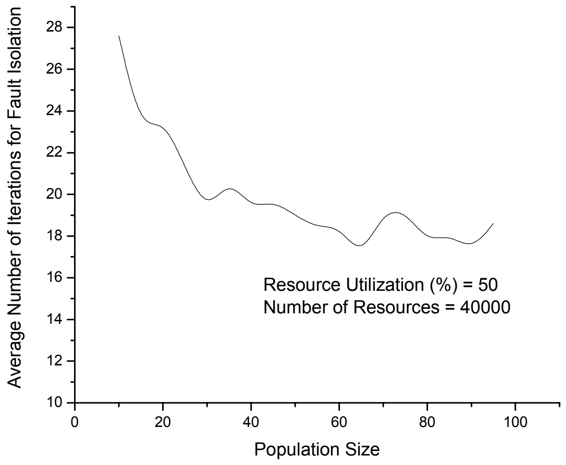

Figure 5: Isolation Performance as a Function of the Population Size

6.2 Effect of Population Size

As the population size increases, fault isolation is expected to become faster, since more information will be available to the algorithm from the increased population size. However, a very high population size may lead to more individuals being affected by the same fault. As shown in Figure 5, the number of iterations required for isolation, with 40000 elements, and 50% resource utilization shows a tendency to decrease with an increase in the population size. For a population of size 60, only an average of 17.2 iterations are required for isolation. Practically, however, a very high population size will imply the need for a higher number of alternative individual configurations. A population size of 30 seems to be an ideal tradeoff between ease of isolation, and the difficulty of generating increased number of individuals.

6.3 Effect of Resource Utilization

The ease with which faults can be isolated also depends on the percentage of available resources utilized by the individuals. If all the available resources are utilized by the individuals, the fault cannot be isolated through the proposed process of repetitive pairing, since all the individuals are equally likely to be affected by the fault. Also, if the utilization is very low, then none of the individuals might be affected by the fault. Except for these extremes, the algorithm always succeeds in isolating the fault. As shown in Figure 6, isolation takes longer when less than 20% or greater than 80% of the available resources are utilized.

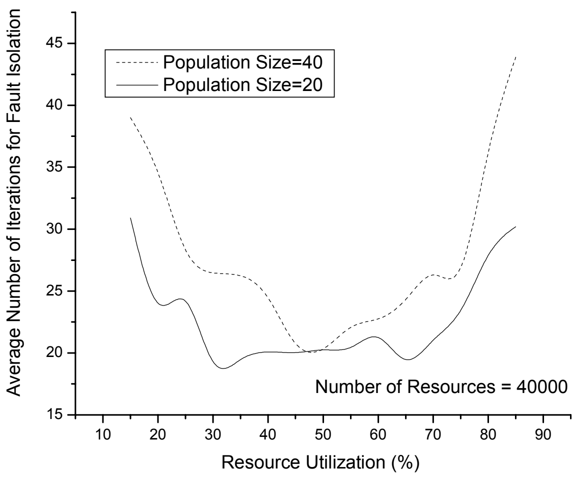

Figure 6: Isolation Performance as a Function of the Resource Utilization

7 Conclusions

A novel intelligent fault isolation method for reconfigurable devices based on combinatorial group testing methods is presented. The dueling algorithm with modified halving consistently isolates the faulty resource requiring as few as 18 iterations on an average to isolate a fault element from among 40000 elements. The use of the algorithm obviates the need for special test inputs, and enables online testing of the device. In conjunction with a discrepancy detector, normal data throughput inputs can be used to isolate faults with a minimal number of iterations. Future work on this topic includes the development of an architecture to enable partial reconfiguration of FPGAs to enable column swapping, and evolutionary algorithms for runtime fault tolerance.

Acknowledgments

This research was supported in part by NASA Intelligent Systems NRA Contract NNA04CL07A.

References

[1] D. Du and F. K. Hwang. *Combinatorial Group Testing and its Applications*, volume 12 of *Series on Applied Mathematics*. World Scientific, 2000.

[2] J. D. Lohn, G. Larchev and R. F. DeMara. "A Genetic Representation for Evolutionary Fault Recovery in Virtex FPGAs," *Proceedings of the Fifth International Conference on Evolvable Systems (ICES '03),* March 2003.

[3] N.A.Touba and E. J. McCluskey. "Logic Synthesis of Multilevel Circuits with Concurrent Error Detection," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 16(7), pp. 783-789, July 1997.

[4] K. Mohanram, E.S. Sogomonyan, M. Gossel and N.A. Touba. "Synthesis of low-cost parity-based partially self-checking circuits," in *Proceedings of On-Line Testing Symposium (IOLTS '03),* pp. 35-40, July 2003.

[5] S. Mitra and E. J. McCluskey, "Which Concurrent Error Detection Scheme to Choose?," in *Proceedings of the International Test Conference 2000,* p. 985, October 2000

[6] M. Garvie and A. Thompson. "Scrubbing away transients and Jiggling around the permanent: Long survival of FPGA systems through evolutionary self-repair," *Proceedings of 10th IEEE International On-Line Testing Symposium,* pp. 155-160. IEEE Computer Society, 2004.

[7] D. P. Siewiorek and R. S. Swarz. *Reliable Computer Systems: Design and Evaluation.* Digital Press, 1992.

[8] S. Vigander. "*Evolutionary fault repair of electronics in space applications.*" Masters thesis, Norwegian University of Science and Technology, Trondheim, 2001.

[9] R. F. DeMara and C. A. Sharma. "Self-Checking Fault Detection using Discrepancy Mirrors," in *Proceedings of 2005 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '05),* June 2005.

[10] R. F. DeMara and K. Zhang, "Autonomous FPGA Fault Handling through Competitive Runtime Reconfiguration," *Proceedings of NASA/DoD Conference on Evolvable Hardware(EH'05),* Washington D.C., U.S.A., June 29 – July 1, 2005.

[11] A. B. Kahng and S. Reda. "Combinatorial Group Testing Methods for the BIST Diagnosis Problem," in *Proceedings of the Asia and South Pacific Design Automation Conference,* January 2004.