Consensus-based Evaluation for Fault Isolation and On-line Evolutionary Regeneration

Kening Zhang, Ronald F. DeMara, Carthik A. Sharma

Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
demara@mail.ucf.edu

Abstract

While the fault repair capability of Evolvable Hardware (EH) approaches have been previously demonstrated, further improvements to fault handling capability can be achieved by exploiting population diversity during all phases of the fault handling process. A new paradigm for online EH regeneration using Genetic Algorithms (GAs) called ***Consensus Based Evaluation*** *(CBE) is developed where the performance of individuals is assessed based on broad consensus of the population instead of a conventional fitness function. Adoption of CBE enables information contained in the population to not only enrich the evolutionary process, but also support fault detection and isolation. On-line regeneration of functionality is achieved without additional test vectors by using the results of competitions between individuals in the population. Relative fitness measures support adaptation of the fitness evaluation procedure to support graceful degredation even in the presence of unpredictable changes in the operational environment, inputs, or the FPGA application. Application of CBE to FPGA-based multipliers demonstrates 100% isolation of randomly injected stuck-at faults and evolution of a complete regeneration within 135 repair iterations while precluding the propagation of any discrepant output. The throughput of the system is maintained at 85.35% throughout the repair process.*

1 Introduction

Evolutionary mechanisms can actively restore mission-critical functionality in SRAM-based reprogrammable devices such as *Field Programmable Gate Arrays* (FPGAs). They provide an alternative to device redundancy for dealing with permanent degradation due to radiation-induced stuck-at-faults, thermal fatigue, oxide breakdown, electromigration, and other local permanent damage without the increased weight and size normally associated with spares. Hence, recent research has focused on employing the capability for reconfiguration inherent in field programmable devices to increase reliability and autonomy [1], [2], [3], [4], [5]. In these experiments, fault-tolerance is evolved at design time, or achieved at repair-time using evolution after taking a detected failed unit offline. In both cases, GAs provided a population-based optimization algorithm with the objective of producing a single best-fit individual as the final product. They rely on a pre-determined static fitness function that does not consider an individual's utility relative to the rest of the population. The evaluation mechanisms used in previous approaches depend on the application of exhaustive test

vectors to determine the individual with the best response to all possible inputs. However, given that partially complete repairs are often the best attainable [4], [2], other individuals may outperform the best-fit individual over the range of inputs of interest. In particular, there is no guarantee that the individual with the best absolute fitness measure for an exhaustive set of test inputs will correspond to the individual within the population that has the best performance among individuals under the subset of inputs actually applied. Thus, exhaustive evaluation of regenerated alternatives is computationally expensive, yet not necessarily indicative of the optimal performing individual among a population of partially correct repairs. Hence, two innovations are developed herein for self-adaptive EH regeneration:

1) Elimination of additional test vectors, and
2) Temporal Assessment based on aging and outlier identification

In *Consensus-based Evaluation (CBE)*, 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 runtime, these individuals compete for selection based on discrepancy favoring fault-free behavior. Discrepant behaviour, where the outputs of two competing individuals do not agree on a bit-by-bit basis, is used as the basis for the performance evaluation process. Any operationally visible fault will decrease the fitness of just those configurations that use it. Over a period of time, as the result of successive comparisons, a consensus emerges from the population regarding the relative fitness of all individuals. This allows the classification of configurations into ranges of relative reliabilities based on their observed performance during online operation.

2 Related Work

Adaptive regeneration has been investigated as an alternative to using pre-determined spares. Most researchers [2], [3], [5], [6] focus on using traditional GAs to identify a single best-fit individual at the termination of the evolutionary computation. Keymeulen, Stoica, and Zebulum [1] use a design-time emphasis to improve fault tolerance. They develop evolutionary techniques so that a circuit is initially designed to remain functional even in presence of various faults. Their *population-based* fault tolerant design method evolves diverse circuits and then selects the most fault-insensitive individual. In this paper we propose a system that achieves improved fault tolerance by using a runtime adaptive algorithm that emphasizes the utilization of responses observed during the actual operation of the device. While their population-based fault tolerance approach provides passive run-time tolerance, CBE is dynamic and actively improves the fault tolerance of the system according to environmental demands.

Yao and Liu [7] emphasize that in evolutionary systems, the population contains more information than any one individual. They develop two examples to demonstrate the use of the information contained in the population in the domains of artificial neural networks and rule based systems respectively. The last population is used efficiently and out-performs the single best-fit individual in these two examples. [8] presents four methods for combining the different individuals in the final population to generate the solution. They provide results for three data sets, namely the Australian credit card assessment problem, the heart disease problem and the diabetes problem, which show that solutions obtained by combining individuals outperform any single individual. While the authors devise a method to utilize the information contained in the population to improve the final solution, they fail to use the information in the population to improve the learning and optimization process

itself. Also, the authors emphasize that learning systems are different from optimization problems, and that information contained in the population is only useful in learning systems. The proposed approach clearly indicates that even optimization and repair problems can benefit from population information. More recently, in [9] the authors describe using fitness sharing and negative correlation to create a diverse population of solutions. A combined solution is then obtained using a gating algorithm that ensures the best response to the observed stimuli. In EHW, it may not always be possible to combine solutions without additional physical resources that may be fault-prone. In our approach, all individuals in the population are recognized as possible solutions, with the best emerging candidate being selected based on their runtime response and performance track record. The authors also claim that applying the described techniques to EHW should be a straightforward matter, but do not describe any applications or examples. They state the absence of an optimal way of predicting the future performance of evolved circuits in unseen environments. We show that it is possible for an adaptive system to keep track of the relative performances of individuals and implicitly build a consensus.

Layzell and Thompson [10] identify Populational Fault Tolerance (PFT) as an inherent quality of EHW. They state that due to the incremental nature of evolutionary algorithms, the solution changes along the course of evolution to adapt to faults. The evolutionary history of the evolved circuit was used to arrive at the conclusion that PFT is an inherent quality in evolutionary design due to the incremental incorporation of additional components into a prototype depending on conditions. They speculate that PFT is less likely to occur for online evolution in varying environments. An evolutionary process that uses absolute fitness measures and exhaustive tests may not be able to provide adaptive fault tolerance.

Previous research has not focused on leveraging the robustness of a population to improve the detection and isolation phases, or to achieve an online evolution process. Problems related to fault tolerance in online evolution identified by the existing approaches are addressed by the new Consensus-based Evaluation scheme. Online evolution defines an essentially different problem from a traditional GA optimization problem. To address the problem effectively, a new fitness evaluation paradigm is required. With relative fitness measures based on competition, a running consensus is produced regarding the fitness of individuals in repsonse to the actual environmental stimuli. This can be used by the regeneration process to adapt to runtime requirements and improve the fault tolerance of the population. The CBE approach presents a new online adaptive repair mechanism that fully exploits the advantages of population-based evolutionary methods. It utilizes a *temporal voting* approach whereby the outputs of two competing instances are compared at any instant and alternative pairings are considered over time. The presence or absence of a discrepancy is used to adjust the *discrepancy values* (DVs) 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 through other pairings of competing configurations. The competitive process is applied repeatedly to form a strong consensus across the diverse pool of alternatives. The fitness of individuals is determined through this continuing runtime process by evaluating the real time performance of individuals in comparison to others in the population. Instead of using an absolute fitness function, with the concomitant exhaustive testing, relative discrepancy values are used as the threshold to identify faulty individuals. Also, the system actively selects individuals that perform the best, given the current environment. Healthy individuals are used to achieve the repair of individuals affected by faults. The proposed approach makes full use of the fact that repair complexity is far less than design complexity. CBE achieves improved fault tolerance by making extensive use of the information contained in the population – both as raw material for creating new individuals, and as information that enables faster and more accurate fault isolation. Any improvement in the fault isolation process

speeds up the regeneration process by directing the GA search in the proper direction. The use of a relative fitness measure and temporal consensus improves the fault tolerance and adaptability of the population.

3 Autonomous Regeneration using CBE

A GA performs a multi-directional search by maintaining a population of potential solutions and encouraging information formation and exchange along these directions. By encouraging direct competition between individuals in the population, a relative fitness measure based on consensus can be generated. The objective fitness function used in traditional GAs can be effectively replaced by the emergent consensus and relative fitness measure. The relative fitness measure is inherently dynamic, and by using an *Evaluation Window* for the individuals, an accurate reflection of the environmental conditions and changes can be achieved. Multiple potential directions for future exploration can be created and utilized depending on the conditions prevalent during the evolutionary process.

In the CBE approach, an initial population of *Pristine* individuals is created by manual design. These primordial configurations are functionally-identical (same input-output behavior), yet they utilize physically-distinct resources (alternative design or place-and-route implementations). For puposes of illustration, assume two competing *half-configurations* labeled Functional Logic Left ("*L*") and Functional Logic Right ("*R*") are loaded in tandem on the physical FPGA platform. The half-configurations occupy mutually exclusive physical resources to implement identical functionality. This realizes a conventional Concurrent Error Detection (CED) arrangement to identify at least any single resource fault with certainty [11]. As in traditional CED approaches, comparison of the outputs of the two resident half-configurations will produce either *discrepant* or *matching* outputs which will indicate the presence or absence of faulty resources in the FPGA hardware platform respectively.

Under CBE, whenever two half-configurations disagree, the *Discrepancy Value (DV)* of both half-configurations are incremented. By repeated pairing over a period of time, only those half-configurations which do not use faulty resources will eventually become preferred. This is because the DV of a faulty half-configuration is always increased regardless of its pairing, yet the DV of fault-free half-configurations which are paired together do not increase. This process occurs as part of the normal processing throughput of the FPGA without additional test vectors or other diagnostic routines. The determination of a configuration's health state is based on its cumulative DV relative to DV of the other individuals in the population evaluated over a period called the *Evaluation Window*, denoted by E^W.

3.1 CBE Procedure

The procedure begins with pre-designed individuals that are fault-free. These individuals are divided into two groups, *L* and *R*, where each group of individuals uses mutually exclusive physical resources. This is essential to ensure that one individual each form both groups can reside and compete in tandem on the FPGA. In addition, every individual can belong to one of four states – *Pristine, Suspect, Under-repair* or *Refurbished.* In the beginning, all individual are pristine. At any given point of time, one individual each from the *L* and *R* groups compete with each other. State transistions occur according to the result of pairwise output comparison. A comparison can lead to

two results - "$L=R$" and "$L\neq R$" indicating whether the two resident half-configurations produce either *matching* or *discrepant* outputs, respectively. When $L=R$ occurs then both individuals retain their *Pristine* state. However when their outputs disagree then both the configurations are demoted to the *Suspect* pool and the DV of both individuals is increased. Whenever such a transition occurs, a *Fault Alert* indicator is issued because two functionally-identical circuits disagree. Hence at least one resource fault must have occurred.

More formally, the *i*-th half configuration remains in the *Suspect* pool until its DV f_i evaluated over the preceding E^W pairings rises above the *Repair Discrepancy Value* ($f_i < DV_R$) which is defined as average DV of entire population accumulated over E^W. The *i*-th half-configuration is then marked as *Under Repair* until its DV drops below the *Operational Discrepancy Value* ($f_i \geq DV_O$) which is defined as average DV of the healthy individuals among the population (*Pristine, Suspect and Refurbished*) accumulated over E^W. Under the fault-free circumstance, $DV_O = DV_R$ until the faulty individuals appear in the population as a result of emergent hardware faults. Thereafter, f_{OT} is modified such that $DV_O \leq DV_R$ which provides dithering immunity such that the configuration is indeed *Refurbished.*

Over a period of time the DV of an individual could increase further and *complete regeneration* becomes possible though not necessarily externally distinguishable from *partial regeneration*. Competing half-configurations remain *Refurbished* unless their DV rises above the Repair DV, at which time they again demoted to the *Under Repair* state.

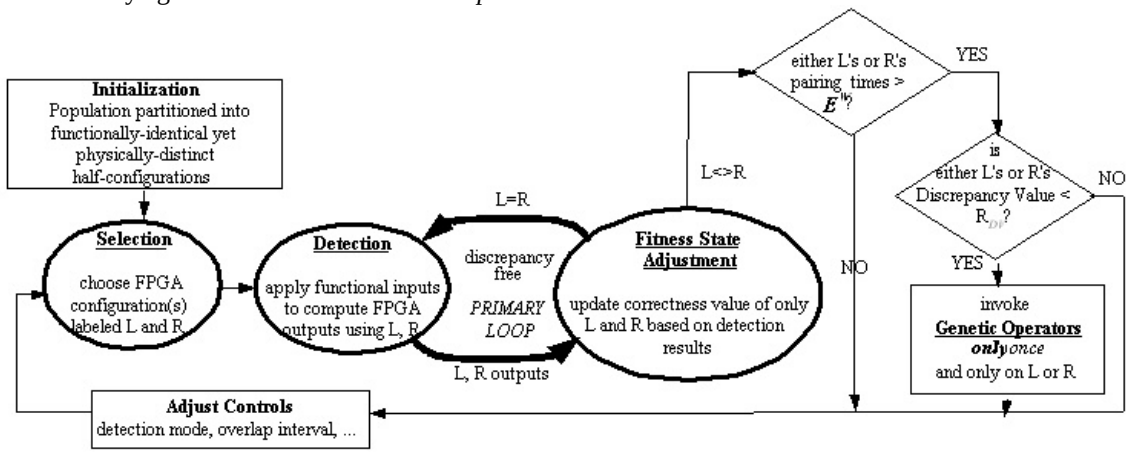

Fig. 1. Procedural Flow in the CBE Technique

The procedural flow of the CBE algorithm that calculates the health state transitions is depicted in Figure 1. After initialization, *Selection* of the *L* and *R* half-configurations occurs which are then loaded into the FPGA. The *Detection* process is conducted when the normal data processing inputs are applied to the FPGA. Based on agreement or disagreement among the outputs of the two competing *L* and *R* half-configurations, *Discrepancy Value Adjustment* for both individuals occurs. The central *PRIMARY LOOP* representing *discrepancy-free* behavior can repeat indefinitely without any reconfiguration of the FPGA. Only when outputs disagree do alternate configurations need to be loaded. For *Under Repair* individuals, if $f_i > DV_R$ then Genetic Operators are invoked only once on the resident configurations. The modified configuration is then immediately returned to the pool of competing configurations and the Selection step is resumed under normal FPGA throughput processing operations.

3.2 Selection and Detection Process

The *Selection* and *Detection* processes are shown in Figure 2. The usual flow is for *Pristine*, *Suspect*, and then *Refurbished* individuals to be preferred in that order for one half-configuration. On the other hand, the other half-configuration is selected based on a stochastic process determined by the *Re-introduction Rate* (λ_R). In particular, *Under Repair* individuals are selected as one of the competing half-configurations on average at a rate equal to λ_R. Henceforth, this now genetically-modified configuration will be *re-introduced* into the operational throughput flow as a new competitor to potentially exhibit fault-free behavior against the larger pool of configurations not currently undergoing repair.

An additional innovation is that λ_R is not only a continuous variable, but can be adapted under autonomous control. In particular, we strive for Mean-Time-To-Repair (*MTTR)* < Mean-Time-Between-Failures *(MTBF)* by monitoring the ratio of the number of computations elapsed between and adjusting λ_R accordingly.

The Detection process is presented in the lower right corner of Figure 2. If a discrepancy is observed as a result of output comparison, the FPGA is reconfigured with a different pair of competing configurations and the output of the device is temporarily held to be recalculated by the newly selected *L* and *R* half-configurations. These repeated computations and comparisons imply no additional cost since the device remains online and operational and the normal data throughput continues uninterrupted.

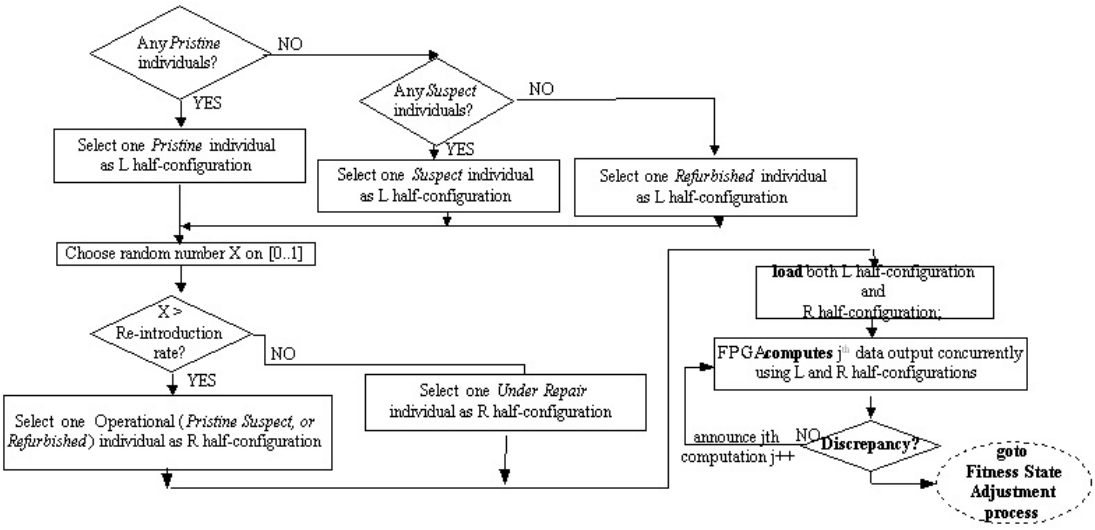

Fig. 2. Selection and Detection in the CBE Technique

3.3 Fitness State Adjustment Process

Figure 3 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 the *Repair Discrepancy Value* DV_R and *Operational Discrepancy Value* DV_O to determine whether the individuals move from one fitness state to another. Ideally, the repair and operational discrepancy values are computed over E^W comparisons for the population. As soon as the all the individuals in the population have completed at least E^W comparisons, new values of these

thresholds are obtained. Since it may be impractical to wait for all individuals to complete the requisite iterations, an individual can undergo a state transition after it finishes E^W iterations. A *Sliding window* is defined, which reduces the latency involved in updating DV_R and DV_O by considering a subset of individuals instead of the whole population. With a sliding window, the values of these thresholds are updated upon the completion of the requisite number of iterations by the number of individuals defined by the sliding window. For *Under Repair* individuals, GA operators are invoked once every E^W iterations.

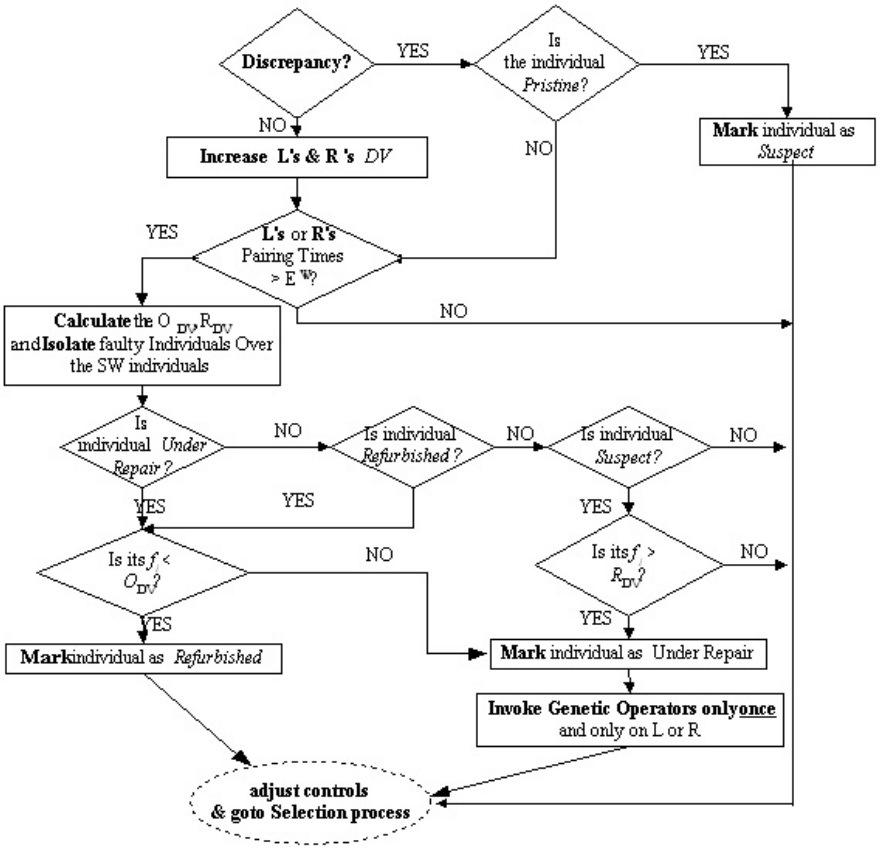

Fig. 3. Fitness State Adjustment Processes in the CBE Technique

4 Evolutionary Fault Repair Circuit

The hypothetical FPGA structure used in the CBE approach is the same as that in Miller, Thomson[12]. 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 are indexed 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 labelling and connection restrictions impose a feed-forward structure on the combinational circuit.

A 3×3 Multiplier is implemented using the above FPGA structure. XOR gates are purposely excluded from the initial designs which leads to designs with a higher number of the gates than conventional 3×3 Multiplier designs to increase the design space. The entire configuration needs 21 CLBs. The population of competing alternatives is then divided into two groups, *L* and *R*, where each group uses an exclusive set of physical resources. For crossover to occur such that offspring are guaranteed to utilize only mutually-exclusive physical resources with other resident half-configurations, a two-point crossover operation is carried out with another randomly selected *Pristine, Suspect* or *Refurbished* individual belonging to the same group. By enforcing speciation breeding occurs exclusively in *L* or *R*, and non-interfering resource use is maintained. The random crossover points are chosen along the boundary of CLBs so that intra-CLB crossover is not possible. The mutation operator randomly changes the LUT's functionality or reconnects one input of the LUT to a new randomly selected output inside the CLB.

5 Experimental Results

An initial population of 20 fault-free configurations was partitioned into mutually exclusive sub-populations *L* and *R*, each containing 10 configurations. Varying stuck-at faults were injected into the architecture that represents permanent physical faults. Several fault isolation and regeneration experiments were carried out using a software simulator. The E^W used in the experiment is 600, which can statistically guarantee that all of 64 input combinations appear at the inputs at least once with probablity of 99.5%, when input combinations are selected at random. For the 3×3 multiplier, the total possible number of input combinations is 2^6=64. Thus $n = 64$ represents the total number of unique input combinations to the simulated FPGA. In the simulation, $m\ (0 \leq m \leq 64)$ is defined as the number of input combinations for which a fault is manifested at the output of the simulated circuit. The number of input combinations for which the output does not match the desired value measures the impact of a fault on an individual. Fault isolation characteristics are analyzed first without considering the regeneration process

The second set of regenerative experiments investigates the regeneration of functionality using CBE. The GA uses a two-point crossover, with a crossover rate of 0.05 and the mutation rate is 0.8. The re-introduction rate is 10%. With the simulated FPGA remaining partially online, all of the regeneration experiments achieved full fault recovery within a few hundred repair operations with normal functional data input. During the regeneration period, data throughput is average 87.94. That is, only 13.16% of the total computations had to be recalculated in order to preclude propagation of discrepant outputs.

5.1 Fault Isolation Experiments

Pairs of individuals, one each from the *L* and the *R* groups are loaded on the FPGA in a repetitive random process. The outputs are compared to check for discrepancies. Judgment on the fault

characteristics of an individual is reserved till it completes E^W pairings, and an *Observation Interval* is complete. A *Sliding Window* of evaluation is defined as five E^W, after which one observation interval is complete and individuals who have completed an E^W are evaluated to identify outliers. The DV of a faulty configuration will increase each time it is compared to another individual. A fault-free individual will see increases in its DV only when it is compared to a faulty individual. Individuals with a DV that exceeds the observed arithmetic mean by one standard deviation are identified as faulty. For example, if 1-out-of-64 outputs are affected in one *L* individual due to a fault, the expected DV of this individual after E^W pairings is DV_L= 1/64* E^W =9.375, assuming equal likelihoods for inputs. A faulty individual can be expected to be identified once every two observation intervals, since the width of each observation interval is defined by 5*E^W. The average *DV* of the R individuals that this is paired with be DV_R =1/64* E^W /10=0.9375, assuming equal selection likelihoods.

Two metrics *Operational DV* (DV_O) and *Repair DV* (DV_R) are calculated and used in the CBA evaluation. DV_O is defined as arithmetic mean of the observed DV of all healthy individuals over a sliding winodw and the DV_R is defined as arithmetic mean of the DV of all individuals considered in the sliding window, including any that may be faulty. If no faulty individuals have been detected, DV_O will equal DV_R, otherwise the $DV_O < DV_R$ as the faulty individuals substantially increase the mean DV. DV_O *and* DV_R are subsequently used in the CBE fault repair mechanism to define the state transitions of individuals. If an individual has a DV < DV_O, it is probably fault-free and can be used for fault-free computation. If the DV of an individual exceeds DV_R, then the individual is placed in the *Under-Repair* group.

In the first experiment, only one individual is affected by a failure in the physical resource, which causes a 1-out-of-64 fault in the individual. Before the fault occurs, the system operates with a 100% throughput, and all individuals have a DV equal to zero. As shown in Figure 4, the fault occurs at time t = 0 and the faulty individual is repeatedly detected and identified at various observation intervals. $DV_O = DV_R$ whenever no faulty individual have been detected over a sliding window. The faulty individual is always detected, but since it has not completed E^W pairing, judgment is reserved, as shown in the plot. When a faulty individual is isolated, the DV_O will be less than DV_R and the faulty DV will be located outside of the DV_R+DV_σ, where DV_σ represents the standard deviation of the discrepancy values.

Figure 5 shows that the isolated individual's DV deviates by 1σ or more, typically 3σ. This shows error-free isolation and that faults are never incorrectly identified. Also, 100% of the faulty individuals are identified within statistically acceptable values for their discrepancies.

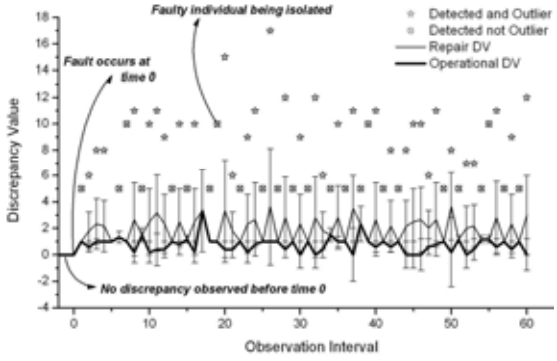

Fig. 4. Isolation of a single faulty *L* individual with a 1-out-of-64 fault impact

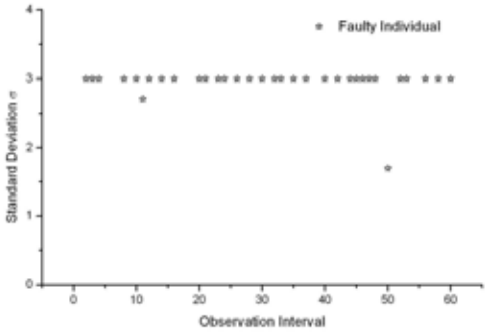

Fig. 5. Plot of Standard Deviations of DV with a 1-out-of-64 fault impact

The average DV of individuals will increase proportionately with fault impact. This leads to increased isolation latency, as shown in Figure 6, for the second experiment, where the characteristics of isolating a single faulty individual with a *10-out-of-64* fault impact are shown. Since there are more faults, the faulty individual is expected to show a discrepancy (10/64)*600 = 93.75 times over its evaluation window. To complete these iterations, it will therefore require (93.75/5) = 18.75 observation intervals, as opposed to 1.88 previously, which leads to both increased discrepancy values for the isolated individuals and an increased time between successive isolations as compared to Figure 4. The detection latency remains unaffected. Figure 7 shows that for a single faulty L individual, with a *10-out-of-64* fault impact, isolation always succeeds when expected.

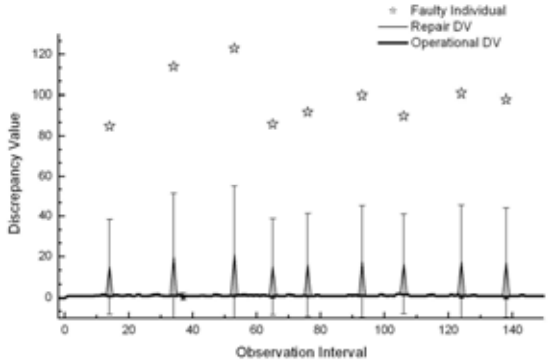

Fig. 6. Isolation of a single faulty *L* individual with a 10-out-of-64 fault impact

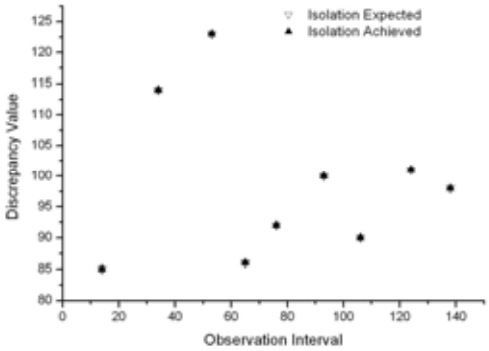

Fig. 7. Performance with a single faulty individual with 10-out-of-64 fault impact

However, when more than one individual is affected by a resource fault, isolation is more time-consuming and difficult as shown in Figure 8, which depicts the isolation characteristics when 4 *L* and 4 *R* individuals are affected by *1-out-of-64* faults. Expected isolations do not occur approximately 40% of the time, as the average discrepancy value of the population is higher, making outlier isolation difficult. The faulty individuals are always detected, but the higher number of discrepancies prevents them from completing E^W iterations within an observation interval. However, a fault-free individual is never incorrectly identified as being faulty.

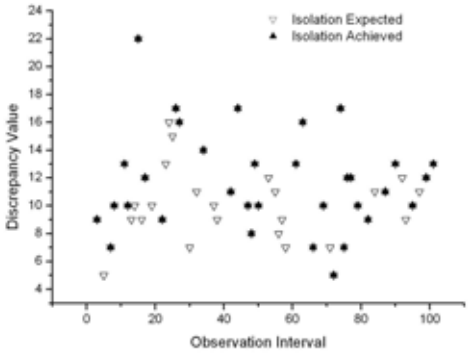

Fig. 8. Isolation of 8 faulty individuals, 4 *L* and 4 *R*, each with a *1-out-of-64* fault impact

5.2 Regeneration of Functionality

CBE-based regeneration experiments were performed on a simulated FPGA platform for the 3x3 multiplier application. Starting with an initial population of 20 viable configurations, random stuck-at faults were injected randomly into one of the 21 CLBs that were utilized to implement the multiplier. The fault reduced the number of correct outputs from 64-out-of-64 to 54-out-of-64. Regeneration was performed using a fitness-state adjustment process that utilized the results of the isolation process described in previous sections. A re-introduction rate of 10% was selected for selecting individuals under repair for performance evaluation. Higher re-introduction rates would lower the throughput whereas if the re-introduction rate is too low, the repair process will be unduly slowed down due to the decreased opportunities to evaluate the performance of the individuals under repair. A low crossover rate of 0.05 was used to ensure that the diversity in the population is preserved. The initial seeding population consists entirely of diverse hand-designed individuals. The mutation rate of 0.8 is required to ensure that the algorithm can explore alternatives by changing the logical functionality of LUTs and the interconnections between them.

While the simulated FPGA remained partially online, regeneration improved correctness to *64-out-of-64* possible outputs. Including iterations that produced functional outputs, the process concluded after a total of 218076 iterations. Complete repair was achieved after only 135 repair iterations when starting with a highly diverse initial population. The fault-affected individual was loaded on the FPGA for a total of 31636 iterations. During the regeneration period, data throughput was 85.54%. Hence, only 14.46% of the total computations needed to be redundant in order to preclude propagation of any discrepant outputs, even when candidate repairs were being re-introduced to refurbish the impacted FPGA configuration without additional test vectors. The throughput will be significantly higher when the system starts from a fault-free situation, since a large number of the initial iterations before the occurrence of the fault will contribute to improving the throughput. Fault isolation using consensus-based evaluation improved the performance of the repair process eliminating the use of an absolute fitness function. The diversity of the initial population provides for increased fault tolerance and also the raw material for realizing the repair.

6 Conclusion

Online EH regeneration essentially defines a problem that is different from offline EH design. CBE leverages the fact that a failed system's *Repair Complexity* can often be much more computationally tractable than either its original *Design Complexity* or its *Re-Design Complexity*, both of which operate in the absence of a diverse population of previously completely correct alternatives. In particular, "repair" implies working design(s) being available before the occurrence of a resource failure. A population of working designs can thus facilitate repair by providing diverse alternates. 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. 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.

Acknowledgments

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

References

1. D. Keymeulen, A. Stoica, and R. Zebulum, "Fault-Tolerant Evolvable Hardware using Field Programmable Transistor Arrays," *IEEE Transactions on Reliability*, Vol.49, No. 3, Sept. 2000
2. S. Vigander, "Evolutionary Fault Repair of Electronics in Space Applications", *Dissertation*, Norwegian University Sci. Tech., Trondheim, Norway, February 28, 2001
3. J. D. Lohn, G. Larchev, and R. F. DeMara, "A Genetic Representation for Evolutionary Fault Recovery in Virtex FPGAs," *Proceedings of the 5th International Conference on Evolvable Systems (ICES)*, Trondheim, Norway, March 17-20, 2003
4. J. D. Lohn, G. Larchev, and R. F. DeMara, "Evolutionary Fault Recovery in a Virtex FPGA Using a Representation That Incorporates Routing," *Proceedings of 17th International Parallel and Distributed Processing Symposium*, Nice, France, April 22-26, 2003
5. M. Garvie and A. Thompson, "Scrubbing away transients and Jiggling around the permanent: Long survival of FPGA systems through evolutionary self-repair," *Proceedings of the 10th IEEE Intl. On-Line Testing Symposium*, pp. 155-160, 2004
6. A. P. Shanthi and Ranjani Parthasarathi, "Exploring FPGA Structures for Evolving Fault Tolerant Hardware", *Proceedings of the 5th NASA / DoD Workshop on Evolvable Hardware*, pp. 184-191, 2003
7. Yao. X, Liu. Y and Darwen. P, "How to make best use of evolutionary learing," In R. Stocker, H. Jelinek, and B. Durnota, editors, Complex Systems: From Local Interactions to Global Phenomena, Amsterdam, pp. 229-242, 1996
8. Yao. X and Liu. Y,"Making use of population information in evolutionary artificial neural networks", IEEE Trans. On Systems, Man and Cybernetics, Part B: Cybernetics, 28(3), pp. 417-425, 1998
9. Yao. X and Liu. Y, "Getting most of evolutionary approaches," In A. Stoica, J. Lohn, R. Kata, D. Keymeulen & R. Zebulum(eds), Proceedings of 2002 NASA/DOD Conference on Evolvable Hardware, IEEE Computer Society, Alexandria, Virginia, pp. 8-14, 15-18 July 2002
10. Layezll. P and Thompson. A., "Understanding the inherent Qualities of Evolved Circuits: Evolutionary History as a Predictor of Fault Tolerance," *Proceedings of Third Conf on Evolvable Systems: From Biology to Hardware (ICES00), Vol. 1801 of LNCS*, pp. 133-142, Springer, April, 2000
11. Subhasish Mitra and Edward J. McCluskey, "Which Concurrent Error Detection Scheme to Choose?" *Proceedings of 2000 International Test Conference*, Atlantic City, NJ, pp. 985-994, Oct. 3-5, 2000
12. Julian F. Miller, Peter Thomson, "Cartesian Genetic Programming", *Proceedings of the Third European Conference on Genetic Programming (EuroGP2000).LNCS*, Vol. 1802, (2000), pp.121-132, Springer-Verlag, 2000