Dynamic Voting Schemes to Enhance Evolutionary Repair In Reconfigurable Logic Devices

Corey J. Milliord, C. A. Sharma, R. F. DeMara
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
milliord, casharma, demara@mail.ucf.edu

Abstract

Success has been demonstrated previously in the use of Genetic Algorithms (GAs) for autonomous fault-handling in Field Programmable Gate Array (FPGA) devices, yet the completeness of a given repair can be improved. This research explores extensions to voting systems to work in parallel with imperfect GA solutions for Local Permanent Damage faults in the FPGA fabric. The benefits are evaluated by comparing performance with a GA that does not utilize voting, for triplex and 5-plex voting arrangements. Results indicate that recovery capability is significantly improved achieving complete repair by a process of synthesis from the incomplete repairs obtained by the GAs. Voting-enhanced systems are shown to increase the likelihood of finding a complete repair, with increases in speed and efficiency. The triplex and 5-plex voting schemes are able to find a complete repair after an average of 113.86 and 48.33 generations respectively in situations where standalone GAs consistently fail.

1. Introduction

Field Programmable Gate Arrays (FPGAs) are reconfigurable logic devices that have become an essential element in critical electronic hardware like those used in space applications such as deep space satellites. Their inherent property of reconfigurability makes them ideal for fault handling research. High levels of radiation and extreme temperatures are examples of operating conditions that induce faults in such devices. If a certain resource being utilized by an FPGA configuration experiences a fault, then the configuration will probably display faulty behavior for a given set of inputs. This fault can be occluded by reconfiguring the FPGA in such a way that the new configuration does not utilize the faulty resource.

GAs can be used to identify suitable alternative configurations for occluding the fault. Starting with an initial population of functionally equivalent, but physically distinct configurations, GAs explore the search space for ideal solutions. Each individual configuration is assigned a specific fitness based on the proportion of number of correct outputs for the given set of inputs of the function being evaluated. In each generation, the individual with the highest fitness is selected as the solution by the GA. The evolutionary process, which utilizes techniques such as crossover and *mutation,* continues till a perfect repair is found, or the pre-determined number of generations is evolved. Refer to [6] for a more detailed description of evolution and GAs, and to [2] for a description of the main concepts in Evolvable Hardware (EHW).

Often, a GA may not be able to synthesize and identify a perfect individual even after a prolonged period. This is where the implementation of a voting scheme is expected to be beneficial. This method involves three or more GAs, rather than just one, and selecting the majority output as the output of the system. For example, in the implementation of a 5-plex voting scheme, five GAs are used concurrently and the outputs of the best fit individual from each GA is compared with that of the others in order to determine the majority output. For this example, three or more equivalent outputs would prevail. Thus five imperfectly repaired configurations can be combined to obtain a more complete repair. For instance, even when none of the five GAs is able to attain a perfectly fit individual as a result of the evolutionary process, a final fitness of 100% is achievable as long as a majority of the configurations do not disagree on any of the outputs. The success of this approach relies on the assumption that the incorrect outputs for each

configuration are random, and that these voting configurations are not producing the same incorrect outputs for the same input combinations. As mentioned in [5], when a set of configurations are designed to implement the same logic function and the resources that are available for design are of a limited quantity, then there a limit on the level of design diversity is to be expected. Provided that the level of design diversity in this work does not hinder the efforts of the GA, the addition of a voting scheme is expected to increase system performance.

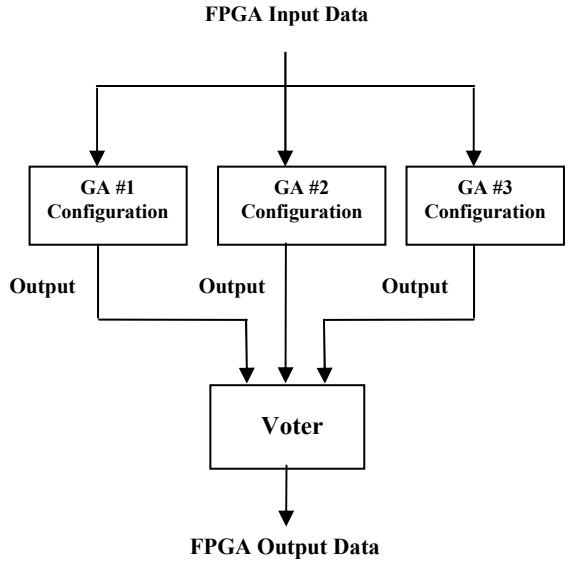

Figure 1: Three-Plex Voting Scheme

2. Related Work

Fault handling techniques that allow the device to remain online and operational are essential in order for the hardware to maintain full functionality throughout the repair process. Although a few offline techniques have shown some promise, such as the one discussed in [4], it is evident that efficiency is increased with a method that uses on-line repair. A good technique should also avoid excessive reliance on spare resources in order to minimize the weight and cost associated with the technique. Another essential characteristic is that the technique should be able to handle multiple faults in the case of critical applications. Finally, it is crucial that the technique can achieve as close to a complete repair, if not a complete repair, as quickly as possible.

The fault-handling method for SRAM-based FPGAs described in [7] is one that is geared towards saving time and minimizing the overheads. The technique uses a re-routing scheme that replaces a faulty CLB with a fault-free spare using replacement wiring segments that can create a connection between the pins of the faulty CLB and the corresponding pins of the replacement CLB. This technique can only repair a single fault in most cases, and is limited by the availability of spares. The fault-repair method is not fully adaptive to accommodate all possible fault scenarios.

In [3], Li, Wang, and Li concentrate on radiation-induced faults in SRAM-based FPGAs. The proposed method uses a TMR (triple modular redundancy) approach which utilizes three separate FPGAs. A majority voter is used to select the output and the entire technique takes place on-line. However, as expected with any TMR approach, there is higher overhead and unnecessary power consumption associated with the degree of redundancy.

The online STARS method proposed in [1] by Abramovici, Emmert, and Stroud describes a technique which utilizes an on-line repair method seeking minimal power consumption. This method uses a roving built-in self-test (BIST) scheme to scan the FPGA chip for faults. When an area of the chip is being tested, its functionality is temporarily relocated until the testing is complete, implying a reliance on spare resources throughout the process. The systematic and cyclical nature of physical testing creates a significant latency issue that increases linearly with the size of the FPGA.

The experimental results that are discussed by Vigander in [6] provide valuable insight regarding the effects of combining a genetic algorithm and a voting system to improve the likelihood of completely repairing a simulated FPGA. Vigander implies that the voting system may not always outperform the best fit individual, due to multiple configurations failing for identical inputs. The primary emphasis of Vigander's work is in utilizing GAs for fault-tolerant design of electronic circuits. In the current work, the applicability of voting-enhanced GAs in EHW repair is analyzed. It is also apparent that the GA could have been optimized prior to the addition of the voting scheme, an issue that is addressed in this paper. It is concluded that further research is necessary in order to provide more meaningful results.

3. Experiment Design

The test bench used for the experimentation is based on a 3x3-bit multiplier, which consists of six inputs and six outputs. The number of possible input combinations is 64, and the corresponding truth table contains 64 rows. With six outputs per row, the total number of output bits in the truth table, for all input combinations, is 384 (64 x 6). However, for faster evaluation, only ten of the input/output combinations

from the truth table were utilized in the following experiments to demonstrate proof of the concept. Hence, in the subsequent experiments, a perfect fitness is defined as 60 (10 x 6). Thus, the overall behavior that is presented in the experiments is representative of larger functions with similar characteristics, without incurring excessive computation costs. The 3x3-bit multiplier problem was chosen for its complexity, to ensure that the GA has a challenging problem that it may not always be able to optimize, working in isolation. This is essential to analyze the incremental benefit of using a voting scheme.

3.1. GA Optimization and Population Seeding

The purpose of the first experiment was to develop an optimal set of GA parameters that maximizes the performance of the GA before a voting scheme is included. The GA parameters were varied independent of the others, throughout a series of analyses. The value of 200 was chosen for the maximum number of generations. The effect of varying mutation rates for several population sizes with a constant crossover rate was studied first, followed by investigations into suitable population sizes, and crossover rates. Optimization of the GA is essential to quantify the increase in performance obtained by including a voting mechanism. In addition to optimizing the GA parameters before voting was included, a set of configurations with perfect fitness values had to be evolved so that a population of perfect individuals could be created for seeding the voting experiments. Here, perfectly fit individuals were created, and stuck-at faults were injected into these perfect configurations to simulate faults in the configurations. The effect of the injected faults was to lower the fitness to a sub-optimal value. This was necessary in order to study the repair process.

3.2. Voting Experiments

After selecting values for the GA parameters according to the results of Experiment One, the implementation of 3-plex and 5-plex voting schemes was tested on the proposed logic function. Several stuck-at faults were injected in each of the runs to simulate the repair process. For both of the voting schemes, graphs were generated to plot the best fit individual versus the number of generations. Each graph consisted of a curve for each of the voting elements, as well as a curve for the whole system in order to highlight the benefit of voting. As an extension, an experiment was also conducted to identify the efficiency of the voting scheme. Here, the performance of one GA, which was allowed to run for three times the number of generations that a voting GA instance was allowed to run was measured. This will allow the comparison of the triplex voting system of GAs, with that of a normal GA that had the same amount of time and computational power available to it.

4. Experimental Results

In the first set of experiments, circuit evolution was performed in an attempt to optimize the GA parameters and to evolve seed configurations with perfect fitness values. In the second phase, the perfectly evolved configurations were modified to simulate stuck-at faults, and then included in the configurations used to seed the GAs. This will enable the measurement of the system performance in a repair application, as opposed to a digital design application. The remaining parameters for the GA were set to the ideal values discovered via earlier experimentation.

4.1. GA Optimization Results

Six sets of empirical experiments were conducted in an effort to obtain the best possible parameter values. The mutation rate in these experiments was varied from .001 to .097 in steps of .004. Each set consisted of twenty-five repetitions for each mutation rate in order to maintain statistical correctness. For these experiments, the number of generations was fixed at 200. Each of the six sets comprised of varying values of population size and number of CLBs. The population size was varied from 15 to 50, and the number of CLBs, which defines the search space available to the GA, was varied from 6 to 36. The GA benefited from a higher population size, and the best performance was observed for a population size of 50, for which the average fitness of the population was the highest observed. The availability of a very high number of CLBs hindered the performance, and the best results were observed when 6-9 CLBs were available for exploration. In addition, the most successful mutations rates were .002, .005, and .093 because they each produced an average fitness of at least 54. The set of ideal parameters obtained as a result of this experiment are shown in Table I.

Table I: Parameter values for evolving perfect individuals

Mutation Rate	Population Size	Number of CLBs
.002	50	9

4.2. Creation of Initial Population

Numerous runs of the GA were completed, with the ideal parameter values, in an effort to evolve a set of five individuals with perfect fitness values. These individuals would serve as the seeds of the populations for the voting experiments. Each experimental run utilized the parameters shown in Table I. Although each run lasted 1000 generations, a vast majority ended without evolving a perfect individual, demonstrating the difficulty involved in evolving a perfect repair. The shortest run needed just 150 generations, while the longest needed 881. The average number of generations necessary to evolve a perfect configuration was 493.6. In order to simulate fault conditions on the FPGA, random faults were introduced in the perfect configurations by injecting stuck-at-one faults at six of the seventy-two LUT inputs. These faults caused the fitness values of the three configurations to decrease to 38, 40, and 47. The GA was then modified so that the population of configurations could be initialized with clones of these individuals. This effectively simulates the condition where a fully operational instance of the system is affected by random faults.

4.3. Triplex Voting Results

Ten triplex voting experiments were run using the parameters shown in Table II. A mutation rate of 0.089 was chosen because it was the only mutation rate that was able to evolve a fitness of 59 in the previous experiments. The triplex experiments involved three separate GA runs whose outputs were voted on after each generation as shown in Figure 1. Each experiment was 400 generations in duration. In seven out of the ten experiments, the triplex voting scheme assisted the GA in arriving at a complete repair, after an average of 113.86 generations.

Table II: Parameter Values for Triplex Voting Experiments

Number of Generations	Mutation Rate	Population Size	Number of CLBs
400	.089	50	9

Table III provides a summary of the triplex voting results, where it can be seen that the system was able to identify a complete repair in as soon as generation 2. The voting performance of a typical repair is shown in Figure 2, which plots the results of trial 7, in which a complete repair is realized by generation 33. The fitness of the individual GAs depicted in the figure show initial success assisted by GA #3, whose fitness quickly increased to 59 in just the 2nd generation. In all of the experiments, no single GA element was able to attain a perfect fitness in isolation. With the addition of the voting scheme, the system is capable of a complete repair exhibiting perfect fitness as soon as after the 33rd generation.

4.4. Efficiency Comparison

To compare the performance of the voting scheme, three additional GA runs were conducted without voting, each lasting 1200 generations with the same parameters as the 3-plex experiments. This ensures that the standalone GA will have the same amount of time, and computational power to arrive at a result. The three runs reached fitness values of 56, 56, and 57, which are well short of perfect. These values were obtained at generations 934, 852, and 274, respectively. It was also observed that for the triplex voting experiments, the voting fitness at Generation 50 was greater than or equivalent to the individual fitness of each of the three GA instances. According to these results, the 3-plex voting scheme is significantly more efficient than a single GA.

Table III: Results from Triplex Voting Experiments

Trial	Earliest Generation of Highest Fitness	GA #1	GA #2	GA #3	Highest Voting Fitness
1	2	56	54	56	56
2	2	54	55	55	56
3	68	57	56	57	58
4	302	55	56	58	60
5	261	58	56	58	60
6	179	57	56	55	60
7	33	56	58	59	60
8	17	58	56	54	60
9	3	56	56	56	60
10	2	55	54	56	60

4.5. Five-Plex Voting Results

Ten experiments were also conducted with a 5-plex voting scheme. Each of these experiments were 300 generations in duration, since repair was comfortably achieved by such time. In this case, a majority vote is determined by a “three-out-of-five” verdict rather than a “two-out-of-three.” Table IV contains the results from the 5-plex experiments. Figure 3 shows the

results of trial 6, in which complete repair was achieved after 34 generations.

The 5-plex experiment result shown in Figure 3 for trial number 6, demonstrates how 100% fitness is achieved even when the highest individual GA fitness is only 56. As shown in the table, nine of the ten runs were able to achieve a perfect fitness. The average number of generations needed to do so was only 48.33. This is advantageous when compared to the average 113.86 generations required in the triplex voting scheme. With more voting elements, the number of common errors among these elements would be lower, leading to faster results.

A voting scheme can have a significant, positive impact on evolutionary repair in FPGAs and similar logic devices. In cases where the evolutionary alogirthm is unable to arrive at a perfect repair, voting-based consensus across the output augments the fitness, and achieves complete repair with far fewer iterations than may be required with a standalone GA. Also, it has been shown that given the same amount of time and computational capacity, voting enhanced GAs outperform single instances. Seventeen of the twenty voting experiments demonstrated the benefits of voting arrangements. With a guaranteed design diversity, where faults do not affect the configurations in the different

4. Conclusions

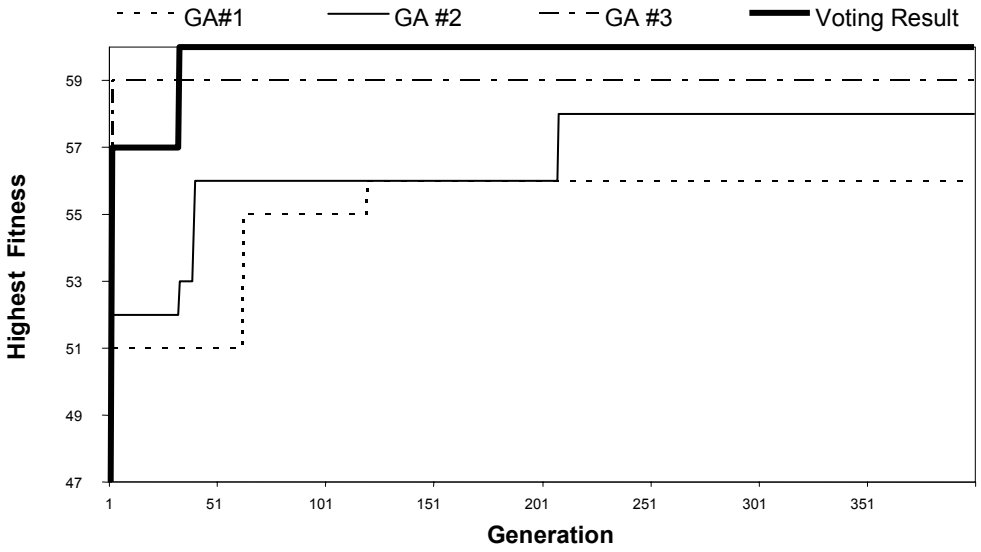

Figure 2: Triplex Arrangement of Separately-Evolved GAs achieves Ideal Fitness at Generation 33

Table IV: Results from Five-Plex Voting Experiments

Trial	Earliest Generation of Highest Fitness	GA #1	GA #2	GA #3	GA #4	GA #5	Highest Voting Fitness
1	68	58	55	55	55	55	59
2	154	56	56	52	58	55	60
3	108	56	54	55	55	53	60
4	55	55	56	57	56	56	60
5	48	55	56	56	57	59	60
6	34	56	58	55	57	55	60
7	27	56	55	57	55	56	60
8	4	56	55	56	56	56	60
9	3	54	55	55	57	56	60
10	2	54	56	54	55	56	60

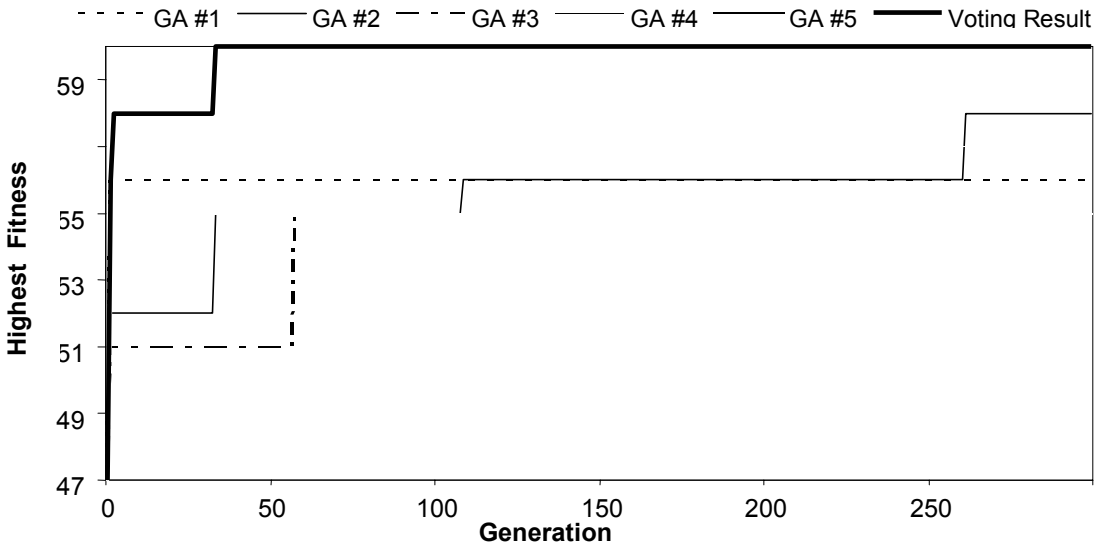

Figure 3: Five-plex Arrangement of Separately-Evolved GAs achieves Ideal Fitness at Generation 34

GA instances in the same manner, complete repair is possible by combining imperfect repairs. The additional space and power needed to implement a voting scheme are justified by the accelerated results and the increased likelihood of identifying a complete repair.

Further research includes similar experiments with a more diverse initial population. Also, as mentioned earlier, only ten of the sixty-four input/output combinations for the 3x3-bit multiplier were tested. If the experiments were repeated with the entire 3x3-bit multiplier truth table, it would probably be more time consuming to obtain a perfect fitness, but it is likely that the characteristics of voting that have surfaced in this research would still be evident. The benefits of utilizing voting-enabled GAs have been clearly identified as increased efficiency, completeness and speed, as compared to utilizing standalone GAs.

References

[1] Abramovici, M., Emmert, J., and Stroud, C., "Roving STARs: an integrated approach to on-line testing, diagnosis, and fault tolerance for FPGAs in adaptive computing systems," Proceedings of The Third NASA DoD Workshop, July 2001, pp. 73-92.

[2] Keymeulen, D., Zebulum, R., Jin, Y., and Stoica, A., "Fault-tolerant evolvable hardware using field-programmable transistor arrays," *IEE Transactions on Reliability*, Vol. 49, No. 3, Sept. 2000, pp. 305-316.

[3] Li, Y., Li, D., and Wang, Z., "A new approach to detect-mitigate-correct radiation-induced faults for SRAM-based FPGAs in aerospace application," Proceedings of the IEE National Aerospace and Electronics Conference, Oct. 2000, pp. 588-594.

[4] Lohn, J., Larchev, G., and DeMara, R., "Evolutionary fault recovery in a Virtex FPGA using a representation that incorporates routing," Proceedings of the Parallel and Distributed Processing Symposium, April 2003.

[5] Mitra, S., Saxena, N., and McCluskey, E., "Efficient design diversity estimation for combinational circuits," *IEE Transactions on Computers*, Vol. 53, No. 11, Nov. 2004, pp. 1483-1492.

[6] Vigander, S., "Evolutionary fault repair of electronics in space applications," Doctorate Dissertation, University of Sussex, Galmer, Brighton, UK, 2001.

[7] Xu, J., Si, P., Huang, W., and Lombardi, F., "A novel fault tolerant approach for SRAM-based FPGAs," Proceedings of the Pacific Rim International Symposium, Dec. 1999, pp. 40-44.