Multiple Fault Tolerant Hardware Structure for Cellular Genetic Algorithm

Peyman Ashooriyan, Yasser Baleghi Damavandi

***Abstract*— This paper presents the hardware simulation (based on VHDL code) of a multiple-fault tolerant cellular genetic algorithm. This study aims to increase the immunity of cellular genetic algorithm in multiple-fault situation. Here, multiple-fault refers to the situation that SEU (single event upset) occurs simultaneously at one or more bits of the chromosome and fitness registers. The fault model includes simultaneous bit inversion in chromosome strings and worst case stuck faults in fitness registers. The main idea of the proposed approach is to control the exploitation/exploration trade off in fault recovery phase. The achievements of this experiment are due to applying CRC encoding; novel recovery strategy and new scheme in connections of processing elements. In order to show valid conclusions, the algorithm is tested with four benchmarks in various fault situations based on popular evaluation metrics. In experimental results, two topologies (two and three-dimensional) of suggested FT-cGA are evaluated. To illustrate the immunity and achieved promotion, the proposed FT-cGA will be compared with canonical version of cGA. The whole results show that the proposed architecture is able to handle multiple-faults with up to 100% of faulty processing elements.**

***Index Terms*— Cellular Genetic Algorithm, Fault Tolerance, Single Event Upset (SEU), Processing Element (PE).**

I. Introduction

A large number of optimization techniques for solving real world problems (NP-hard) exist in the literature [1, 2]. Among them, Evolutionary Algorithms (EAs) are very popular optimization techniques that imitate the biological process found in Nature [3, 4, 5]. Evolutionary algorithms are intrinsically parallel and suitable for hardware implementation.

Serial and parallel versions are two members of Evolutionary Algorithms that have been applied on a set of individuals (population), where each individual represents a possible solution to the problem. When serial version is considered the evolutionary process, it is applied on a single population; in contrast, in parallel version a special structure of population is used. Moreover, in parallel version, each individual can be evolved independently. Two main models of Parallel EAs are, distributed and cellular. In distributed model, several independent subpopulations (islands) interact with other islands but in cellular EA there is only one population of processing element that is processed in parallel [6]. In this class, each processing element has data interaction only with its neighborhoods. In this paper, we focus on cellular genetic algorithm (cGAs).

The balance between exploitation of good solutions and exploration of new areas of the search space made by this kind of algorithms is one of the important elements for their high performance. This exploration/exploitation tradeoff can be improved with some different parameters of the cellular genetic algorithm such as the shape of the population, shape and size of the neighborhood, the different applied operators or the probability of applying them [7-11].

Technology developments such as considerable reduction in size of transistors and use of new materials and system on chip (SOC) architectures carry on to increase the sensitivity of a system to soft errors [12]. Single event upset (SEU) is the sub class of soft error. SEUs have the main effect (80%) of radiation effects in the space [13]. After SEU, was discovered in space in 1975, Ziegler and et al. illustrated the potential for microelectronics on the ground to be sensitive to SEU from cosmic ray secondary's, primarily neutrons [14]. Genetic algorithm has a lot of application in engendering however, in this paper; we focused on two applications of genetic algorithm in space such as evolvable hardware and GPS attitude determination problem [15, 16]. Since in these applications the immunity of Genetic algorithm is important and more impressive, the goal of this paper is designing the hardware structure of fault tolerant genetic algorithm. Moreover, since the cellular architecture can bring about fault tolerance [17] in our approach, designing the cellular version of genetic algorithm is targeted.

Previous works in fault tolerant cellular genetic algorithm (FT-cGA) field can be categorized in two groups that each group has a distinct fault model. First, single event error (SEE) that occurs at registers that scores chromosome and next is the SEEs incurring at fitness score register. Reyes. et al [18] designed fault tolerant cellular genetic algorithm by using the basic competency of this algorithm. In this reference, fault tolerance characteristic was obtained in appropriate exploitation of basic parameters and genetic operators in cGAs. The fault model of this method is occurrence of single hard error (SHE) at fitness score registers. Moreover, the use

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456".

Peyman Ashooriyan (e-mail: p.ashooriyan@stu.nit.ac.ir), Yasser baleghi damavandi (email: y.baleghi@nit.ac.ir) are with the Department of Electrical & Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran. Corresponding Author Email: y.baleghi@nit.ac.ir.

of migration operator and controlled selection intensity, are important gadgets in this method. In [19] an adaptive method was described that is robust to SHEs when incurred at chromosome registers. In this method, decreasing genotype diversity of chromosome register is caused by occurrence of SHE errors. In [20], cellular Genetic algorithm and the distributed one are merged to one structure. SHEs tolerance when affecting at fitness register is the fault model of this reference. In this strategy, when a fault is diagnosed, the migration is done between cellular and the distributed structure. Meanwhile, the impression of migration policies and adaptive design is very important in this method. As quoted in the recent literature [8, 21, 22], these authors designed fault tolerant cellular genetic algorithm that is immune to SEU fault when applied up to 40% of processing elements fitness registers.

In the previous works, Hardware simulation is not reported, while, in this study the VHDL implementation of fault tolerant cellular genetic algorithm is targeted. Another important contribution of this paper is Multiple Fault analysis. In this work both chromosome and fitness registers are prone to stuck faults. Consequently, the proposed design has been developed due to the increased probability of SEU phenomenon that may cause multiple fault situations.

In this paper, two and three dimensional fault tolerant cellular Genetic algorithm structures are implemented in VHDL code. The experimental results indicated the ability of the proposed theory to self-repairing property of all processing elements.

This paper is divided into six chapters. The five remaining chapters are organized as follows: Section 2 presents the fundamental definitions of proposed theory. The recommended architecture is discussed in Section 3. Section 4 presents the simulation setup and shows obtained results. The discussion part is presented in section 5. Finally, concluding notes are given in Section 6.

II. FUNDAMENTAL DEFECTIONS

In this section, we briefly summarize a background for the cellular genetic algorithm. We also introduce concepts about fault tolerance and CRC error-coding method that used in this study.

A. Fault Tolerance

The digital circuits located in the space environment and at ground level [23] are attacked by the charged particles generated by the solar flash. Consequently, additional techniques must be used to avoid radiation defect. Therefore, the performance of a circuit is more sensitive to radiation environment. Dense devices need less significant feature size; this means the data is stored with a smaller amount charge or current. Each of these properties creates the device more hazardous against radiations anomaly. Single Event Effect (SEE) is a temporary effect hit by a single charged particle through the silicon. Significant sources of SEE in space are trapped protons, solar protons, neutrons and heavy ions from galactic cosmic rays [24]. Single Event Effects are divided into two main categories: soft and hard SEEs. Soft SEEs are not stable they are overset by resetting the system or rewriting data in a memory cells. In contrast, Hard SEEs have stable change to the operation of a device. Soft errors called Single Event Upsets (SEU) that changes the state of a memory cell. An SEU may occur in analogue, digital, optical components, or may have effects in surrounding interface circuitry. SEUs can be categorized in first (A single bit upset), second and third (multiple bit upset) order effects, according to the number of defects that occur at the same time in the circuit. multiple bit upset (MBU) can occur when a single charged particle traveling through the IC at a narrow angle, nearly parallel the surface of the die, concurrently attacks two sensitive junctions by direct ionization or atomic eject [25,26]. Single hard error is similar to SEU but the memory cell got a stable value and it causes permanent functional damages.

In previous works, the SEU mitigation techniques on digital circuits can be done by software, hardware and combinational of both strategies. In hardware strategy, the mitigate operation is fast and the main drawback of this approach is the cost. Moreover, this strategy cannot restrain all types of random and multiple bit errors but in software approach Time Redundancy, Hardware Redundancy, Triple Modular Redundancy (TMR) and multiple redundancies with voting are several members of this schema. Since the SEE had 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.

B. Cellular Genetic Algorithm

Cellular Genetic Algorithms (cGA) are a subclass of evolutionary algorithm as consisting of a decentralized population. Generally, in cGA each individual is assigned a lattice position (cell); the lattice configuration is typically arranged on an n-dimensional grid that having a linear, square, or rectangular geometric shape. Each cells called processing element and the evolutionary process is runed in each processing elements. Cells of lattice interact with other local neighborhoods cells [27].

Algorithm 1 shows the pseudo-code of the canonical cGAs. A cGA starts with a random population, next, a fitness value assigned to each individual. After that, the genetic operators such as selection, crossover, mutation and replacement are applied to each individual and this process continues until the termination condition is seemed.

Algorithm 1: Pseudo-code of a canonical cGA

```
1. Procedure cGA:
2. Pop ← Generate initial population (Pop);
3. Pop ← Evaluation (Pop);
4. While not stop_condition do
5.    For I ←1 , Pop.size do
6.       Neighbors ← find.neighbours (position(i));
7.       Parent1 ←  position(i);
8.       Parent2 ← local_selection (neighbours);
9.       offspring← recombine (Pc , parent1, parent2);
10.      Evaluation ← fitness (offspring);
11.      Replacement (position(i), offspring, Popaux );
12.   End for;
13.   Pop←Popaux;
14.  End while;
15.End procedure;
```

C. *Cyclic Redundancy Check (CRC)*

Many communication systems use the cyclic redundancy check (CRC) method to protect important data from errors [28-31]. The CRC has been widely used in digital communications because of its performance, simplicity and low usage of software and hardware resource [32-34]. In this study, the ability of error detecting of CRC is used but this method in some case can used for error correcting applications [32].

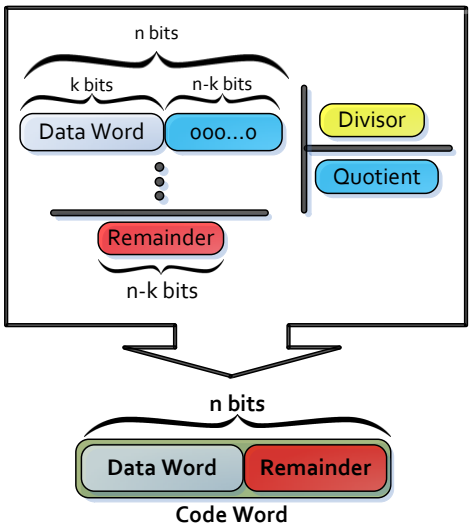


Fig. 1. Principle of CRC algorithm.

In the CRC coding, data word (M) has k bits; the code word (T) has n bits. The size of the data word is enhanced by adding n-k, 0s to the right-hand side of the word (e.g., 2^{n-k}M or M is shifted left n-k bit). The n-bit result is generated sent to the generator. The generator uses a divisor of size n-k+1, predefined and agreed upon. The generator divides the enhanced data word by the divisor (P). The quotient of the division is discarded; the remainder (R) that is called CRC (n-k bits) is added to the shifted data word (2^{n-k}M) to create the code word (T). In other word (see Fig. 1), the data word (M) shifted n-k bit left and divides to predefined divisor (P) then add to the remainder of this divide ($T=2^{n-k}M+R$). In this case, addition and subtraction are the same. We use the XOR operation to do both [35].

The code word (T) can change during operation (in this case, SEU fault injection). The decoder and encoder part have a same division process. The remainder (R) of the division is the syndrome. If the syndrome is a zero vector, code word is healthy and the data word (M) is separated from the T and accepted. Otherwise, everything is discarded.

Although, mathematically, a CRC can be explained as a polynomial over Galois Field (GF) and performing polynomial division by a generator polynomial G(x), which is commonly called a CRC polynomial. Common CRC polynomials can detect following types of errors:

- All single bit error
- All double bit errors
- All odd number of errors
- Any burst error for which the burst length is less than the polynomial length
- Most large burst errors [36].

One of the benefits of a cyclic code is that the hardware implementation of the encoder and decoder is done using a minority of electronic devices. Linear Feedback Shift Register (LFSR) with serial data feed [37] has been used to implement the CRC algorithm (see Fig. 2). As all hardware implementations, this method directly performs a division and then the reminder, which is the resulting CRC checksum, is stored in the registers (D-flip flop) after each clock cycle. The registers can be read after all of data word is fed the LFSR. Simplicity and low power consumption are the main advantages and a low speed to produce remainder is the disadvantage of this method. Although the parallel version and fast calculation of the CRC algorithm were designed [38,39,40] but in this study the serial version of CRC method is fulfilled ultimate architecture [35,36].

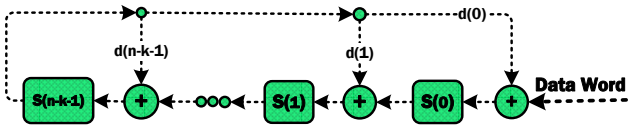


Fig. 2. The CRC encoder and decoder design using shift registers.

III. PROPOSED ARCHITECTURE

This section, discusses the proposed architecture that has the ability of dealing with SEUs. In this approach, it is assumed that the SEU attacked to the fitness and chromosome registers of the processing element at the same time. This section also describes the various modules of the Fault Tolerant cellular Genetic Algorithm (FT-cGA) system that is shown in Fig. 3. These FT-cGA modules are implemented in VHDL and consist of six sub blocks. Processing Element (PE), Initialization Block, Generation Controller, Comparator, Stop in Criteria and Fault Injection are proposed architecture sub-modules.

A. *Processing Element (PE)*

In a cGA, the population is usually arranged in an n-dimensional grid of PEs. The boundary of PEs of the grid is

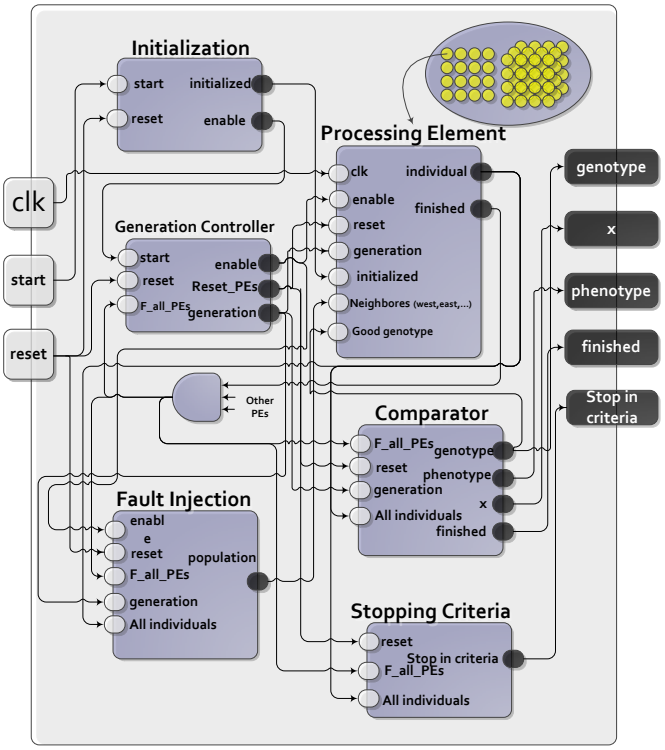


Fig. 3. Block diagram of proposed Fault Tolerant cellular Genetic Algorithm.

connected to the PE located in the opposite borders in the same row/column, depending on the case. In fact, all the PEs has exactly the same number of neighbors. In cGAs, the PEs can only interact with their local neighbors in the reproductive process (line 6 to 11 of Algorithm 1) where the genetic operators (selection, crossover and mutation and so on) are applied. This reproductive process is performed inside the PEs on PE's individual and its local neighborhood.

In this paper, the two and three dimensional structures of cGA are implemented. In 3D topology each PE has six neighborhoods -east, west, horizontal south and north, vertical south and north - but in 2D topology each PE has four neighborhoods- North, east, west and south.

The main sub block of the cGA that determine the cell resistance is processing element (PE). The suitable guideline and schemas in this module can improve performance of cellular genetic algorithm in unsafe and threatening, environment.

In this paper, it is assumed that the SEU fault occurs simultaneously at PEs chromosome and fitness registers. The operation, which is done in each processing elements of proposed structure, is shown in flowchart of Fig. 4.

First, the preprocessing algorithm (algorithm 2) is done on the neighborhoods' genotype. Detecting error in algorithm 2 is done via CRC coding method that is aforementioned. When this plan assess fault in chromosome register, the recovery process is started. This fault can be equivalent to the natural mutation that occurs at chromosome register. Increasing in exploration parameter can be observed due to this kind of error. Thus it can disturb the tradeoff between exploitation and exploration parameters. Algorithm 2 is used to control this

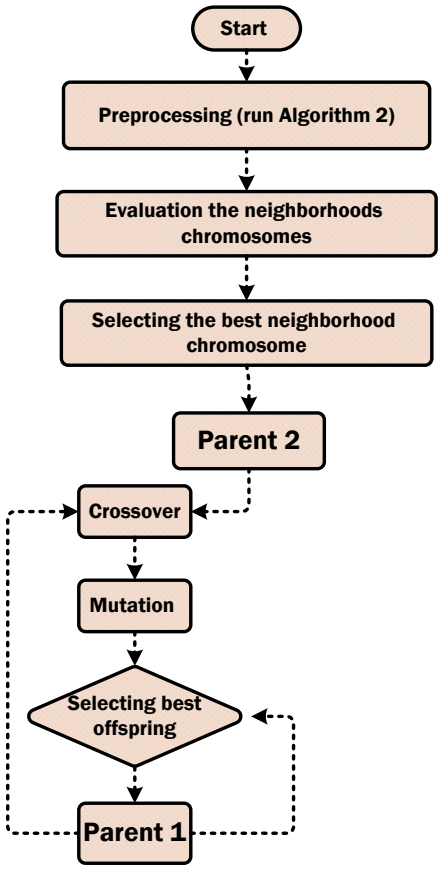


Fig. 4. Proposed process for processing element flowchart.

kind of fault and the previously mentioned tradeoff. In this algorithm, the recovery process is started when neighbors chromosome are identified as faulty. After fault-detection process when each neighbor's chromosome is assumed faulty, four candidate's chromosomes are built from present faulty chromosome and fittest chromosome of previous generation. Then reproduction process is done according to algorithm 2 (lines 7 to 14). After that, the best offspring is selected to replace the faulty neighbor's chromosome ones. After preprocessing unit, the evaluation process of fault free neighborhood is done. PEs selects the best fault free neighborhoods as parent 2. Parent 1 (internal parent) is in the current PEs. Then crossover operator (in this paper one-point cross over) recombines parent 1 and parent 2 to produce two offsprings and the fittest offspring is selected to mutation operator. The mutated offspring is evaluated and the fittest individual between mutated offspring and parent 1 is selected to stay in this PE as parent 1 for next generation. Since the fault model in this paper is SEU that occurs at fitness and chromosome register simultaneously, to prevent the error spread in population and abide it, the fitness value of individual is not used to PEs' interaction. Therefore, before start of the PEs operation (after preprocessing unit), fitness values of neighborhood genotypes are recalculated in each PE.

B. Initialization Block

In Genetic Algorithm, the initial population is usually generated randomly allowing the entire range of possible

solutions (the search space), although it is also typical to use some seeding technique in order to speed up the search by starting from excellent solutions. Similarly, in this paper, the initialization block generates random chromosomes in first generation for all PEs.

C. Generation Controller

Generation controller is a simple counter that controls and determined the generation number for all PEs. This block triggered when the entire operation of all PEs in previous generation is finished.

Algorithm 2: preprocessing pseudo code.

```
1. 1. Procedure preprocessing (cGA)
2. Neighbors ←Find_Neighbors (position (i, j, k));
3. For i ← 1 to neighbors_count do
4.   If (CRC ( neighbors(i) ) = null vector) Then
5. healthy neighbors(i)=CRC_decode (neighbors(i));
6.   Else
7.       C1 ←CRC_decode ( neighbors(i));
8.       C2 ←mutation (C1);
9.       C3 ←good_genotype;
10.      C4 ←mutation (C3);
11.    (C5,C6) ←Selection two worst C (C1,C2,C3,C4);
12.    (C7,C8) ←Selection two best C (C1,C2,C3,C4);
13.     (Child1,Child2) ← Crossover (C5,C6);
14.     healthy_ neighbors(i) ←Selection best individual
(Child1,Child2,C7,C8);
15. End if;
16.End for;
17.return healthy_ neighbors;
18.END procedure preprocessing (cGA);
```

D. Comparator

At the end of each generation when all operation of PEs is finished, this sub module selects the best chromosome of present population. Therefore, the comparator module shows the optimum solution of each generation.

E. Stopping Criteria

The genetic algorithm has a three kind of convergence conditions such as the number of generations, the number of uniquely determined schedules, and algorithm runtime. In this paper at the end of each generation stop in criteria module test the convergence criteria of evolutionary process. The termination criterion in this module is, if the average fitness of population is lower than the threshold value that is set first, the algorithm terminated. This threshold value is set based on the global optima point of benchmark functions.

F. Fault Injection

To simulate the faulty environment and test our structure in this domain we need module that injects fault. Therefore, fault injection module, injects SEU faults on PEs chromosome and fitness register simultaneously, based on defined fault percentage. The fault percentage is set as input variable. To injection fault, the candidate generations and individuals are randomly selected at the begging of proposed structure process.

IV. Experiential Study

The previous sections defined the FT-cGA algorithm and described its sub modules implementation. At the end of following section, we assess the accuracy and resistance of our predictive model. This section concentrates on proposed fault model, benchmark functions, experimental set up, evaluation metrics and discussion of obtained result.

A. Fault Model

In this paper, the fault model includes SEUs errors, specifically when targeting the individuals' fitness and chromosome score registers simultaneously. Failures happen in one or more bits in the chromosome and fitness score registers but in fitness register in such a way that keep their fitness values stuck at logical one or zero. Furthermore, this error can be found in every generation, on each PEs, randomly. Thus, in algorithmic level, a mismatch happens between chromosomes and their finesses due to aforementioned fault model. Therefore, the local selection method selects the faulty individuals (individuals with the maximum/minimum fitness values and incorrect chromosome are the fittest) and spreads the incorrect solutions in whole population and guides algorithm to the weak and incorrect solutions. To illustrate the ability of proposed structure to deal with the failures, different amounts of fault are considered in this part, from 0% to 100% of the population size.

B. Case Study

There are several benchmarks, which have been widely used in the collected works to test the performance of optimization methods. Four test functions as benchmarks have been used in this study such as: Rastrigin, Griwank, Michalewicz and Dropwave Functions.

- "Rastrigin" Function

The Rastrigin Function (1) is a classic example of non-linear multimodal function [41,42].

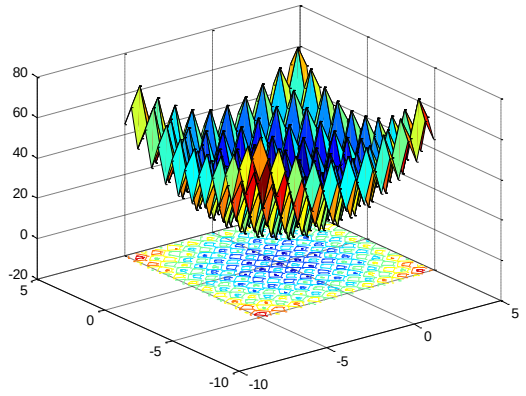


Fig. 5. 2D "Rastrigin" function

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

This function is a challenging problem due to its enormous

search space and its large number of local minima. The locations of the local minima are repeatedly distributed. The complexity of Rastrigin function is O (nlog(n)), where n is the dimension of the problem. The search domain is [-5.12, 5.12] in each variable and the optimum solution of the problem is the vector v = (0, ..., 0) with F(v) = 0 (see Fig. 5).

- "Griewank" Function

The Griewank function [41,42] is multimodal and non-

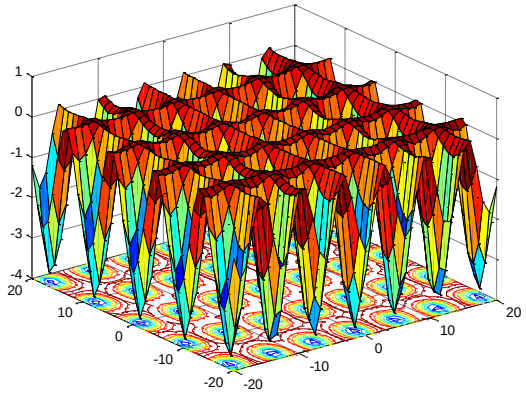


Fig. 6. 2D "Griewank" function

separable and is alike to Rastrigin but the count of its local optima is larger. The function is defined as follows:

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

This function has several local optima (Fig. 6) within [-600, 600], where n is the number of dimensions of the function. Its global minimum equal f(x) = 0 is obtainable for xi = 0, i = 1,...,n. Moreover, Griewanks number of minima grows exponentially as its number of dimensions increases.

- "Michalewicz" Function

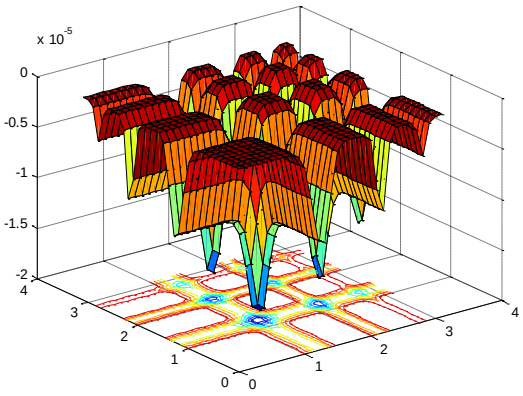


Fig. 7. 2D "Michalewicz"

The Michalewicz function [41,42] is a multimodal and separable test function with n! Local optima (3).

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

In (3), the variables domain is in the [0,π]. In this equation,

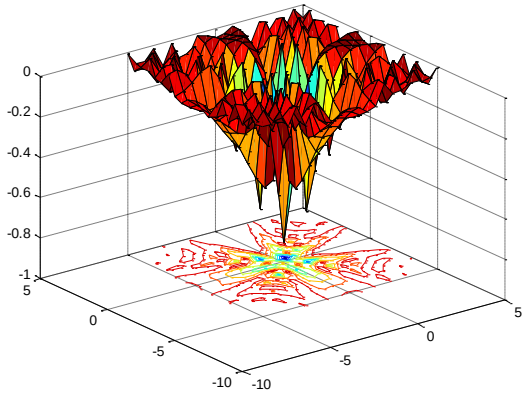


Fig. 8. 2D "Drop wave"

the m parameter defines the "steepness" of the valleys of function (Larger m leads to more difficult search). In this study m set as five. In Fig.7 the two dimensions of this function is plotted.

- "Drop wave" Function

Drop wave is a multimodal test function [41,42]. The test area of this function is in for [-5.12, 5.12] all variables. (4) defines the Drop wave's function:

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

The two variable of this function in aforementioned search space is showed in Fig. (8).

TABLE I
BENCHMARK FUNCTIONS

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

C. Simulation Setup

To show the ability of proposed structure in faulty space, in the previous section, 4 test functions are introduced. Table (1) reviews the characteristics of selected test functions. 'n' is set to '1' for all test functions.

The last column of above table shows the convergence condition of genetic algorithm in optimization of each test function separately. There are three techniques to converge of optimization algorithms. One of them is applying the rules and when the algorithm satisfies that condition, the optimization is

TABLE II
DETAILS OF SIMULATION SETUP

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	28	28
Replacement	Replace if better	Replace if better
Termination condition	Avg_fitness ≤ threshold value	Avg_fitness ≤ threshold value
CRC length	16	16

stopped. In this study when the average number of population finesses in each generation is lower than the threshold value, the algorithm is finished. This threshold value is shown in the last column of above table that is selected according to global optima point of each test function.

The proposed architecture (FT-nD-cGA) is implemented as two and three dimensional structures of population. The details of simulation setups are displayed in Table (II).

D. *Evaluation Metrics*

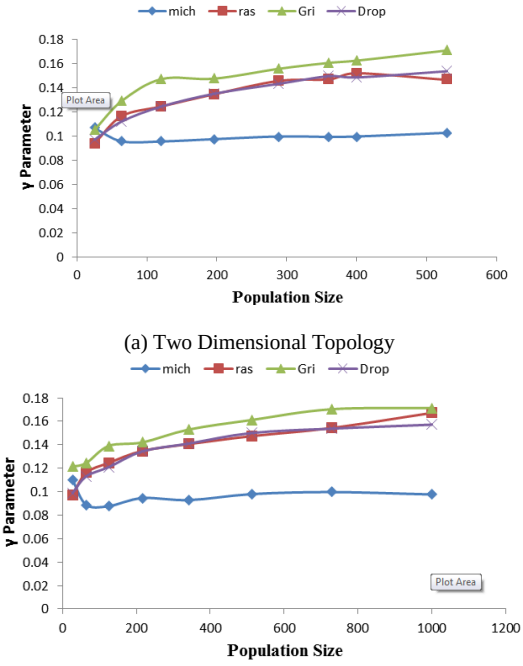
In optimization algorithms, some evaluation metrics are needed to display the performance or superiority of different structures of them. In this part, according to evaluation metrics that are used in [6, 8, 18-22] two metrics are used in this study. These metrics are efficiency and efficacy. Efficiency and efficacy are measured as the average number of generation and the search success rate of successful experiments out of 100 independent runs respectively. Therefore, the best architecture of optimization algorithm for each test function needs to have low efficiency and high efficacy as a result in low γ. Equation (5) provides the formula of γ parameter.

$$\gamma = \frac{efficiency}{efficacy} \tag{5}$$

E. *Simulation Result*

In this part, diagrams are according to γ parameter because of its generalization. Therefore, γ metric shows the performance of algorithm.

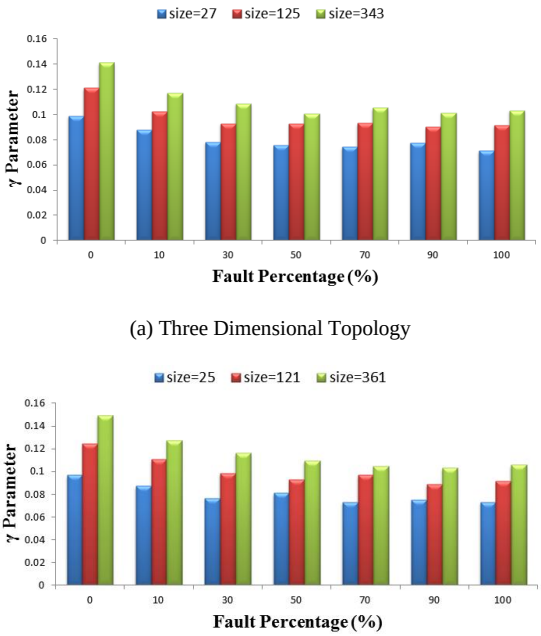
In Fig.9, the γ, is represented vs. population size in two and three dimensional structures of population in proposed architecture. Moreover, these diagrams are obtained when all of PEs are healthy and no SEU faults occurred. In these diagrams, γ parameter is enhanced due to increasing of the population size for each test function. Furthermore, the algorithm converges slower (based on desired convergence condition). In addition, we can infer from these diagrams to show the rank of test functions based on γ parameters.



(a) Two Dimensional Topology

(b) Three Dimensional Topology

Fig. 9. 'γ' parameter for 3D and 2D structure of FT-cGA.



(a) Three Dimensional Topology

(b) Two Dimensional Topology

Fig. 10. Fault Percentage and population size for 3D and 2D structure of FT-cGA in "Drop wave" function.

Fig.10 shows the influence of fault percentage on proposed structure. In this figure, the effect of fault percentage in ratio parameter for three separate population size and two distinct

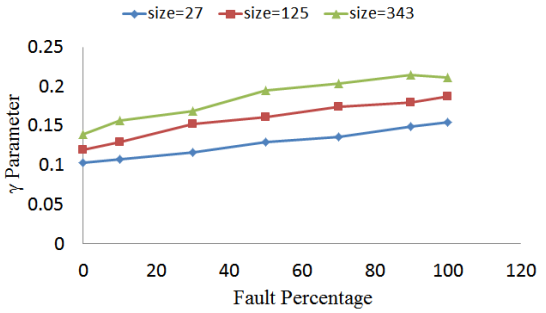
structures are shown (these diagrams are acquired for "Drop Wave" test function).

The reason of applying Fig.11 in this part is to make a comparison between canonical and proposed cellular genetic algorithm in faulty environments. These diagrams are designed for "Rastrigin" function in three-dimensional topology. The superiority of the proposed method versus canonical method can be seen in these figures.

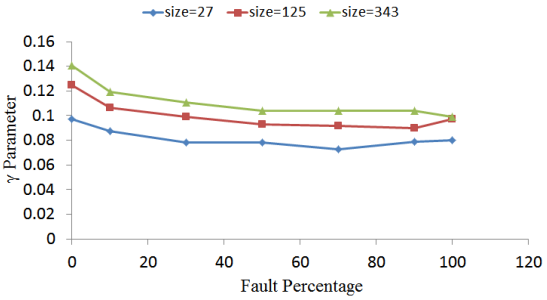
F. Discussion

In Fig.9, it can be seen that (Except for function diagrams obtained mich) the γ parameter is increased due to increase the size of population. In the structure with bigger population, more effort is needed to reduce the average number of generation under the defined threshold so the average number of generation for bigger population is increased.

In proposed structure, SEU errors on chromosome register can be detected by CRC method. The best individual of previous generation and faulty chromosome is sent to



(a) Canonical cGA



(b) Proposed FT-cGA

Fig. 11. Fault Percentage and Population Size for 3D structure of canonical-FT-cGA and proposed-FT-cGA in "Rastrigin" function.

algorithm 2 to produce the healthy and graceful chromosome to replace by faulty one. Therefore, we can see in Fig.10 that an increase in error percentage reduces γ parameter. In other word, the proposed algorithm used best individual of previous generation and faulty chromosome as a natural mutant individual to enhance the mutation and crossover percentage and improve the trade of between exploration and exploitation of algorithm. Consequently, the proposed structure has a good performance in the faulty environment.

It can be seen in Fig. (11), when the fault percentage in radiation environment is grown, the γ parameter of proposed structure decreases (the algorithm converges faster) but in canonical version of cGA (Fig. (11)-a) the enhancing fault percentage has a direct effect with γ parameter and the canonical version has not a good performance and in some cases it is may not be able to reach the optimum solution. In addition, this figure shows that proposed structure used fault to speed up the convergence time.

The empirical results illustrate that the algorithm performance significantly depends on the type of benchmark function, population size and fault percentage. In Table (3) the two and three-dimensional topology of proposed structure based on mentioned parameters are compared. In this table, the superiority of three-dimensional structure is shown, since good solution can spread faster in population due to the large number of neighborhoods.

V. Conclusion

This study proposed a VHDL implementation of a family of FT-cGA to control failures that occur in individuals' fitness and chromosome score registers, simultaneously. This approach is based on the canonical cGA that is explained in algorithm 1, and is a complete hardware reconfiguration. Herein, we considered the worst fault model (stuck at 0) together with different fault percentages. The worst-case fault model was injected in combining with different fault ratios.

The idea behind this study was to recuperate the immunity and performance of FT-cGA over control of the exploration/exploitation trade-off (detect fault and recover based algorithm 2) and comparing proposed FT-cGA performance when a two and three-dimensional topology of structure is used to solve hard optimization functions in different fault percentage. The CRC method is used to detect fault and enabled recovery method. The recovery method helped to control the exploration/exploitation trade off with use of crossover and mutation operator in recovery step (algorithm 2). On the other hand, the fault enhanced the exploration parameter but recovery tools fixed the exploitation/exploration trade off. Furthermore, fitness wire in PEs is not used in connectivity to prevent the fault spread effect in population based on fitness failure.

To evaluate the improvements, proposed FT-cGA was compared with canonical-cGA in terms of γ parameter (efficiency, efficacy). The canonical version of cGA is intrinsically immune but the fault percentage affects its performance. This problem is solved in proposed architecture. In the proposed architecture the fault is used to enhance the performance. Furthermore, proposed structure can recover from up to 100% faults on PEs. Using algorithm 2 as a relief method to immunity, the approach offers considerable improvements in γ parameter and reliability of the algorithms, especially for cases with high ratio of faults. In experimental results, we note that the proposed architecture with initiative recovery tools and changes in PEs connectivity, showed good performance. In addition, using the well-organized recovery policy, the algorithm can use the previous generation fittest answers to recover faulty individual; therefore, algorithm does not have much time overhead so this structure is good for real

TABLE II
COMPARISON OF TWO STRUCTURES OF FT-CGAS IN RATIO PARAMETER FOR ALL TEST FUNCTIONS, DIFFERENT SIZES OF POPULATION AND FAULT PERCENTAGES

Function	Michalewicz		Rastrigin		Griewank		Drop wave	
Dimensional→ Fault ↓	2D	3D	2D	3D	2D	3D	2D	3D
0%	Size<40	Size>40	Size<484	Size>484, Size ϵ [209,484]	Size<58	Size>58	Size<107	√
10%	×[a]	√[b]	Size<78	Size>78	Size<58	Size>58	×	√
30%	×	√	√	×	Size>265	Size<265	Size<50	Size>50
50%	Size<123 Size ϵ [123,189]	Size>189 Size ϵ [123,189]	Size<243	Size>243	×	√	×	√
70%	Size<307	Size>307	×	√	Size<50	Size>50	Size<45 Size ϵ [231,292]	Size ϵ [45,292]
90%	×	√	√	√	Size>76	Size<76	Size<231, Size ϵ [231,328]	Size>328
100%	Size>115	Size<115	Size<249	Size>249	×	√	×	√

[a]×= the selected dimension is not suitable for optimization.
[b]√= the selected dimension is suitable for optimization in all domains.

time application. Finally, Table-II was considered, that represents which topology of proposed FT-cGA (2D or 3D) is better in a variety of conditions (population size, fault percentage occurred for optimization of all test functions).

REFERENCES

[1] E. Alba. Parallel Metaheuristics: A New Class of algorithms. Wiley, October 2005.
[2] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys, 2003,35(3):pp.268–308.
[3] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation, October 2002, 6(5):pp.443–462.
[4] T. B¨ack. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, New York, 1996.
[5] T. B¨ack, D.B. Fogel, and Z. Michalewicz, editors. Handbook of Evolutionary Computation. Oxford University Press, 1997.
[6] A. Morales Reyes, Fault tolerant and dynamic evolutionary optimization engines, PHD thesis, The University of Edinburgh, 2011.
[7] M. Giacobini, M. Tomassini, A.G.B. Tettamanzi, and E. Alba. Selection intensity in cellular evolutionary algorithms for regular lattices. Evolutionary Computation IEEE Transactions on, October 2005, 9(5):pp. 489–505.
[8] A. Al-Naqi, A.T. Erdogan, T. Arslan, Fault tolerant three-dimensional cellular genetic algorithms with adaptive migration schemes, in: NASA/ESA Conference on Adaptive Hardware and Systems, AHS, 2011, pp. 352–359.
[9] E. Alba, B. Dorronsoro, The exploration/exploitation tradeoff in dynamic cellular genetic algorithms, IEEE Trans. Evol. Comput. 2005, 9 (2), pp.126–142.
[10] A. Al-Naqi, A.T. Erdogan, T. Arslan, Balancing exploration and exploitation in adaptive three-dimensional cellular genetic algorithm via probabilistic selection operator, in: NASA/ESA Conference on Adaptive Hardware and Systems, AHS, 2010, pp. 258–264.
[11] E. Alba and J.M. Troya. Cellular evolutionary algorithms: Evaluating the influence of ratio. In Parallel Problem Solving from Nature PPSN VI, 2000, pp. 29-38. Springer.
[12] de Lima, F.G., E. de Qualificação, and R.A. da Luz Reis, Single event upset mitigation techniques for programmable devices. Porto Alegre, 2000. 14.
[13] R. Velazco, R. Ecoffet, F. Faure, How to characterize the problem of SEU in processors & representative errors observed on flight, in: IEEE IOLTS, 2005, pp. 303–308.
[14] J. F. Ziegler and W. A. Lanford, "Effect of Cosmic Rays on Computer Memories", Science, 1979. 206(4420): p. 776-788.
[15] J. Xu, T. Arslan, Q. Wang, and D. Wan. An EHW architecture for real-time GPS attitude determination based on parallel genetic algorithm. in Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on. 2002. IEEE.
[16] Stefatos, E.F. and T. Arslan. High-performance adaptive GPS attitude determination VLSI architecture. in Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on. 2004.
[17] C. Ortega-Sanchez, D. Mange, S. Smith, A. Tyrrell , Embryonic: A bio-inspired cellular architecture with fault-tolerant properties, Genetic Programming and Evolvable Machines, 2000,1(3): p. 187-215.
[18] A. Morales-Reyes, E.F. Stefatos, A.T. Erdogan, T. Arslan, Fault tolerant cellular genetic algorithm, in: IEEE Congress on Evolutionary Computation, CEC, 2008, p. 2676–2682.
[19] A. Morales-Reyes, A.T. Erdogan, T. Arslan, E.F. Stefatos, Towards fault tolerant systems based on cellular genetic algorithms, in: IEEE NASA/ESA Conference on Adaptive and Hardware and Systems, AHS, 2008, pp. 398–405.
[20] A. Morales-Reyes, A.T. Erdogan, T. Arslan, E.F. Stefatos. Towards fault-tolerant systems based on adaptive cellular genetic algorithms. in Adaptive Hardware and Systems, 2008. AHS'08. NASA/ESA Conference on. 2008. IEEE.
[21] A. Al-Naqi, A.T. Erdogan, T. Arslan, Fault tolerance through automatic cell isolation using three-dimensional cellular genetic algorithms, in: IEEE Congress on Evolutionary Computation, CEC, 2010, pp. 1–8.
[22] A. Al-Naqi, A.T. Erdogan, T. Arslan, Dynamic Fault-Tolerant three-dimensional cellular genetic algorithms. Journal of Parallel and Distributed Computing, 2013. 73(2): pp. 122-136.
[23] E. Normand, Single event upset at ground, IEEE transactions on Nuclear Science , 1996. 43(6): pp. 2742-2750.
[24] D. Lima, F. Gusmão, D. Qualificação, E. and D. Luz Reis, R. Augusto, Single event upset mitigation techniques for programmable devices, Porto Alegre, , 2000. 14.
[25] JA. Zoutendyk, LD. Edmonds and LS. Smith, Characterization of multiple-bit errors from single-ion trackcs in integrated circuits. Nuclear Science, IEEE Transactions on, 1989. 36(6): pp. 2267-2274.
[26] G. Rémi, Single Event Effects: Mechanisms and Classification, in Soft Errors in Modern Electronic Systems. 2011, Springer. pp. 27-54.
[27] E. Alba, B. Dorronsoro, Cellular Genetic Algorithms, Springer Sciences, 2009, Vol. 42.
[28] T.V. Ramabadran and S.S. Gaitonde, "A Tutorial on CRC Computations," IEEE Micro, Aug. 1988.
[29] W.W. Peterson and D.T. Brown, "Cyclic Codes for Error Detection," Proc. IRE, Jan. 1961.
[30] W. Stallings, Data and Computer Communications. Prentice Hall, 2000.
[31] N.R. Saxena and E.J. McCluskey, "Analysis of Checksums, Extended Precision Checksums and Cyclic Redundancy Checks," IEEE Trans. Computers, July 1990.

[32] S. Shukla, NW. Bergmann N W, Single bit error correction implementation in CRC-16 on FPGA. In: IEEE International Conference on Field-Programmable Technology. Brisbane, Australia, 2004: pp. 319-322.

[33] CA. Johnston, HJ. Chao, The ATM layer chip: An ASIC for B-ISDN applications. IEEE Journal on Selected Areas in Communications, 1991, 9(5):pp. 741-750.

[34] OO. Khalifa, MR. Islam, S. Khan, Cyclic redundancy encoder for error detection in communication channels. In: RF and Microwave Conference. Selangor, Malaysia, 2004: pp. 224-226.

[35] B. Forouzan, Data Communications & Networking, McGraw-Hill, 4th Edition, 2007.

[36] U. Nordqvist, T. Henriksson, D Liu, CRC generation for protocol processing, in proceedings of NORCHIP, 2000, p. 288-293.

[37] W.W. Peterson and D. T. Brown "Cyclic Codes for Error Detection", Proc. IRE, Jan 1961, pp. 228-235.

[38] J.R. Engdahl, D. Chung, Fast parallel CRC implementation in software. In Control, Automation and Systems (ICCAS), 2014 14th International Conference on. IEEE.2014,pp.546-550.

[39] F. Monteiro, A. Dandache, A. M'sir, A fast CRC implementation on FPGA using a pipelined architecture for the polynomial division. Electronics, Circuits and Systems, 2001. ICECS 2001. The 8th IEEE International Conference on, 2001, vol. 3, pp. 1231-1234.

[40] G Campobello, G Patane and M. Russo, Parallel CRC realization. Computers, IEEE Transactions on, 2003. 52(10): p. 1312-1319.

[41] M.Molga, and C. Smutnicki, Test functions for optimization needs, 2005:http://www.robertmarks.org/Classes/ENGR5358/Papers/functions.pdf.

[42] H. Pohlheim, Examples of objective functions. Retrieved,2007.4(10):http://www.geatbx.com/download/GEATbx_ObjFunExpl_v37 .pdf.