Multiple-Fault Tolerant Hardware Structure for Cellular Genetic Algorithm

Peyman Ashoorian, Yasser Baleghi Damavandi

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

ARTICLE INFO

Article history:
Received: 25 May 2015
Revised: 26 February 2016
Accepted: 07 October 2016
Published Online: 25 January 2017

Keywords:

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 a multiple-fault situation. Here, multiple-fault refers to the situation that SEU (single event upset) occurs simultaneously at two or more bits of the chromosome and fitness registers. The fault model includes simultaneous bit inversion in chromosome strings and the worst case stuck faults in fitness registers. The main idea of the proposed approach is to control the trade-off between exploration and exploitation in fault recovery phase. The achievements of this experiment are novel recovery strategy due to applying CRC encoding and a new scheme in connections of processing elements. In order to show valid results, 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 MFT-cGA are evaluated. To illustrate the achieved immunity and improvement, the proposed MFT-cGA is compared with the 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.

© 2016 JComSec. All rights reserved.

1 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 processes found in nature [3–5]. Evolutionary algorithms are intrinsically parallel and suitable for hardware implementation.

Serial and parallel implementations are two versions of Evolutionary Algorithms that have been applied to a set of individuals (population), where each individual represents a possible solution to the problem. When a serial version is considered, the evolutionary process 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 the 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 trade-off 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 the considerable reduction in the size of transistors and the use of new materials and system on chip (SOC) architectures continue to increase the sensitivity of a system to soft errors [12]. Single event upset (SEU) is a subclass of soft error. SEUs are the main part (80%) of radiation effects in the space [13]. After SEU was discovered in space in 1975, Ziegler 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 applications in optimization however, in this paper; we focused on applications of genetic algorithm in space such as evolvable hardware [15,10]. Since in these applications the immunity of Genetic algorithm is important and much impressive, the goal of this paper is to design 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 the genetic algorithm is targeted.

Previous works in fault tolerant cellular genetic algorithm (FT-cGA) field can be categorized into two groups where each one has a distinct fault model. The first model is, the single event upset (SEE) that occurs at registers that store the scores of chromosomes. The next model, is the SEE incurring at fitness score register. Reyes. et al [15] designed fault tolerant cellular genetic algorithm using the basic competency of this algorithm. In this reference, fault tolerance characteristic was obtained with the appropriate exploitation of basic parameters and genetic operators in cGAs. The fault model in this method is the occurrence of a single hard error (SHE) at fitness score registers. Moreover, the use of migration operator and controlled selection intensity, are important gadgets in the 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 the chromosome register is caused by the occurrence of SHE errors. In [20], cellular Genetic algorithm and the distributed one are merged to one structure. SHEs affect the fitness register as the fault model of 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 to up to 40% of processing elements of fitness registers.

A recent work [23] reports a comparison between 2D and 3D implementations of the cellular genetic algorithm. The proposed architectures were subjected to single stuck faults and the simulation results showed a mixed dependency of the performance to the architecture dimensions, fault percentage, benchmark function and population size. However the mentioned work is in the simulation phase and does not contain any synthesis reports.

In this study the VHDL implementation of fault tolerant cellular genetic algorithm is targeted and a synthesis report containing hardware specifications of each block is given. Another important contribution of this paper is Multiple Fault analysis. In this work both chromosome and fitness registers are prone to multiple stuck faults. Consequently, the proposed design has been developed due to the increased probability of SEU phenomenon that may cause multiple fault situations. The original idea of this work refers to the problem of multiple faults. To resolve this issue, applying fault tolerant codes to chromosome strings is suggested. In this paper CRC code is used to immunize the process of CGA with the presence of multiple faulty chromosomes in the neighboring cells.

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 method to provide self-repairing property for all processing elements.

This paper includes five chapters. The four remaining chapters are organized as follows: Section 2 presents the fundamental definitions of the proposed method. The recommended architecture is discussed in Section 3. Section 4 presents the simulation setup, results and the discussion. The paper concludes in Section 5.

2 Fundamental Descriptions

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

2.1 Fault Tolerance

The digital circuits located in the space environment and at ground level [24] 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 the radiation environment. Dense devices need less significant feature size; this means that the data is stored with a smaller amount of charge or current. Each of these properties makes 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 memory cells. In contrast, Hard SEEs have a stable change to the operation of a device. Soft errors called Single Event Upsets (SEU) change the state of a memory cell. An SEU may occur in analog, digital or 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 to the surface of the die, concurrently attacks two sensitive junctions by direct ionization or atomic eject [26, 27]. Single hard error is similar to SEU but the memory cell gets 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 or a combinational of both strategies. In hardware strategy, the mitigation operation is fast and the main drawback of this approach is the area overhead. Moreover, this strategy cannot restrain all types of random and multiple bit errors.

Since SEE has functional effects by spreading possibly through all system modules, developing SEE tolerant systems is nowadays supported from a functional rather than a physical perspective.

2.2 Cellular Genetic Algorithm

Cellular Genetic Algorithms (cGA) are a subclass of evolutionary algorithm consisted 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 has a linear, square, or rectangular geometric shape. Each cell is called processing element (PE) and the evolutionary process is performed in each processing element. Cells of lattice interact with other local neighborhood cells [28].

Algorithm 1 shows the pseudo-code of the canonical cGA. A cGA starts with a random population; then, a fitness value is 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 reached.

2.3 Cyclic Redundancy Check (CRC)

Many communication systems use the cyclic redundancy check (CRC) method to protect important data from errors [29–32]. CRC has been widely used in digital communications because of its performance, simplicity and low usage of software and hardware resources [33–35]. In this study, the error detection ability of CRC is used but this method in some cases can also be used for error correcting applications [33].

In CRC encoding, a 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 generated n-bit result is sent to the generator. The generator uses a predefined and agreed upon divisor of size n – k + 1. 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 Figure 1), the data word (M) is shifted n – k bits to the left and is divided by a predefined divisor (P) and then is added to the remainder of this division (T = 2^n-k M + R). In this case, the addition and the subtraction are the same. We use the XOR operation to do both [36].

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

Figure 1. Principle of CRC Algorithm.
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 (neighbors);
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

Otherwise, everything is discarded.

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

- All single bit errors
- 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 [37].

One of the benefits of a cyclic code is that the hardware implementation of the encoder and the decoder is done using minimal number of electronic devices. Linear Feedback Shift Register (LFSR) with serial data feed [30] has been used to implement the CRC algorithm (see Figure 2). As all hardware implementations, this method directly performs a division and then the remainder, 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 the data words are 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, 40], in this study the serial version of CRC method is utilized with ultimate architecture [36, 37].

3 Proposed Architecture

This section discusses the proposed architecture that has the ability to deal with SEUs. In this approach, it is assumed that the SEU attacks the fitness and chromosome registers of the processing element at the same time. This section also describes the various modules of the Multiple Fault Tolerant cellular Genetic Algorithm (MFT-cGA) system that is shown in Figure 3. These MFT-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 the proposed architecture’s sub-modules.

3.1 Processing Element (PE)

In a cGA, the population is usually arranged in an \( n \)-dimensional grid of PEs. The boundary of each PE of the grid is connected to the PE located in the opposite borders in the same row/column, depending on the case. In fact, all of 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 determines the cell resistance is processing element (PE). The suitable guideline and schemas in this module can improve the performance of the 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
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 mentioned before. When CRC finds fault in chromosome register, the recovery process is started. This fault can be equivalent to the natural mutation that occurs at chromosome register. The increase in exploration parameter can be observed due to this kind of error. Thus it can disturb the trade-off between exploitation and exploration parameters. Algorithm 2 is used to control this kind of fault and the previously mentioned trade-off. In this algorithm, the recovery process is started when the neighbors’ chromosomes are identified as faulty. After fault-detection process when each neighbor’s chromosome is assumed faulty, four candidate’s chromosomes are built from the present faulty chromosome and the fittest chromosome of the 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. After the preprocessing unit, the evaluation process of fault free neighborhood is done. PEs select the best fault-free neighborhoods as parent 2. Parent 1 (internal parent) is in the current PE. Then crossover operator (in}
In Genetic Algorithm, the initial population is usually generated randomly allowing the entire range of possible solutions (the search space). It is also possible to use some seeding technique in order to speed up the search by starting from good solutions. Similarly, in this paper, the initialization block generates random chromosomes in the first generation for all PEs.

3.3 Generation Controller

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

3.4 Comparator

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

3.5 Stopping Criteria

Genetic algorithm has three kinds of convergence conditions such as the number of generations, the number of uniquely determined schedules and algorithm runtime.

The termination criterion in this module is: “if the average fitness of the population is lower than the threshold value that is set first, the algorithm is terminated”. This threshold value is set based on the global optima point of benchmark functions.

3.6 Fault Injection

To simulate the faulty environment and test our structure in this domain we need a module that injects faults. 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 an input variable. To inject fault, the candidate generations and individuals are randomly selected at the beginning of the proposed structure process.

4 Experimental Study

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

4.1 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 keeps their fitness values stuck at logical
Algorithm 2 Pseudo-code of Preprocessing

1: procedure PREPROCESSING(cGA)
2: Neighbors ← FindNeighbors (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:  (Child1,Child2) ← Selection best individual (Child1, Child2, C7, C8);
15: end if
16: end for
17: return healthy_neighbors
18: end procedure

one or zero. Furthermore, this error can be found in every generation and 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 the 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.

4.2 Case Study

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

“Rastrigin” Function

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

\[
F_{Ras}(x) = 10n + \sum_{k=1}^{n} (x_k^2 - 10\cos(2\pi x_k)) \tag{1}
\]

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 \(O(n \log(n))\), 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 \(v = (0, \ldots, 0)\) with \(F(v) = 0\) (see Figure 5).

“Griewank” Function

The Griewank function [41, 42] is multimodal and non-separable like Rastrigin but the number of its local optima is larger. The function is defined as follows:

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

where \(n\) is the number of dimensions of the function.

This function has several local optima (Figure 6) within \([-600, 600]\). Its global minimum equal \(f(x) = 0\) and is obtainable for \(x_i = 0\), \(i = 1, \ldots, n\). Moreover, the number of minima in Griewanks grows exponentially as its number of dimensions increases.
Multiple-Fault Tolerant Hardware Structure for Cellular Genetic... — P. Ashoorian and Y. B. Damavandi

Figure 6. 2D “Griewank” Function.

Figure 7. 2D “Michalewicz” Function.

“Michalewicz” Function

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

\[
F_{mich}(x) = -\sum_{i=1}^{n} \sin(x_i) \left( \sin \left( \frac{i x_i^2}{\pi} \right) \right)^2 m
\]  

(3)

In Equation (3), the variables domain is in the \([0, \pi]\). In this equation, the \( m \) parameter defines the “steepness” of the valleys of function (Larger \( m \) leads to more difficult search). In this study, \( m \) is set to five. In Figure 7, the two-dimensional plot of this function is shown.

“Drop Wave” Function

Drop wave is a multimodal test function [40, 41]. The test area of this function is in \([-5.12, 5.12]\) for all variables. Equation (4) defines the Drop wave’s function:

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

(4)

Table 1. Benchmark Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Search Domain</th>
<th>Global Optima (X,Y)</th>
<th>Stopping Criteria (Threshold)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_Mich</td>
<td>([0, \pi])</td>
<td>(2.18328, -0.806919)</td>
<td>-0.806</td>
</tr>
<tr>
<td>F_Ras</td>
<td>([-5.12, 5.12])</td>
<td>(0, 0)</td>
<td>0.00005</td>
</tr>
<tr>
<td>F_Gri</td>
<td>([-600, 600])</td>
<td>(0, 0)</td>
<td>0.00005</td>
</tr>
<tr>
<td>F_Drop</td>
<td>([-5.12, 5.12])</td>
<td>(0, -1)</td>
<td>-0.99999</td>
</tr>
</tbody>
</table>

The two-variable version of this function in aforementioned search space is shown in Figure 8.

4.3 Simulation Setup

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

The last column of the above table shows the convergence condition of genetic algorithm in the optimization of each test function separately. There are three techniques to determine the convergence of optimization algorithms. One of them is applying the rules and when the algorithm satisfies that condition, the optimization is stopped. In this study when the average of population fitness in each generation is lower than the threshold value, the algorithm is stopped. This threshold value is shown in the last column of the above table that is selected according to global optima point of each test function.

The proposed architecture (MFT-nD-cGA) is implemented for two and three-dimensional structures of the population. The details of simulation setups are displayed in Table 2.
Table 2. Details of Simulation Setup

<table>
<thead>
<tr>
<th>Parameters</th>
<th>MFT-2D-cGA</th>
<th>MFT-3D-cGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighborhood</td>
<td>east, west, north, south</td>
<td>east, west, vertical north and south, horizontal north and south</td>
</tr>
<tr>
<td>Parent Selection</td>
<td>Best Neighborhood</td>
<td>Best Neighborhood</td>
</tr>
<tr>
<td>Recombination</td>
<td>One Point Cross Over</td>
<td>One Point Cross over</td>
</tr>
<tr>
<td>Mutation</td>
<td>Bit Flip</td>
<td>Bit Flip</td>
</tr>
<tr>
<td>Max Generation</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Chromosome Length</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Replacement</td>
<td>Replace if better</td>
<td>Replace if better</td>
</tr>
<tr>
<td>Termination Condition</td>
<td>Avg_fitness ≤ threshold value</td>
<td>Avg_fitness ≤ threshold value</td>
</tr>
<tr>
<td>CRC Length</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

4.4 Evaluation Metrics

In optimization algorithms, some evaluation metrics are needed to display the performance or superiority of different structures. 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 generations and the search success rate of successful 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 leading to low $\gamma$. Equation (5) provides the formula of $\gamma$ parameter.

$$\gamma = \frac{\text{efficiency}}{\text{efficacy}}$$ (5)

4.5 Simulation Result

In this part, diagrams are according to $\gamma$ parameter due to its generalization. Therefore, $\gamma$ metric shows the performance of the algorithm.

In Figure 9, the $\gamma$ is represented vs. population size in two and three-dimensional structures of the population in proposed architecture. Moreover, these diagrams are obtained when all of PEs are healthy and no SEU fault is occurred. In these diagrams, $\gamma$ 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 the performance of the proposed architecture against each test function based on the given $\gamma$ parameters in the mentioned diagrams.

Figure 10 shows the influence of fault percentage on the proposed architecture. In this figure, the effect of fault percentage in ratio parameter ($\gamma$) for three separate population size and two distinct structures are shown (these diagrams are acquired for “Drop Wave” test function).

Figure 11 makes 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 the canonical method is obvious in these figures.
Table 3. ISE Design Properties

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-Level Source Type</td>
<td>HDL</td>
</tr>
<tr>
<td>Family</td>
<td>Artix 7</td>
</tr>
<tr>
<td>Device</td>
<td>XC7A100T</td>
</tr>
<tr>
<td>Package</td>
<td>CSG324</td>
</tr>
<tr>
<td>Speed</td>
<td>-2L</td>
</tr>
<tr>
<td>Synthesis Tool</td>
<td>XST (VHDL/Verilog)</td>
</tr>
<tr>
<td>Simulator</td>
<td>ISim (VHDL/Verilog)</td>
</tr>
<tr>
<td>Preferred Language</td>
<td>VHDL</td>
</tr>
<tr>
<td>VHDL Source Analysis Standard</td>
<td>VHDL-200X</td>
</tr>
</tbody>
</table>

Table 4. MFT-cGA Synthesis Setups

<table>
<thead>
<tr>
<th>Parameters Name</th>
<th>Sets Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology Dimension</td>
<td>3D</td>
</tr>
<tr>
<td>Topology Size (2 x 2 x 2)</td>
<td></td>
</tr>
<tr>
<td>Chromosome Length</td>
<td>5 bits</td>
</tr>
<tr>
<td>CRC Length</td>
<td>4 bits</td>
</tr>
<tr>
<td>CRC Polynomial Vector</td>
<td>“1101”</td>
</tr>
<tr>
<td>Optimization Function</td>
<td>$Y = x^2$</td>
</tr>
<tr>
<td>Search Area ($[x_{min}, x_{max}]$)</td>
<td>$[-5.12, 5.12] \in R$</td>
</tr>
<tr>
<td>Max Number Of Generation</td>
<td>100</td>
</tr>
<tr>
<td>Floating Point Size</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

4.6 Synthesis Result

In this part, the result of the hardware implementation of proposed structure (MFT-cGA) in ISE Design Suite 14.2 and the simulation results in MODESIM 6.3 (illustrated in Appendix A) are presented. The ISE design properties are shown in Table 3.

In Table 3, the Family, Device and the package of FPGA and the language of hardware implementation are shown.

Due to the implementation restrictions, the proposed structure of the genetic algorithm was used with decreased parameters. Therefore, based on this fact the simulations parameters that are synthesized in ISE are arranged in Table 4.

Finally, the detailed synthesis reports of the utilization of hardware components in FPGA are organized in Table 5.

4.7 Discussion

In Figure 9, it can be seen that (Except the results for function Michalewicz) the $\gamma$ parameter is increased...
Table 5. Component Synthesis Reports

<table>
<thead>
<tr>
<th>Parameter/ Component</th>
<th>Initialization</th>
<th>Generation Controller</th>
<th>Comparator</th>
<th>Stopping Criteria</th>
<th>Processing Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Registers</td>
<td>0</td>
<td>9</td>
<td>78</td>
<td>36</td>
<td>398</td>
</tr>
<tr>
<td>Number of Slice LUTs</td>
<td>1</td>
<td>22</td>
<td>133</td>
<td>75</td>
<td>14646</td>
</tr>
<tr>
<td>Number Used as Logic</td>
<td>1</td>
<td>22</td>
<td>132</td>
<td>74</td>
<td>14583</td>
</tr>
<tr>
<td>Number of LUT Flip Flop pairs used</td>
<td>1</td>
<td>22</td>
<td>134</td>
<td>77</td>
<td>14680</td>
</tr>
<tr>
<td>Maximum Frequency</td>
<td>284.216 MHZ</td>
<td>209.312 MHZ</td>
<td>204.732 MHZ</td>
<td>15.744 MHZ</td>
<td></td>
</tr>
<tr>
<td>Total Memory Usage</td>
<td>452288 kb</td>
<td>441728 kb</td>
<td>443904 kb</td>
<td>453312 kb</td>
<td>917632 kb</td>
</tr>
<tr>
<td>Minimum Input Arrival Time Before Clock</td>
<td>1.366 ns</td>
<td>1.400 ns</td>
<td>3.858 ns</td>
<td>1.451 ns</td>
<td>64.937 ns</td>
</tr>
<tr>
<td>Maximum Output Required Time After Clock</td>
<td>0.737 ns</td>
<td>0.742 ns</td>
<td>0.742 ns</td>
<td>0.742 n</td>
<td>1.564 n</td>
</tr>
<tr>
<td>Delay</td>
<td>3.518 ns</td>
<td>4.778 ns</td>
<td>4.884 ns</td>
<td>61.625 ns</td>
<td></td>
</tr>
</tbody>
</table>

due to increase in the size of the population. In the structure with a bigger population, more effort is needed to reduce the average number of generation under the defined threshold so the average number of generation for the bigger population is increased.

In the proposed structure, SEU errors on chromosome register can be detected by CRC method. The best individual of the previous generation and the faulty chromosome are sent to Algorithm 2 to produce a healthy and graceful chromosome which replace the faulty one. Therefore, we can see in Figure 10 that an increase in error percentage reduces $\gamma$ parameter. In other word, the proposed algorithm used the best individual of the previous generation and faulty chromosome as a natural mutant individual to enhance the mutation and crossover percentage and improve the trade-off between exploration and exploitation of algorithm. Consequently, the proposed structure has a good performance in the faulty environment.

It can be seen in Figure 11 when the fault percentage in radiation environment is grown, the $\gamma$ parameter of the proposed structure decreases (the algorithm converges faster) but in canonical version of cGA (Figure 11a) the enhanced fault percentage has a direct effect on $\gamma$ parameter and the canonical version has not a good performance and in some cases it may not be able to reach the optimum solution. In addition, this figure shows that the 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 6 the two and three-dimensional topology of the proposed structure based on mentioned parameters are compared. According to this table, the three-dimensional structure is superior, because good solutions can spread faster in population due to the large number of neighborhoods.

5 Conclusion

This study proposed a VHDL implementation of a family of MFT-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.

The idea behind this study was to recuperate the immunity and performance of MFT-cGA by controlling the trade-off between exploration and exploitation (detect the fault and recover based Algorithm 2). Also, a comparison is given for MFT-cGA performance when two and three-dimensional topologies of the structure are used to solve the hard optimization functions in different fault percentages. The CRC method is used to detect the fault and to enable the recovery process. The recovery process helped to control the exploration/exploitation trade-off by using the crossover and mutation operators in recovery step (Algorithm 2). On the other hand, when fault occurrence enhances the exploration parameter, the recovery tools fix the exploration/exploitation trade-off. Furthermore, to prevent the fault spread effect in a population due to fitness failure, the fitness values of neighboring PEs were not used.

To evaluate the improvements, proposed FT-cGA was compared with canonical-cGA in terms of $\gamma$ parameter (efficiency, efficacy). The canonical version of cGA is intrinsically immune but the fault percentage affects its performance. This problem is solved in the 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 reparative method for immunity against fault, the approach
Table 6. Comparison of Two Structures of FT-cGAs in Ratio Parameter for All Test Functions, Different Sizes of Population and Fault Percentages

<table>
<thead>
<tr>
<th>Function</th>
<th>Michalewicz</th>
<th>Rastrigin</th>
<th>Griewank</th>
<th>Drop Wave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensional Fault</td>
<td>2D</td>
<td>3D</td>
<td>2D</td>
<td>3D</td>
</tr>
<tr>
<td>0%</td>
<td>Size &lt; 40</td>
<td>Size &gt; 40</td>
<td>Size &lt; 484,</td>
<td>Size &gt; 484,</td>
</tr>
<tr>
<td>10%</td>
<td>✓</td>
<td>✓</td>
<td>Size &lt; 78</td>
<td>Size &gt; 78</td>
</tr>
<tr>
<td>30%</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>50%</td>
<td>Size &lt; 123,</td>
<td>Size &gt; 189,</td>
<td>Size ∈ [123, 189]</td>
<td>Size ∈ [123, 189]</td>
</tr>
<tr>
<td>70%</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>100%</td>
<td>Size &gt; 115</td>
<td>Size &lt; 115</td>
<td>Size &lt; 249</td>
<td>Size &lt; 249</td>
</tr>
</tbody>
</table>

✓: The selected dimension is suitable for optimization.
×: The selected dimension is not suitable for optimization.

offers considerable improvements in $\gamma$ parameter and reliability of the algorithms, especially for cases with a high ratio of faults.

In experimental results, we noted 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, the algorithm does not have much time overhead. Thus, this structure is suitable for real time application. Finally, Table 6 illustrates which topology of proposed MFT-cGA (2D or 3D) is better in a variety of conditions (population size, fault percentage occurred for optimization of all test functions).

References


A Simulation Waveform Results in MODELSIM

In this part, the Timing Simulation waveform of the implementation of Multiple Fault Tolerant Cellular Genetic Algorithm in MODELSIM is shown in two figures. In Figure A.1 the two-dimensional structure of MFT-cGA in 100th run of Rastrigin function in size 3*3 of cellular shapes in fault percentage of 0% to 50% is illustrated. Figure A.2 is similar to group (a) but the fault percentage is different. In Figure A.2 the fault percentage is between 70% and 90%.

Peyman Ashoorian was born in 1990 in Tonekabon (Iran). He received his B.Sc. degree in Hardware Computer engineering in 2004 from Babol Noshirvani University of Technology, Babol, Iran. He then worked toward the M.Sc. degree in the same university in Electronic Engineering and graduated in 2014. His research interests include the field of evolvable and adaptive hardware and fault tolerant system design.

Yasser Baleghi Damavandi is an Assistant Professor of Electronic Engineering at Babol Noshirvani University of Technology. He holds a Ph.D. degree in Electronic Engineering from Iran University of Science & Technology. His research interests are evolvable and adaptive hardware, image processing and fault tolerant system design.
Figure A.1. Output waveforms of two-dimensional structure of MFT-cGA in 100th run in fault range of 0% to 50%
Figure A.2. Output waveforms of two-dimensional structure of MFT-cGA in 100th run in fault range of 70% to 90%