IJE TRANSACTIONS C: Aspects Vol. 28, No. 9, (September 2015) 1276-1285

International Journal of Engineering

Journal Homepage: www.ije.ir

RESEARCH NOTE

A Comparative Study of VHDL Implementation of FT-2D-cGA and FT-3D-cGA on Different Benchmarks

P. Ashooriyan, Y. Baleghi*

Department of Electrical & Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran

P A P E R I N F O

Paper history:
Received 18 September 2014
Received in revised form 30 June 2015
Accepted 30 July 2015

Keywords:
Fault Tolerance
Cellular Genetic Algorithm
FT-2D-cGA
FT-3D-cGA
Processing Element
Single Error Upset

A B S T R A C T

This paper presents the VHDL implementation of fault tolerant cellular genetic algorithm. The goal of paper is to harden the hardware implementation of the cGA against single error upset (SEU), when affecting the fitness registers in the target hardware. The proposed approach consists of two phases; Error monitoring and error recovery. Using innovative connectivity between processing elements and efficient correction policy, the PEs will prohibit spreading the faulty evaluated individual in the population. In the experiments, three metrics and four test functions are used to show the performance of the proposed structures. Two structures (2D and 3D) of proposed FT-cGAs are set to optimize various test functions. The experimental results illustrate the robustness of the proposed system. An outstanding outcome was that the implemented fault tolerant algorithm was able to reach the optimal solution when at least one processing element is healthy in population.

***doi**: 10.5829/idosi.ije.2015.28.09c.04*

1. INTRODUCTION

EAs have been intensively studied in the last decades due to their worthy performance in solving extensive range of problems and especially the NP hard ones. EAs are intrinsically parallel and offer significant advantages in terms of hardware implementation.

While serial EAs are applied, a single population is considered. On the other hand, parallel EAs use structured populations. The parallel EAs can be categorized into cellular and distributed ones [1].

Distributed EAs divide the population into multiple subpopulations (islands). Each island is allocated to a processing element (PE). Processing element (PE) is a hardware element that executes a stream of instructions (in this paper the EA operations are considered). The islands evolve separately and the migration between them is possible.

Instead, in cellular EAs, the PEs are located in an n-dimensional grid that can interact with each other while evolving the assigned population. The interactions are usually performed among the neighboring PEs.

*Corresponding Author's Email: *y.baleghi@nit.ac.ir* (Y. Baleghi)

In this paper, the cellular genetic algorithms (cGAs) are considered. The balance between exploration and exploitation in cGAs is discussed in recent works [2-6].

The growing probability of single event upset (SEUs) in electronic circuits due to the wide reduction in size [2] increased the requirement to novel approaches to create systems that can be robust to this model of fault.

The idea that cellular architecture can bring about fault tolerance is well discussed in [7]. Next, in [8] the idea of applying "parallel GA" in a real world application (GPS attitude determination problem) was presented; however still no fault tolerant feature was reported. In [9, 10], a 3D architecture of cGA was developed that resulted in overall improvements in the performance of the algorithm when compared with smaller grid dimensions. Another reason for using a 3D topology is its flexibility to be implemented with new advanced custom silicon chip technologies [11], while again the fault tolerance was not investigated. By Morales-Reyes et al. [12], the intrinsic capability of cGAs targeting fault tolerance has been considered. The goal of this reference is to approach fault tolerance over the appropriate exploitation of essential parameters and genetic operators in cGAs.

Please cite this article as: P. Ashooriyan, Y. Baleghi, A Comparative Study of VHDL Implementation of FT-2D-cGA and FT-3D-cGA on Different Benchmarks, International Journal of Engineering (IJE), TRANSACTIONS C: Aspects Vol. 28, No. 9, (September 2015) 1276-1285……

TABLE 1. Summary of fault tolerant cellular genetic algorithms

Name	Fault Type	Fault Location	Diagnosed Fault Method	Fault Recovery Method
FT-cGA [12]	SHE	Fitness Register	– – –	Using the appropriate exploitation of inherent parameters and genetic operators specific to cellular GAs
Adaptive-cGA [13]	SEU/SHE	Chromosome Register	The difference average of Hamming distance between two consecutive generations	Changing the square structure to rectangle structure and vice versa
dcGA [14]	SHE	Fitness register	The difference between the average fitness of two consecutive generations	Migrating population from distributed to cellular structure
FT-3D-cGA [6,15]	SEU	Fitness Register	Computing the genotypic entropy	Genetic diversity is used to identify and isolate faulty individuals, and then applies a new migration schema (replace faulty individual with first fault free neighborhood) for modifying error, and then another cGA is run without updating and communicating isolated individuals with their healthy neighborhood.
D-FT-3D-cGA [16]	SEU	Fitness register	Computing the genotypic entropy	Computing genetic diversity and isolate faulty individual and replacement faulty individual with first fault free neighborhood and recalculate the maximum number of generations based on the ratio of fault, and then cGA is run without interfering faulty individual in neighborhoods process

The fault model contains occurrence of SHE at fitness score registers. It was shown that using migration operator and controlled selection intensity, the feature of fault tolerance emerges intrinsically. Morales-Reyes et al. [13], an adaptive method is discussed that is robust to SHEs when incurred at chromosome registers. In this reference, error discovering is performed via monitoring the decreasing genotype diversity of chromosome register that is caused by SHEs error. By Morales-Reyes et al. [14], cGA and dGA are fused to approach adaptive and fault tolerant GA that aims to solve the GPS altitude problem. The proposed dcGA of this reference is resistant to SHEs when affecting the fitness register. Experimental results have exhibited that the fusion of two structures can overtake cGA in faulty situations. Meanwhile, influence of migration policies and adaptive schemes is very important in dcGA structure. Al-Naqi et al. [6, 15], fault tolerant three dimensional design of cGA suitable to implement with recent advanced custom silicon chip technology is proposed.

This structure is fault tolerant and alleviates SEUs that apply in phenotype registers. The experimental result of this reference illustrates that this algorithmic approach is able to persist up to 40% SEUs error. Al-Naqi et al. [16] that is an improved version of previous work [6, 15], the stopping criterion is dynamically assigned and called "Dynamic Fault-Tolerant 3D-cGA". Table 1 gives a brief summary of recent fault tolerant cGAs with different fault models and recovery methods. No hardware implementation or hardware simulation is reported in references of Table 1, while the present paper considers the VHDL implementations of fault tolerant cGAs.

In the present paper two members of fault tolerant n-dimensional cGA family [6] are implemented in VHDL that are tolerant to SEU faults targeting PE's registers that correspond to fitness score registers. The overall results demonstrate the ability of our method to maintain system's functionality despite an increasing number of faults, until one processing elements (PEs) be healthy, and clearly illustrate the importance of migration and connectivity of PEs in our structure.

The remainder of this paper is divided into five sections. Section 2 presents the basic definitions of this paper. The proposed architecture is discussed in section 3. Section 4 presents the simulation setup and results. The results are discussed in section 5. Finally, concluding notes are given in section 6.

2. BASIC DEFINITIONs

This section presents the basic characterizations, failure types and cellular genetic algorithm, considered in this study.

2. 1. Fault Tolerance Single Event Effect (SEEs) have been considered by researchers in recent years. SEEs can be categorized into Single Event Upsets (SEUs) and Single Hardware Errors (SHEs). SEUs occur when a single charged particle changes the state of one or more memory cells inside the device. If only a single memory cell changes state, the SEU is referred to as a single bit upset (SBU) else, if multiple cells change state, the SEU is called a multi-bit upset (MBU) [17]; however, this phenomenon has been recently observed at the ground level [18]. After data rewriting or a system

reset, the functionality of the system is recovered. SHEs are a subclass of SEEs, but these cause a stable alteration to the operation of the system, for example, when one-bit or several-bits of important data at registers or memory are changed due to stable stuck at zero or one logic. Previous works have used several techniques to add fault tolerance ability to systems. These methods are classified into hardware and software techniques or a combination of both. In hardware techniques detecting fault is faster than soft approaches, but it has more overhead, which increases the cost and complexity of systems and, cannot repel all types of random and multiple bit errors. Triple Modular Redundancy (TMR), DICE (Dual Interlocked Storage Cell), redundancy, and check pointing are main members of these techniques. Since the energy from SEE causes functional effects by spreading possibly through all system modules, therefore developing SEE tolerant systems is nowadays supported from a functional rather than a physical perception.

2. 2. Cellular Genetic Algorithm (cGA) genetic algorithms maintain a population of individuals (see Figure 1) that grow according to selection rules and other genetic operators, such as mutation and crossover. For each individual there is a measure of fitness. Selection focuses on high fitness individuals. General heuristics that are provided by mutation and crossover simulate reproduction process. These operators cause some changes in parent individuals in order to generate distinct offspring individuals [19].

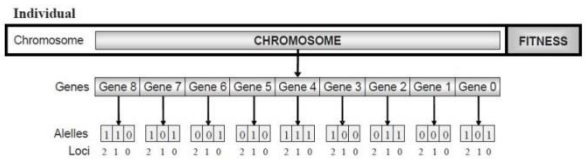


Figure 1. Typical structure of individuals in EAs [20]

Algorithm 1: Pseudo-code of a canonical cGA [20]
1. proc Evolve (cga) // Parameters of the algorithm in 'cga'
2. *Generate Initial Population* (cga.pop);
3. *Evaluation* (cga.pop);
4. While! *Stop Condition* () do
5. for individual ← 1 to cga.pop Size do
6. neighbors ← *Calculate Neighborhood* (cga, *position* (individual));
7. parents ← *Selection* (neighbors);
8. off spring ← *Recombination* (cga.Pc, parents);
9. off spring ← *Mutation* (cga. Pm, off spring);
10. *Evaluation* (off spring);
11. *Replacement* (*position* (individual), auxiliary pop, off spring);
12. end for
13. cga.pop ← auxiliary pop;
14. end while
15. end proc Evolve

Algorithm 1 presents the pseudo-code of a canonical cGA. At the first step, this algorithm generates and evaluates an initial population. After that, the genetic operators such as selection, recombination, mutation and replacement are applied to each individual until the termination criteria is met. The cGA's population can be arranged as an *n*-Dimensional lattice where each individual is allocated to a lattice's position (cell). In this model each cell has internal genetic operators, and just can interact with its local neighbors [20].

3. PROPOSED ARCHITECTURE

The proposed FT-nD-cGA (FT and nD respectively stand for Fault Tolerant and n dimensional) comprises of two main stages. First, the cellular genetic algorithm is started, and during the run applies rules to identify the faulty individuals. In addition, the first stage is carried out until the termination criterion is met. At the end of the first stage, if all individuals are healthy, the algorithm will finish. In contrast, if at least one individual is faulty in population, the second stage of algorithm will start. This algorithm is structured in two blocks as shown in Figure 2.

The Error Monitoring Block in Figure 3 that runs the cGA is the first aforementioned stage. Whenever an error is monitored in this stage, the error monitoring block Enable signal stimulates the second stage that is performed by Error Recovery Block. In the second stage the isolation list, current population and fittest individual are transferred to Error Recovery Block via the corresponding signals. This time the cGA initializes with the current population after healing the faulty individuals by altering them with the fittest individual. Error Monitoring Block (Figure 3) is composed of five sub blocks that are described here. The Error Recovery Block has a very similar structure and contains the same sections.

3. 1. Processing Element (PE) The cellular topology consists of several processing elements (PEs).

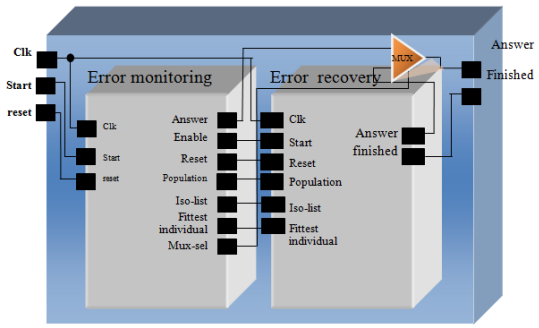


Figure 2. Fault tolerant 2D(/3D) cGA hardware architecture

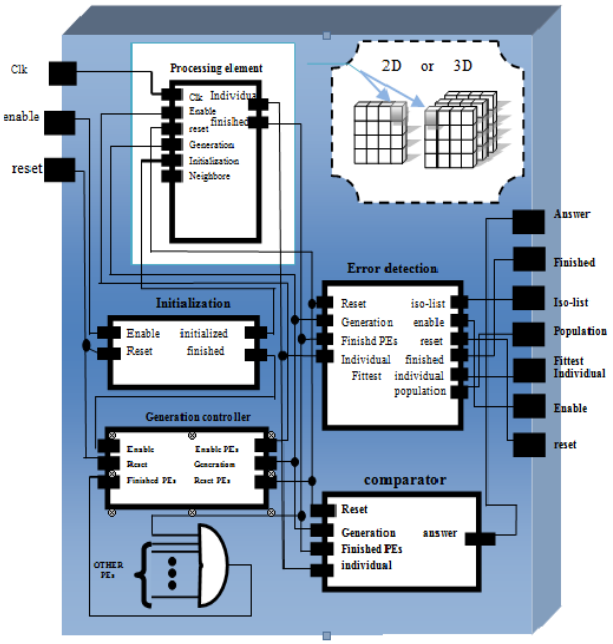
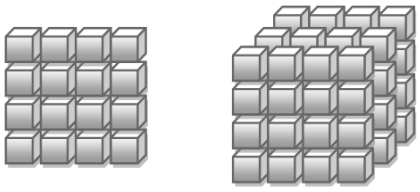


Figure 3. Internal structural error monitoring block

In cGA, the population is arranged as an n-dimensional lattice (see Figure 4) where each network position (cell) is assigned to a PE. PEs operate genetic algorithm separately and have two important registers in their structure that store genotype and the fitness. Each PE has relations within its defined neighbours. This paper considers the 2D and 3D structures. In 3D topology (see Figure 4b) each PE has six neighborhoods -east, west, horizontal south and north, vertical south and north - but in 2D topology (see Figure 4a) each PE has four neighborhoods - North, east, west and south. Algorithm 1 runs in all PEs. First, each PE evaluates the fitness of neighborhoods' genotypes; then, the fittest genotype is selected as the first parent. The second (internal) parent is the current individual (lines 6, 7 of algorithm 1). Then, crossover operator (in this paper one-point cross over) recombines the selected parent to produce 2 offspring's (line 8) and the fittest offspring is selected to be mutated (line 9). The modified offspring is evaluated and the fittest individual between modified offspring and the internal parent stay in this PE for next generation. Since the fault model in this paper is SEU and occurs in fitness register, to prevent the error spread in population, the signal that contains the fitness of individual is not used in PEs' connections. Therefore, before the PEs operation start, fitness values of neighborhood genotype is calculated in each PE.

3. 2. Initialization Component This block generates initial value in first generation for all PEs.



(a) 2D topology **(b)** 3D topology

Figure 4. Cellular 2 and 3 dimensional topologies

In Figure 5, initialization block uses random number to generate initial value for PEs in Error Monitoring.

However, in Error Recovery Block of Figure 2, the initialization block is somehow different. It uses three factors to generate initial value for second part of algorithm as mentioned in Section 3. The additional inputs of this sub-block are depicted in Figure 6.

3. 3. Generation Controller Basically, Generation controller (Figure 7) is a counter block that determines the generation number for all PEs. Also, the maximum generation number is saved in this block. This block is activated when the entire operation of PEs in previous generation in each part is finished.

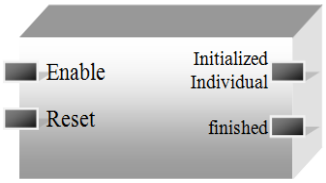


Figure 5. Initialization in Error Monitoring Block

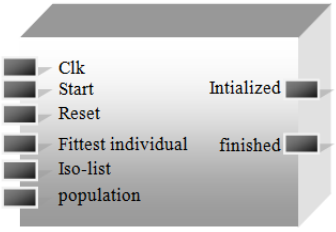


Figure 6. Initialization in Error Recovery Block

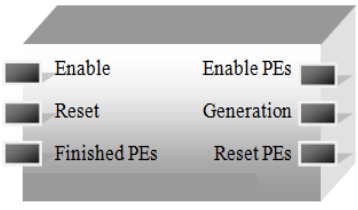


Figure 7. Generation controller component

3. 4. Error Detection In this component (see Figure 8), the faulty PEs of population are identified. Then a list of isolated PEs is created. Next, if all PEs were healthy the algorithm is finished. Otherwise, even if one PE was faulty, the second step of algorithm starts with activation of error recovery block. In addition, this block finds best fault free individual from current generation. The best fault free individual is used to transfer to Error Recovery Block.

Since the fitness value signal is not used in PE bindings and the fitness is re-evaluated in each PE, the SEU errors in fitness score register cannot spread in new populations. Using this replacement policy in Error Recovery Block, the algorithm is very fast to reach the optimal answer. In fact, after repairing the population in recovery block, the initial population resumes with fitter individuals. A PE is recognized faulty: if a specific individual remains unaffected in the processing element during the total generations.

3. 4. Comparator At the end of each generation when the PEs task are finished, this block (Figure 9) selects the best individual of population, and therefore the optimized answer in each generation is chosen .

3. 5. Design Methodology The mission of the proposed structure is to prevent error propagation. The main idea to accomplish this mission is to use the healthy individuals to repair the faulty ones.

The error detection block is responsible for identifying the faulty individuals. The replacement procedure is performed in Initialization component that is located in Error Recovery Block. This component uses the isolation list – that includes the faulty PEs – to replace the faulty PEs with the fittest individual of the last generation of Error Monitoring phase.

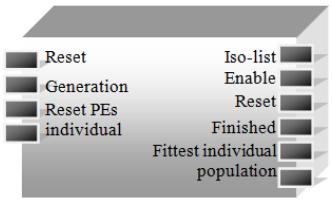


Figure 8. Error detection block

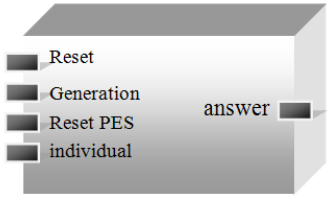


Figure 9. Comparator component

4. SIMULATION AND RESULTS

This section concentrates on considered fault model and benchmark problems that are used to assess the performance of proposed architecture.

4. 1. Fault Model In this paper we consider the SEU error that occurs at fitness score register and assume that this fault model has occurred in first generation. It means that the cellular genetic algorithm starts its operation with a mixed population of faulty and healthy individuals. SEUs happen on one or more bits of registers and flip their values to low or high fitness. The bit(s) flipping in fitness score register keep their fitness value stuck at '1' or '0'. In our approach we consider only the worst case of fault model by forcing the fitness register of faulty individuals to '0'. This fault model (stuck at '0') is the most hazardous situation in which the faulty individuals misguide the algorithm as they seem to be good solutions. The good solution in minimum optimization has the minimum fitness value ('0' in this paper). If the faulty individual has a minimum fitness value this individual is identified as a good solution in population and can hurt the algorithm convergence.

4. 2. Benchmark Problems In order to determine how well an optimization algorithm works, a variety of test functions have been used as benchmarks. There are several benchmarks which have been widely used in the literature to test the performance of optimization algorithms. Four functions as benchmarks have been used in this study. In each case we give a general form of the function, a plot of its values in one dimension and give the global optima in its one dimensional form.

4. 2. 1. "Rastrigin" Function Rastrigin function (f_{Ras}) is a multimodal, separable and symmetric function. The objective function shown in (1) has to be minimized [21].

$$f_{RAS}(x) = 10n + \sum_{k=1}^{n}\left(x^2 - 10\cos(2\pi x_k)\right) \quad (1)$$

Equation (1) is plotted in one dimension in Figure 10. The variables range within the interval of [−5.12, +5.12], and the global minimum value is 0.0. This function is properly difficult due to its large search space and large number of local minima.

4. 2. 2. "Griewank" Function Griewangk's function has many widespread local minima regularly distributed. The function has the following definition (2).

$$f_{Gri}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^{\,2} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (2)$$

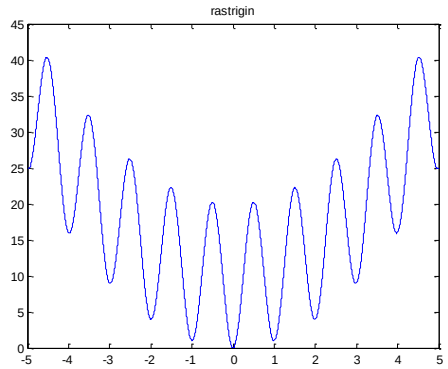


Figure 10. One dimensional Rastrigin function

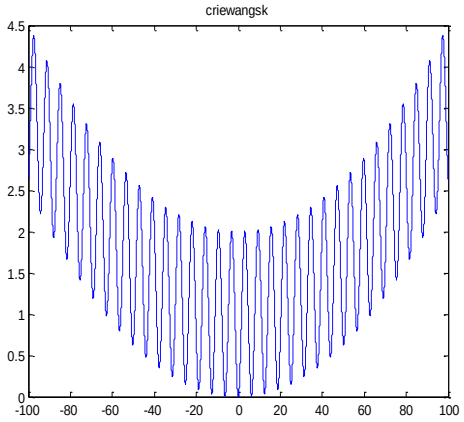


Figure 11. One dimensional Griewank function

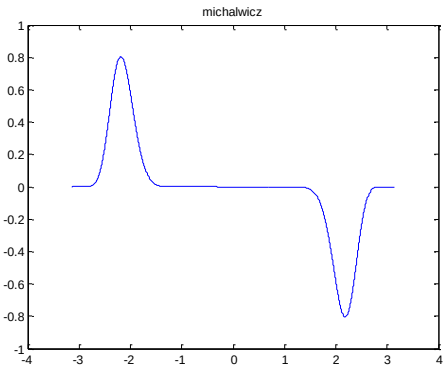


Figure 12. One dimensional Michalewicz function

Test area is usually restricted to hypercube $x_i \epsilon [-600,600]$, i = 1,...,n. Its minimum is obtainable in $f(x) = 0$ for $x_i = 0$, i = 1,...,n. The general overview suggests convex function, medium-scale view suggests existence of local extremum, and finally, zoom on the details indicates complex structure of numerous local extremum [20]. This function is plotted in one dimension in Figure 11.

4. 2. 3. "Michalewicz" Function Michalewicz function is a multimodal and separable test function with n! Local minima. The objective function to minimize is shown in (3), and the variables range in the interval of [0, π]. One dimensional form of this function is shown in Figure 12, where m = 5 and defines the steepness of the valleys [20].

$$f_{mich}(x) = -\sum_{i=1}^{n} \sin(x_i).\left(\sin\left(\frac{i.x_i^{\,2}}{\pi}\right)\right)^{2.m} \quad (3)$$

4. 2. 4. "Drop wave" Function This is a multimodal test function [20]. The objective function to minimize is shown in (4), and test area is usually restricted to the square $x_i \epsilon$ [-600,600]. One dimensional Drop wave function plot is shown in Figure 8.

$$f_{drop}(x_1, x_2) = -\frac{1+\cos\left(12\sqrt{x_1^{\,2}+x_2^{\,2}}\right)}{\frac{1}{2}\left(x_1^{\,2}+x_2^{\,2}\right)+2} \quad (4)$$

The considered search space dimension is n=1 for all the above mentioned problems.

4. 3. Simulation Setup As mentioned in previous section, four functions have been used as benchmarks in this simulation. Equations (1), (2), (3) and (4) represent Rastrigin's, Griewank's, Michalewicz's and Drop wave's function, respectively. Table 2 summarizes the details of our test functions. The proposed FT-cGA terminates when the *average fitness-value* of population is below a threshold value (*average fitness-value* ≤ threshold) that is shown in the last column of Table 2.

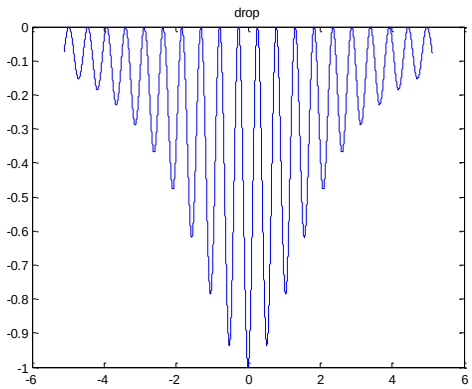


Figure 13. One dimensional Drop wave function

TABLE 2. The properties of Benchmark functions including their dimensions, Global optima and stopping criteria

Function	Domain of x	Global optima (x, y)	Stopping criteria (threshold)
F_Mich	X ϵ [0,π]	(2.18328,-0.806919)	-0.806
F_Ras	X ϵ [-5.12,5.12]	(0,0)	0.00005
F_Gri	X ϵ [-600,600]	(0,0)	0.00005
F_Drop	X ϵ [-5.12,5.12]	(0, -1)	-0.99999

TABLE 3. FT-nD-cGA parameters used in the simulation

Parameters	FT-2D-cGA	FT-3D-cGA
Neighborhood	east, west, north, south	east, west, vertical north and south, horizontal north and south
Parent selection	Best neighborhood	Best neighborhood
Recombination	One point cross over	One point cross over
Mutation	Bit flip	Bit flip
Max generation	300	300
Chromosome length	32	32
Replacement	Replace if better	Replace if better
Termination condition	Avg_fitness ≤ threshold value	Avg_fitness ≤ threshold value

The proposed FT-nD-cGA was simulated in 2D and 3D structures. Several tests are considered in order to show the ability of the FT-nD-cGA to overcome SEUs at fitness score registers. Considering different population sizes, the algorithm's parameters are summarized in Table 3.

4. 4. Evaluation Metrics The results are analyzed based on three metrics: the efficiency is measured as the average number of generations for successful runs out of 100 independent runs., The second one is the search success rate of successful experiments out of 100 independent runs which represents the efficacy of the algorithm, and the third is the ratio of efficiency to efficacy (see Equation (5)).

The best structure of optimization algorithm for each test function must have low efficiency and high efficacy which results in low γ. Equation (5) gives the formula of performance measure (γ).

$$\gamma = \frac{effciency}{efficacy} \quad (5)$$

5. RESULTS

In Figures 14-16, the average number of generation, search success rate and γ are depicted respectively vs. population size. In these diagrams we can see if the population size increases, the average number of generation and ratio parameter decrease and search success rate increases. Furthermore, the algorithm converges faster. In addition these diagrams show the rank of our test functions in three metric parameters.

Figures 17- 19, are similar to Figures 14- 16 respectively, but the only difference is in the structure dimensions of FT-cGA.

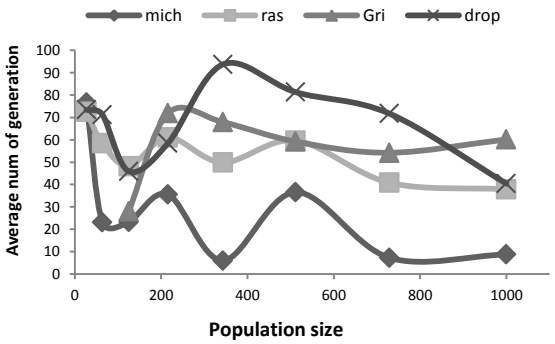


Figure 14. Efficiency parameter for 3D structure of FT-cGA. Efficiency is the average number of generations for successful runs.

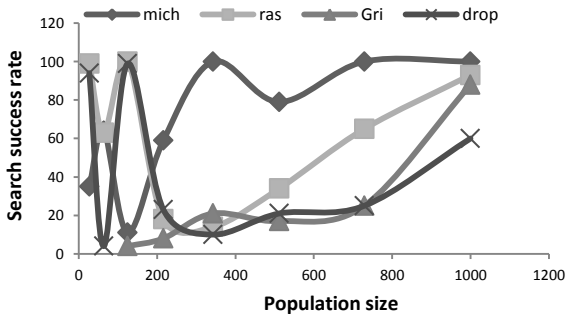


Figure 15. Efficacy parameter for 3D structure of FT-cGA. Efficacy is the search success rate of successful experiments.

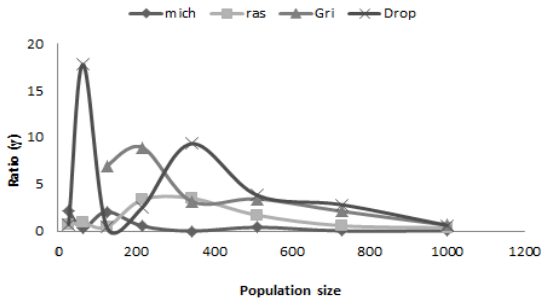


Figure 16. 'γ' parameter for 3D structure of FT-cGA.

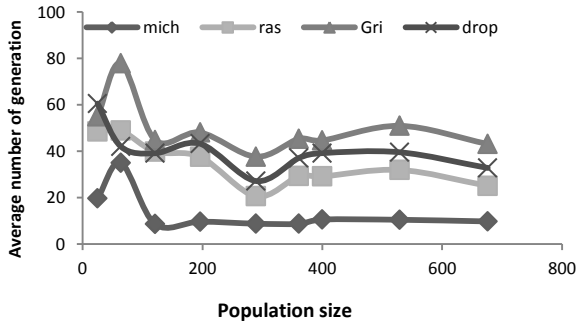


Figure 17. Efficiency parameter for 2D structure of FT-cGA. Efficiency is the average number of generations for successful runs.

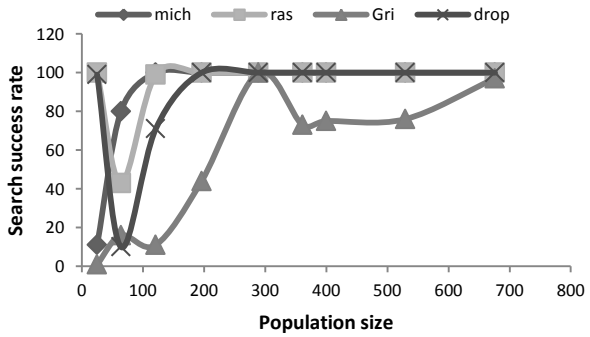


Figure 18. Efficacy parameter for 2D structure of FT-cGA. Efficacy is the search success rate of successful experiments.

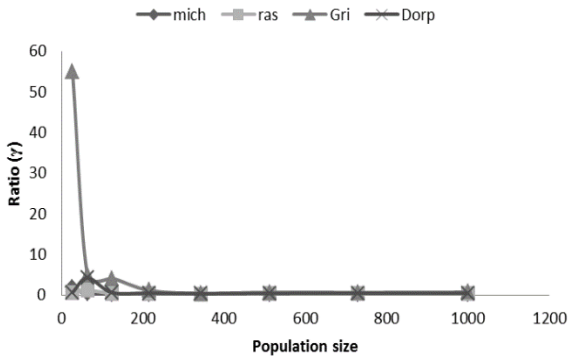


Figure 19. 'γ' parameter for 2D structure of FT-cGA

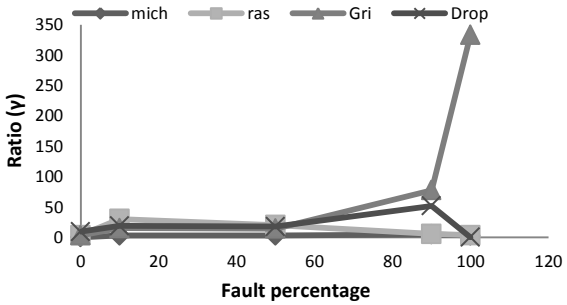


Figure 20. Impact of fault percentage in ratio parameter for all test function in 3D structure (population size = 343)

In these diagrams we can see that the population size has a direct relation with efficacy and has inverse relation with efficiency and ratio parameter. Meanwhile, in this structure (2D), we can see an increase of convergences speed of algorithm when the population size increases. Figures 20 and 21 show the influence of fault percentage in ratio parameter when the population size is 343 PEs for 3D structure and 361 PEs for 2D structure (these diagrams are obtained for all test function).

6. DISCUSSION

In all diagrams (Figures 14 - 19), it is observed that when the population size increases, the efficiency and ratio (γ) parameter decrease and the efficacy parameter increases. This happens due to the increase of cGA computational capacity caused by the population size enlargement. Figures 14- 19 can be used to rank the test functions in efficacy, efficiency and γ parameters. Considering Figures 20 and 21, it can be deduced that for fault percentage below 90%, the 2D is better than 3D structure and for fault percentage more than 90% the 3D structure has a better performance. This outcome can be used to minimize the hardware overhead while minimizing the benchmark functions in different faulty conditions. A more general conclusion from Figures 14-19 can be about determining the best dimensions for the cGA architecture. Based on the aforementioned data, Table 4 gives the best dimension of cGA architecture for each benchmark as a function of the fault percentage and population size. The summarization is not as straightforward as the previous discussed parameters (efficiency,…). It may be expected that an increase in dimensions (2D to 3D) will result in better ratio (efficiency/efficacy); however, this is not the observed phenomenon. The experimental results support this fact that the performance highly depends on the type of benchmark function, population size and fault percentage.

TABLE 4. Comparison of two structures of FT-cGAs in ratio parameter for all test functions, different sizes of population and fault percentages.

fault percentage \ function	F_mich	F_ras	F_Gri	F_drop
0%	2D	2D , pop size < 68 3D , 68 < pop size < 115 2D , pop size > 115	3D , pop size < 108 2D , pop size > 108	2D , pop size < 44 3D , 44 <pop size < 128 2D , pop size > 128
10%	3D , pop size < 57 2D , 57 <pop size < 341 3D , pop size > 341	3D , pop size < 125 2D , pop size > 125	3D , pop size < 76 2D , pop size > 76	3D , pop size < 139 2D , pop size > 139
50%	3D , pop size < 68 2D , 68 <pop size < 341 3D , pop size > 341	3D , pop size < 248 2D , pop size > 248	2D	3D , pop size < 265 2D , pop size > 265
90%	2D	3D , pop size < 119 2D , pop size > 119	2D	3D , pop size < 133 2D , pop size > 133
1 PE healthy	3D , pop size < 133 2D , pop size > 133	2D	3D , pop size < 251 2D , pop size > 251	3D , pop size < 119 2D , pop size > 119

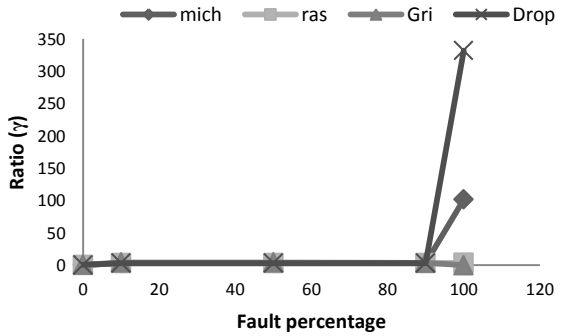


Figure 21. Impact of fault percentage in ratio parameter for all test function in 2D structure (population size = 361)

In this table we can see in low population size, to optimize all test functions, the three-dimensional structure of fault tolerant cGA, is superior to the two-dimensional structure, since the count of neighborhood in three dimensional is bigger than two dimensional structure.

7. CONCLUSION AND FUTURE WORK

In this study, we proposed VHDL implementation of a family of fault tolerant cellular genetic algorithm that is robust to SEUs when applied to fitness score register. This approach is based on canonical cellular genetic algorithm that is explained in algorithm 1. The connectivity of PEs and migration policies are important tools used in our strategy to prevent spreading faulty answer and repair effects of faulty individuals.

In the experiments, two topologies have been tested (3D and 2D) and their performance were analyzed and compared with three metrics: efficacy, efficiency and their ratio, and we consider the worst case of fault model (stuck at '0' in minimum optimization) with different fault percentages. The experimental results show that the proposed method is very robust to recovering SEU error up to at least one PEs of population is healthy. In addition, using the efficient replacement rule in restoration step, the algorithm can use the correct answers that were produced in first stage; therefore, the algorithm does not have a lot of time overhead for reaching the optimal answer. Finally, the table was designed, that represents which topology of fault tolerant cGA is excellent in a variety of conditions (the conditions include; size of population and error percentage occurred for optimization of all test functions). In this paper the reliability of proposed structures are evaluated with some metrics; However, other metrics can be used to measure their reliability inspiring from literatures [21, 22]. Another task for future can be the study of behavior of emergence of fault tolerance while using this simulated evolution in genetic algorithms as in reported by Damavandi et al. [22].

8. ACKNOWLEDGEMENT

The authors wish to thank the anonymous reviewers and editors for the useful and constructive comments which have improved the article.

9. REFERENCES

1. Giacobini, M., Tomassini, M., Tettamanzi, A.G. and Alba, E., "Selection intensity in cellular evolutionary algorithms for regular lattices", ***Evolutionary Computation, IEEE Transactions on***, Vol. 9, No. 5, (2005), 489-505.
2. Tomasini, M., Spatially structured evolutionary algorithms., Springer, Berlin/Heidelberg. (2005)
3. Alba, E. and Troya, J.M., "Cellular evolutionary algorithms: Evaluating the influence of ratio", in Parallel Problem Solving from Nature PPSN VI, Springer. (2000), 29-38.
4. Alba, E. and Dorronsoro, B., "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms", ***Evolutionary Computation, IEEE Transactions on***, Vol. 9, No. 2, (2005), 126-142.

5. Al-Naqi, A., Erdogan, A.T., Arslan, T. and Mathieu, Y., "Balancing exploration and exploitation in an adaptive three-dimensional cellular genetic algorithm via a probabilistic selection operator", in Adaptive Hardware and Systems (AHS), NASA/ESA Conference on, IEEE. (2010), 258-264.
6. Al-Naqi, A., Erdogan, A.T. and Arslan, T., "Fault tolerant three-dimensional cellular genetic algorithms with adaptive migration schemes", in Adaptive Hardware and Systems (AHS), NASA/ESA Conference on, IEEE. (2011), 352-359.
7. Ortega-Sanchez, C., Mange, D., Smith, S. and Tyrrell, A., "Embryonics: A bio-inspired cellular architecture with fault-tolerant properties", ***Genetic Programming and Evolvable Machines***, Vol. 1, No. 3, (2000), 187-215.
8. Xu, J., Arslan, T., Wang, Q. and Wan, D., "An ehw architecture for real-time gps attitude determination based on parallel genetic algorithm", in Evolvable Hardware,. Proceedings. NASA/DoD Conference on, IEEE. (2002), 133-141.
9. Breukelaar, R. and Bäck, T., "Using a genetic algorithm to evolve behavior in multi dimensional cellular automata: Emergence of behavior", in Proceedings of the 7th annual conference on Genetic and evolutionary computation, ACM. (2005), 107-114.
10. Morales-Reyes, A., Al-Naqi, A., Erdogan, A.T. and Arslan, T., "Towards 3d architectures: A comparative study on cellular gas dimensionality", in Adaptive Hardware and Systems,. AHS. NASA/ESA Conference on, IEEE. (2009), 223-229.
11. Das, S., Chandrakasan, A. and Reif, R., "Three-dimensional integrated circuits: Performance, design methodology, and cad tools", in VLSI,. Proceedings. IEEE Computer Society Annual Symposium on, IEEE. (2003), 13-18.
12. Morales-Reyes, A., Stefatos, E.F., Erdogan, A.T. and Arslan, T., "Fault tolerant cellular genetic algorithm", in Evolutionary Computation,. CEC.(IEEE World Congress on Computational Intelligence). IEEE Congress on, (2008), 2671-2677.
13. Morales-Reyes, A., Erdogan, A.T., Arslan, T. and Stefatos, E.F., "Towards fault-tolerant systems based on adaptive cellular genetic algorithms", in Adaptive Hardware and Systems,. AHS'08. NASA/ESA Conference on, IEEE. (2008), 398-405.
14. Morales-Reyes, A., Haridas, N., Erdogan, A.T. and Arslan, T., "Fault tolerant and adaptive gps attitude determination system", in Aerospace conference, IEEE , (2009), 1-8.
15. Al-Naqi, A., Erdogan, A.T. and Arslan, T., "Fault tolerance through automatic cell isolation using three-dimensional cellular genetic algorithms", in Evolutionary Computation (CEC), IEEE Congress on. (2010), 1-8.
16. Al-Naqi, A., Erdogan, A.T. and Arslan, T., "Dynamic fault-tolerant three-dimensional cellular genetic algorithms", ***Journal of Parallel and Distributed Computing***, Vol. 73, No. 2, (2013), 122-136.
17. Schmidt Jr, F.H., "Fault tolerant design implementation on radiation hardened by design sram-based fpgas", Massachusetts Institute of Technology, (2013),
18. Normand, E., "Single event upset at ground level", ***IEEE Transactions on Nuclear Science***, Vol. 43, No. 6, (1996), 2742-2750.
19. Nedjah, N. and de Macedo Mourelle, L., "An efficient problem-independent hardware implementation of genetic algorithms", ***Neurocomputing***, Vol. 71, No. 1, (2007), 88-94.
20. Molga, M. and Smutnicki, C., "Test functions for optimization needs", ***Test Functions for Optimization Needs***, (2005).
21. Nailwal, B. and Singh, S., "Reliability measures and sensitivity analysis of a complex matrix system including power failure", ***International Journal of Engineering-Transactions A: Basics***, Vol. 25, No. 2, (2012), 115-123.
22. Damavandi, Y.B., Mohammadi, K., Upegi, A. and Thoma, Y., "On feasibility of adaptive level hardware evolution for emergent fault tolerant communication", ***International Journal of Engineering-Transactions A: Basics***, Vol. 27, No. 1, (2013), 101-112.

A Comparative Study of VHDL Implementation of FT-2D-cGA and FT-3D-cGA on Different Benchmarks

RESEARCH NOTE

P. Ashooriyan, Y. Baleghi

Department of Electrical & Computer Engineering,Babol Noshirvani University of Technology, Babol, Iran

P A P E R I N F O

Paper history:
Received 18 September 2014
Received in revised form 30 June 2015
Accepted 30 July 2015

Keywords:
Fault Tolerance
Cellular Genetic Algorithm
FT-2D-cGA
FT-3D-cGA
Processing Element
Single Error Upset

چکیده

این مقاله، شبیه سازی سخت‌افزاری (بر اساس کد VHDL) خانواده‌ای از الگوریتم ژنتیک سلولی تحمل‌پذیر خطا را ارائه می‌دهد. هدف این مقاله افزایش مقاومت الگوریتم ژنتیک سلولی تحمل‌پذیر خطا در زمانی است که خطای واژگونی رخداد یکتا (SEU) مقدار ثبّات برازندگی جواب‌ها را تغییر داده است. الگوریتم مورد نظر، از دو مرحله تشکیل می‌شود، مرحله‌ی تشخیص خطا و مرحله‌ی ترمیم جواب خطادار. استفاده از قوانین ابتکاری اتصال و قوانین کارآمد سالم‌سازی المان‌های پردازشی مهم‌ترین نکات استفاده شده در الگوریتم مورد نظر می‌باشند. در نتایج آزمایش‌ها، سه معیار و چهار تابع آزمون برای نمایش عملکرد الگوریتم مورد نظر استفاده شده است. دو ساختار (دوبعدی و سه‌بعدی) از الگوریتم ژنتیک تحمل‌پذیر خطای مورد نظر، برای بهینه‌سازی توابع آزمون جهت آنالیز و مقایسه‌ی دو ساختار استفاده شد. نتایج آزمایش‌ها، نیرومندی الگوریتم پیشنهادی را اثبات می‌کند و نشان می‌دهد که الگوریتم مورد نظر حتی زمانی که فقط یک المان پردازشی سالم در جامعه باشد می‌تواند به جواب بهینه برسد.

***doi**: 10.5829/idosi.ije.2015.28.09c.04*