# Application of a Genetic Algorithm in a Fault-tolerant Filter

Mónica Lovay[1], Gabriela Peretti[1,2], Eduardo Romero[1,2], Carlos Marqués[2]

[1] Mechatronics Research Group, Facultad Regional Villa María, Universidad Tecnológica Nacional, Avda. Universidad 450, 5900 Villa María, Argentina
gecam@frvm.utn.edu.ar
[2] Electronics and Instrumentation Development Group, Universidad Nacional de Córdoba, Facultad de Matemática, Astronomía y Física, Medina Allende S/N, 5000 Córdoba, Argentina
marques@famaf.unc.edu.ar

**Abstract.** This paper presents an eighth order low-pass filter which has characteristics of fault tolerance through the use of evolvable hardware (EHW). A field programmable analog array (FPAA) is used to implement the filter under study. The reconfiguration process of the filter involves the execution of a genetic algorithm (GA) in an external computer, after a fault is detected. To perform the test of the filter, we assume that a frequency response characterization test is used. A parametric fault model that considers deviations in the values of one of the capacitors or one of the input amplifiers (IA) is used to evaluate the performance of developed GA. The results show that GA finds filter configurations that meet the restrictions set for all the simulated faults. Additionally, this work shows better results compared to those previously obtained using another EHW scheme for the same low-pass filter.

**Keywords:** Evolvable hardware, field programmable analog array, fault-tolerant filter, genetic algorithm, multi-objective optimization.

## 1 Introduction

Fault tolerance is a desirable characteristic for most electronics systems, especially for those operating in harsh environments with maintenance difficult to achieve.

Evolvable hardware (EHW) is a methodology that consists in the application of genetic algorithm (GA) to hardware [6], [10], [17]. When used with reconfigurable hardware, the designer establishes performance goals and GA searches the possible hardware configurations that present the better performances, even under the presence of faults. In this way, EHW offers an alternative to traditional fault-tolerant schemes [8]. Additionally, although reconfiguration does not always guarantee that complete functionality can be restored, it allows gracefully degradation [9] and is an alternative for systems with limited free space [7].

Among others, different EHW fault-tolerance schemes have been proposed for field programmable gate arrays [2], field programmable analog arrays (FPAAs) [9], and programmable system on chip [11-12].

In this work, we address the problem of providing fault tolerance to an eighth-order low-pass filter implemented in a field programmable analog array (FPAA), employing an EHW approach. The presented scheme is useful when the system presents a topology with a host computer interacting with remote measurement units deployed in field, like wireless sensor networks.

For implementing the EHW scheme, a test embedded in the system detects when the filter fails its specifications. This test establishes the filter frequency response by using a multitone signal [16]. Then a GA, running in an external computer, finds new configurable parameters values of the filter that meet pre-established specifications. The evolved values are loaded into the hardware for continuing the normal operation.

In the EHW scheme proposed here GA performs a very important role, because it has to solve a multi-objective optimization problem to find the new system settings in a very large search space. For this reason, this work focuses in the evaluation of the performance of the proposed GA, in normal and faulty conditions.

It should be noted a previously EHW scheme has been proposed for the filter under study in [13]. However, the presented here considers most demanding performance specifications and a different test strategy. The use of the frequency response test method requires the definition of a new fitness function in the GA, which now has less information to qualify individuals and guide the search. Additionally, the evaluation under fault conditions is more complete because new failures modes are contemplated. These new requirements cause the development of a new GA.

## 2 Fault-tolerant Filter

To implement the filter under study we adopt the FPAA device ispPAC10 from Lattice Semiconductor [4], [13]. The FPAA consists or four programmable analog cells interconnected with programmable switching networks. The user can select independent gains for the input amplifiers (IA), and can control the bandwidth of the output amplifiers (OA) by setting their feedback capacitor values (CF). Each capacitor adopts 128 possible values, from 1.07pF to 61.59pF. The gain values of the IA can be programmed from -10 to +10 in steps of 1.

The design is performed using Pac Designer [14], which programs the connections using the internal resources and sets the corresponding value for the programmable components. By means of this tool, we design an eight-order low-pass filter, with DC gain equal to one, as a cascaded of four biquadratic sections. The topology of one of the biquadratic section can be seen in Fig. 1.

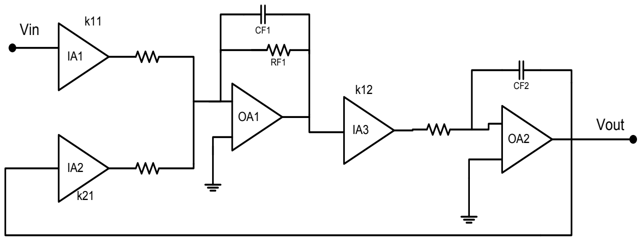


**Fig. 1**. Implementation of a biquadratic section in the device ispPAC10.

The transfer function of a biquadratic section with low pass filter characteristics is:

$$H_{LP}(s) = \frac{Vout}{Vin} = \frac{\frac{k11 \cdot k12}{(CF1 \cdot 250k\Omega) \cdot (CF2 \cdot 250k\Omega)}}{s^2 + \frac{s}{(CF1 \cdot 250k\Omega)} - \frac{k12 \cdot k21}{(CF1 \cdot 250k\Omega) \cdot (CF2 \cdot 250k\Omega)}} \quad \textbf{(1)}$$

In (1), *k11, k12* and *k21* are programmable gains for each IA (Fig. 1). The *CF* capacitors correspond to the feedback capacitors and the resistors of 250kΩ represent the input resistance of each IA (not shown in Fig. 1).

An expression for the transfer function (*Hy(s)*) of an eight-order filter is:

$$H_y(s) = \prod_{i=1}^{4} \left( \frac{\frac{k11_i k12_i}{(CF1_i 250k\Omega) \cdot (CF2_i 250k\Omega)}}{s^2 + \frac{s}{(CF1_i 250k\Omega)} - \frac{k12_i k21_i}{(CF1_i 250k\Omega) \cdot (CF2_i 250k\Omega)}} \right) \quad \textbf{(2)}$$

In (2), *i* indicates the biquadratic stage. According to exp. (2), each filter implementation with $H_y(s)$ transfer function requires to configure 8 capacitors ($CF1_i$ and $CF2_i$) and 12 gains ($k11_i$, $k12_i$ and $k21_i$) in the FPAA device.

It should be noted that the biquadratic implementation restricts the values that can adopt the gains of the IA, in order to reach a stable implementation. For this reason, is considered that $k11_i$ and $k12_i$ can be programmed from -10 to -1, and $k21_i$ can be programmed from 1 to 10.

Fig. 2 shows the frequency response of the filter taken as a case study, denominated $H_n(j2\pi f)$, which is the so-called nominal response. This characteristic has to be maintained during in-field operation. That is, the filter response has to be maintained within tolerance limits, despite the presence of faults.

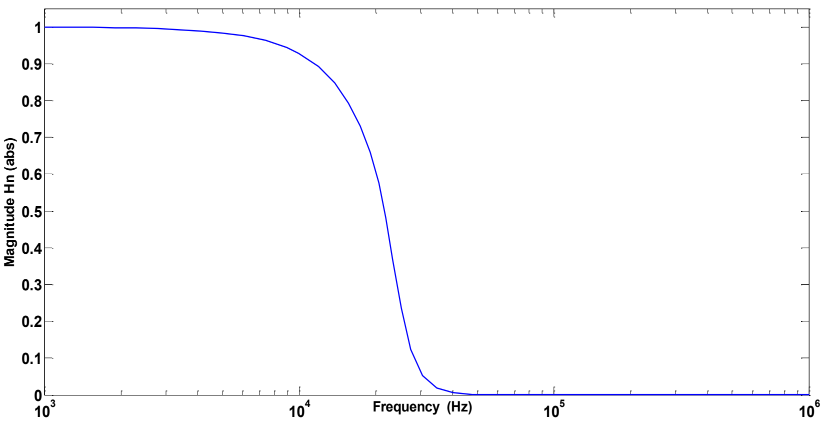


**Fig. 2**. Nominal response of the designed filter ($H_n$).

To perform the test of the filter, we assume that a frequency response characterization test is used. In this test, the filter frequency response is evaluated by measuring the gain of the circuit as the input signal frequency is varied [16]. This test requires the generation of a multitone periodic signal. The frequency of the tones is chosen at specified points in the pass band, the cutoff point, the transition and the attenuation band, to be specified in section 4.2. The frequency response is calculated by performing a Fourier analysis (in the host computer) on the waveforms collected at the filter input and output. As the feasibility of this test was previously demonstrated, in this work we assume that the filter gains at the specified points are available after the test routine. These gains are used as input for establishing the fitness of the filter.

# 4 Genetic Algorithm

GA finds the gain values of the IA (*k11*$_i$, *k12*$_i$ and *k21*$_i$) and the values of the capacitors (*CF1*$_i$ and *CF2*$_i$) for each biquadratic stage, with the goal of maintaining the filter response $H_y(s)$ within specifications. The GA has to solve a multi-objective optimization problem, with an overall search space of 7.21E+28 different filter implementations.

## 4.1 GA Implementation

GA is a search and optimization technique based on the principles of genetics and natural selection [6], [15], [18-19]. Fig. 3 shows a flowchart of the developed GA. For the sake of clarity, in the following we explain each block of the flowchart.

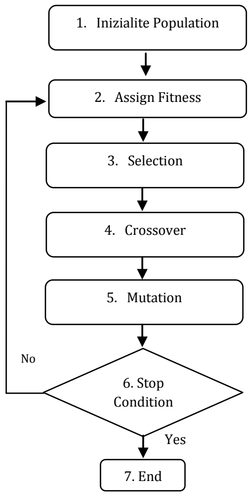


**Fig. 3.** Flowchart of a traditional GA.

**Block 1. Initialize Population:**
The algorithm starts by randomly generating an initial population ($pop_{in}$) of individuals that are possible solutions to the problem. In this case study, each individual represents a possible eighth-order filter configuration. The individual is characterized by the gain values of the IA (*k11*$_i$, *k12*$_i$ and *k21*$_i$) and the values of the capacitors (*CF1*$_i$ and *CF2*$_i$) of each biquadratic stage. The GA represents each individual using a chromosome consisting of a set of twenty genes. Each gene represents the value of a capacitor or the gain value of an IA corresponding to a biquadratic stage. Each gene is represented using integer coding. Figure 4 shows the chromosome structure used by the algorithm.

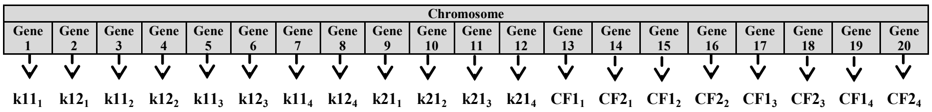


**Fig. 4.** Chromosome structure used by the GA.

**Block 2. Assign Fitness:**

For every evolutionary step, or generation, the individuals in the current population are evaluated according to some predefined quality criterion called the fitness function. Fig. 5 shows the operations performed by the GA in this block.

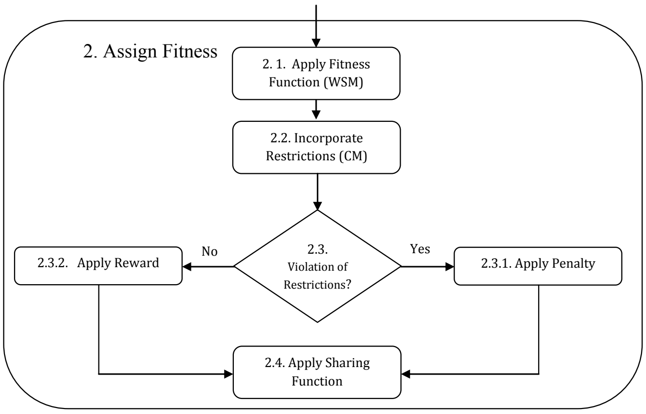


**Fig. 5.** Sub diagram flow for Block 2: "Assign Fitness".

*2.1. Apply Fitness Function*: In this case study, we consider that the most important objective is to maintain the filter response within specifications at the cutoff point. Less stringent are the specifications in the pass band, the transition band and in the attenuation band. This fact allows using the so-called a priori method to generate the fitness function, which transforms a multi-objective problem into a single-objective one. Weighted Sum Method (WSM) [1], [3], [5], [19] combines the different objectives into a single one, generally in a linear way. The use of this method allows employing traditional GA, as the described in Fig. 3 and in [15], [18-19].

The GA implements the fitness function (*f*) as follows:

$$f(y) = B - \sum_{k=1}^{4} w_k \left[REM_y(f_k)\right], \tag{3}$$

where $w_k \in [0 \ldots 1]$ and $\sum_{k=1}^{4} w_k = 1$. In (3), *k* adopts the values 1, 2, 3 and 4, for the pass band, the cutoff point, the transition band and the attenuation band, respectively. The values of $w_k$ are called weights, representing the degree of importance assigned to the error in each band. $REM_y(f_k)$ is the relative error in magnitude for the individual *y* in the band *k*. The purpose of the GA is to minimize the sum of the relative errors proposed. However, the fitness function (*f*) is formulated in (3) for obtaining a maximum. For this reason, a constant B is added for avoiding negative numbers. The $REM_y(f_k)$ is obtained from the following expression:

$$REM_y(f_k) = \left|\frac{|H_y(2\pi f_k)| - |H_n(2\pi f_k)|}{|H_n(2\pi f_k)|}\right|,\ k = 1,2,3,4. \tag{4}$$

In (4), $f_k$ is the frequency considered in the band *k*. If *k*=1 (pass band), $f_k$ is the frequency corresponding to the maximum *REM* obtained from three frequencies of that band. In each of the other bands is considered only one frequency.

$H_y$ is the transfer function of the individual *y* and $H_n$ is the nominal response defined as the target response. It should be noted that the values of $H_y$ are delivered by the test strategy (Section 2).

The use of the frequency response test method precludes the use of relative errors in frequency, as it was considered in [13]. This consideration involves a new challenge for the GA because it has less information to evaluate individuals and guide the search.

*2.2. Restrictions*: GA transforms all the objectives into restrictions using an a priori method, named ε-Constraint Method (CM) [3], [5]. Constraints are represented in the following expression:

$$REM_y(f_k) \leq \varepsilon_k, \quad k=1, 2, 3, 4. \tag{5}$$

In (5), the values of $\varepsilon_k$ represent the maximum tolerable error defined for each band *k*.

*2.3.1. Penalty*: In order to apply the restrictions, the algorithm penalizes the individuals with errors above $\varepsilon_k$. For these individuals, GA calculates the penalty factor *p* for each band *k* in which the error is greater than $\varepsilon_k$, according to the following expression:

$$p_k(y) = \begin{cases} \frac{REM_y(f_k)-\varepsilon_k}{\varepsilon_k} & if\ REM_y(f_k) > \varepsilon_k \\ 0 & other\ case \end{cases} \tag{6}$$

The new fitness calculation *(Lf)* considers the penalty factor ($p_k$) and the level of importance assigned to each band ($w_k$). The value *Lf* is obtained as follows:

$$Lf(y) = f(y).(1 - \sum_{k=1}^{4} w_k.p_k)] \tag{7}$$

*2.3.2. Reward*: GA increases the *f* value to the individuals that present errors less than or equal to $\varepsilon_k$ in the *three* bands. For these individuals, the algorithm calculates the awards factor *a* for each band *k*, according to the following expression:

$$a_k(y) = \frac{\varepsilon_k - REM_y(f_k)}{\varepsilon_k} \tag{8}$$

The new fitness (*Hf*) is calculated considering the awards factor $a_k$ and the level of importance assigned to each band $w_k$, as follows:

$$Hf(y) = f(y).(1 + \sum_{k=1}^{4} w_k.a_k)] \tag{9}$$

2.4. *Sharing Function*: The algorithm incorporates a strategy of diversification of the population. The aim of the function is to keep the population distributed across multiple regions of the search space to prevent premature convergence in local optima. The sharing function method [15], [19] considers that the fitness should be shared as a single resource among similar individuals in a population. This method works by degrading the fitness of each individual, according to an indicator related to the number of similar individuals in the population. Specifically, the new fitness *f '* of an individual *y* is equal to the original fitness *f* (*Lf o Hf*) divided by a counter, which contains the sum of the values of the function sharing *sh* evaluated between the individual and other individuals population:

$$f'(y) = \frac{f(y)}{\sum_{j=1}^{n} sh(d(j,y))} \quad \textbf{(10)}$$

In (10), $n$ is the population size. The sharing function $sh$ is a function of the distance between two population elements ($j$ and $y$), which returns a "1" if the elements are identical, a "0" if they cross some threshold of dissimilarity (specified by the distance constant, σ), and an intermediate value for intermediate levels of similarity. The function $sh$ is as follows:

$$sh(d(j,y)) = \begin{cases} 1 - \frac{d(j,y)}{\sigma} & if\ d(j,y) < \sigma \\ 0 & other\ case, \end{cases} \quad \textbf{(11)}$$

where $d(j,y)$ is a function that calculates the distance between the individuals $j$ and $y$, from the difference between the gain values of IA and the values of the capacitors of both individuals, in each biquadratic stage $i$, according to the following expression:

$$d(j,y) = \sum_{k=1}^{4}[(CF1i_j - CF1i_y) + (CF2i_j - CF2i_y) + (k11i_j - k11i_y) + (k12i_j - k12i_y) + (k21i_j - k21i_y)] \quad \textbf{(12)}$$

**Block 3. Selection:**

To form a new population (the next generation), individuals are selected according to their fitness; high-fitness individuals present better chances to appear (“survive”) in the next generation, while low-fitness ones are more likely to disappear.

The selection is performed through the method of the rotating roulette [6]. The probability of an individual to be selected for crossover is proportional to its fitness.

**Blocks 4 and 5. Crossover and Mutation:**

The crossover operator selects two individuals, called parents, and exchanges parts of their information to form two new individuals, called offspring. If the two parents do not undergo the crossover operation, they are copied unchanged to the new pool of individuals.

The mutation operator is applied to the new pool of individuals produced after the application of crossover. This operator prevents premature convergence to local optima. After mutation, a new generation of individuals is produced.

The developed GA uses adaptive strategies for crossover and mutation in order to perform in the first generations an exploration of the search space. The algorithm starts by applying the technique of three-point crossover (with a crossover probability $p_c$ which remains fixed through the generations) and uniform random mutation method [6], [19] (with an initial mutation probability $p_{mi}$). When GA finds at least a solution where the REM in each band approaches (in certain percentage) to maximum tolerable error ($\varepsilon_i$), it begins to apply the technique of crossover at a single point, with the purpose of making a more limited search to a subspace of possible solutions. If after a certain amount of consecutive generations GA does not found solutions that meet the established performance criteria, then uses a higher mutation probability $p_{mh}$ (only for the present generation) and starts using the crossover technique in three points.

**Block 6. Stop Condition:**

The new generation goes through the process described above, from the fitness evaluation to the mutation step. The cycle repeats until the stop condition is met, which consists in reaching a maximum number of generations *($gen_{max}$)*, or finding a solution that meets the restrictions enunciated in (5).

### 4.2 Parameters

The algorithm is implemented in Matlab, by considering the following parameters: $pop_{in}$=200, $gen_{max}$=200, $p_c$=1, $p_{mi}$=0.01, $p_{mh}$=0.4, B=100, σ=3. These values are chosen from previous experiments where different alternatives were considered for each parameter.

We set the value of the maximum tolerable error for each band as follows: $\varepsilon_1$=±10%, $\varepsilon_2$=±5%, $\varepsilon_3$=±15% and $\varepsilon_4$=±20%. Fig. 6 shows a graphical interpretation of these values. In this figure, the frequencies $f_{11}$, $f_{12}$ and $f_{13}$ correspond to the pass band, $f_2$ is the cutoff frequency, $f_3$ and $f_4$ correspond to the transition band and attenuation band respectively.

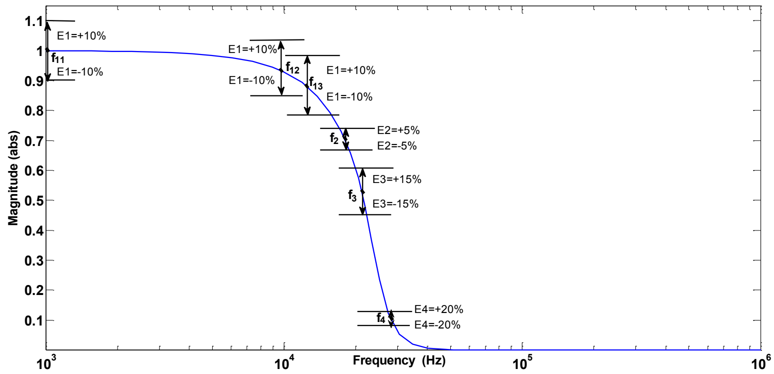


**Fig. 6.** Graphical interpretation of the values of $\varepsilon_k$ and $f_k$.

The values assigned to the weights used in (3) are $w_1$=0.4, $w_2$=0.3, $w_3$=0.15 and $w_4$=0.15. These values were obtained after performing experiments with different combinations of weights, considering the degree of importance assigned to the error in each band. The combination of weights that provides the best performance was selected.

## 5 Simulation results

### 5.1 Simulation setup

The performance of the fault tolerance scheme with GA is evaluated by means of fault injection. To implement an experiment involving fault injection and simulation are required two elements: a simulation model of the system under test and a compatible fault model. The transfer function of the biquadratic sections is adopted as simulation model (eq. 1), allowing to reduce the high computational cost of the fault injection campaigns. In the following sections the adopted fault models are presented.

The fitness assignment of each individual is performed by the interaction of the GA with the frequency response test method. In the simulation, this is achieved by the valuation of the transfer function using the gain and capacitor values corresponding to the current generation of individuals.

### 5.2 Fault Free Operation

The simulation of the fault-free operation is performed considering that the component values that represent an individual are the nominal values.

Fig. 7 shows the relative error in magnitude (*REM*) for GA in each band, in 50 runs. Each run is a solution to the optimization problem changing the seed for the random generation of the first population. In the four bands, the *REM* satisfies the performance criteria established.

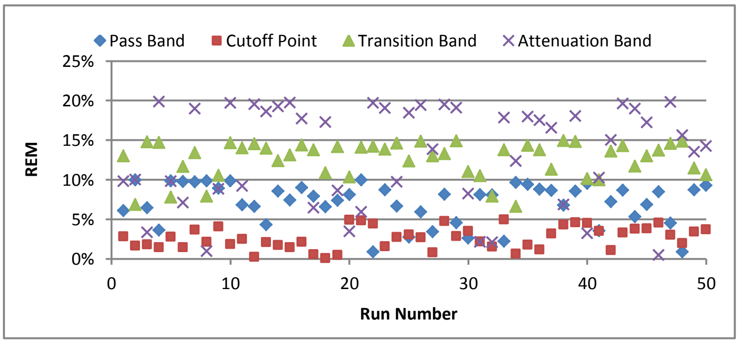


**Fig. 7**. Errors for GA. Fault-free operation

Table 1 shows a characterization of *REM* in each band. We adopt the median as a measurement of central tendency because the data distribution is not normal. We also present the maximum and minimum as a measurement of dispersion. GA maintains the maximum relative errors within the criteria established in the four bands.

On the other hand, in all cases, the algorithm completes its execution at most in 190 generations, with a median of 37 generations. That is, the algorithm always finds a good solution before reaching the established maximum number of generations.

**Table 1**. *REM* characterization under fault-free conditions.

| Error | Cutoff Point | Pass Band | Transition Band | Attenuation Band |
|---|---|---|---|---|
| Median (%) | 8.00 | 2.73 | 13.51 | 15.33 |
| Minimum (%) | 0.88 | 0.09 | 6.63 | 0.48 |
| Maximum (%) | 9.98 | 4.98 | 14.96 | 19.87 |

### 5.2 Operation under fault condition

The simulation of the operation under fault conditions is performed by altering the values of the components according to two different models. The first considers that there is a deviation in one of the capacitor banks; the second assumes a deviation in one of the AIs. For each faulty component, we consider deviations in a percentage of its nominal value, ± 10%, ±20%, ±30% and ±40%.

*5.2.1 Faults in capacitors:*

For space reasons, we only report in Fig. 8 the fault injection results corresponding to one of the capacitor banks of the second biquadratic section, which presents the worst results. The figure shows the *REM* in each band versus the percent deviation in the capacitor. From the simulation results, it is observed that GA is able to maintain system performance within specifications.

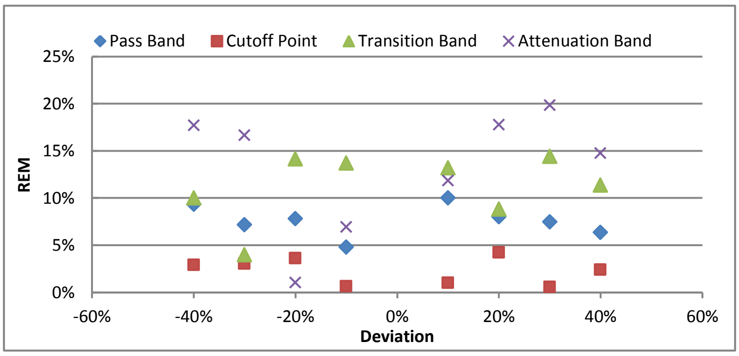


**Fig. 8**. Errors for GA. Fault condition in a capacitor.

Table 2 summarizes the effects of deviation faults. Comparing the normal (Table 1) and deviation fault conditions (Table 2), the faulty system presents a decrease in the median. The maximum *REM* is almost constant in all bands, while the minimum *REM* increases in the pass band and in the attenuation band, decreasing in the other two bands.

Regarding to the number of generations, the algorithm presents a slight increase in the maximum number of generations reached (197) and in the median (53), with respect to normal operation.

**Table 2**. *REM* characterization of GA under fault condition in a capacitor.

| Error | Cutoff Point | Pass Band | Transition Band | Attenuation Band |
|---|---|---|---|---|
| Median (%) | 6.83 | 2.69 | 13.14 | 14.58 |
| Minimum (%) | 2.55 | 0.01 | 0.30 | 0.79 |
| Maximum (%) | 9.99 | 4.94 | 14.98 | 19.83 |

*5.2.2 Faults in IA:* Fig. 9 depict the results obtained under fault condition in one of the IA corresponding to the third biquadratic section. We report only these results because this IA presents the worst performance in the fault injection process. The figure shows that the *REM* meets the performance criteria established in the four bands.

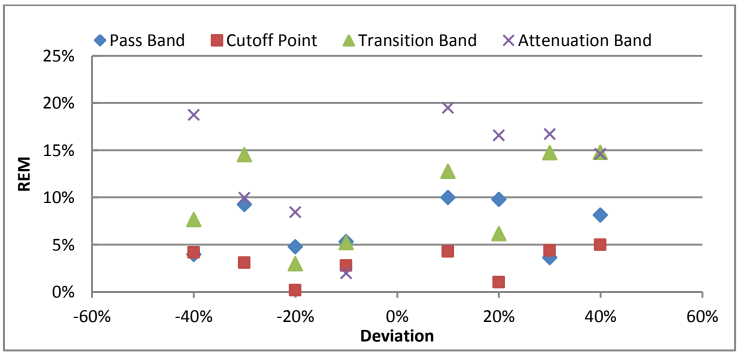


**Fig. 9**. Errors for GA. Fault condition in a IA.

Table 3 summarizes the effects of deviation faults in IA. Comparing the normal (Table 1) and deviation fault conditions (Table 3), the faulty system presents a maximum *REM* that is almost constant in all bands, except at the attenuation band where it slightly increases. The *REM* increases in the pass band, while decreases in the other bands. The median value increases in the cutoff point and the attenuation band, while diminishes in the other two bands.

**Table 3**. REM characterization of GA under fault condition in a IA.

| Error | Cutoff Point | Pass Band | Transition Band | Attenuation Band |
|---|---|---|---|---|
| Median (%) | 6.92 | 2.93 | 13.15 | 16.56 |
| Minimum (%) | 2.28 | 0.04 | 0.92 | 0.03 |
| Maximum (%) | 10.00 | 4.99 | 14.96 | 19.94 |

On the other hand, in all simulations, the algorithm reaches a maximum of 170 generations with a median of 35.5 generations. That is, GA always finds a good solution before reaching the established maximum number of generations (200), the presence of faults.

The EHW scheme presented here shows better results compared to those previously obtained in [13], because the GA always finds the solutions with a median in the number of generations similar to the one obtained in [13]. However, it should be noted that this performance is obtained in a more adverse situation. The GA has now less information for evaluating the individual and for guiding the search, and it has to fulfil more demanding specifications. Additionally, the evaluation of the GA considers more complex failure scenarios.

## 6 Conclusions

We present an eight-order low-pass filter, with fault tolerance characteristics obtained through evolvable hardware. The filter under study was implemented in a field programmable analog device. GA developed is robust for the faults addressed in our evaluation. The fault simulation results show that the filter is able to maintain their functionality despite the presence of deviation faults in capacitors and IA. The EHW scheme presented here shows better results compared to those previously obtained using a similar EHW scheme, even if it considers a different test strategy and most demanding performance specifications.

## References


1. Branke, J., Deb, K., Miettinen, K.: Multiobjective optimization: interactive and evolutionary approaches. Springer (2008)
2. Canhoam, R., Tyrrell, A. M.: Evolved fault tolerance in evolvable hardware. In: Congress on Evolutionary Computation, pp. 1267--1272, Hawaii (2002)
3. Collette, Y., Siarry, P.: Multiobjective optimization: principles and case studies. Springer (2003)
4. Designing Higher-order Filters, Lattice Semiconductor Corporation. (2002)
5. El-Ghazali Talbi: Metaheuristics From Design to Implementation. Wiley (2000)
6. Goldberg, D.: Genetic Algorithm. Search, optimization and machine learning. Addison-Wesley (1989)
7. Greewood, G.: On the practicality of using intrinsic reconfiguration as a fault recovery method in analog systems. In: IEEE Trans. Sys. Man & Cyber, pp. 87--97. (1999)
8. Haddow, P. C., Hartmann, M., Djupdal, A.: Addressing the metric challenge: evolved versus traditional fault tolerant circuits. In: Second NASA/ESA Conference on Adaptive Hardware and Systems, pp. 431--438. Edinburgh (2007)
9. Hereford, J.: Fault-tolerant sensor systems using evolvable hardware. In: IEEE Trans. Instrum. Meas, vol. 55, pp. 846--853. (2006)
10. Ji, Q., Wang, Y., Xie, M., Cui, J.: Research on fault-tolerance of analog circuits based on evolvable hardware. In: 7th international conference on Evolvable systems: from biology to hardware, pp. 100--108. (2007)
11. Lovay, M., Arregui, A., Gonella, J., Peretti, G., Romero, E., Lubaszewski, M.: Fault tolerant amplifier system using evolvable hardware. In: Proceedings of the Argentine School of Micro-Nanoelectronics, Technology and Applications, pp. 50--55. (2010)
12. Lovay, M., Peretti, G., Romero, E., Marqués, C.: An Adaptive Amplifier System for Wireless Sensor Network Applications. Journal of Electrical and Computer Engineering, vol. 2012, Article ID 762927, 14 pages. (2012)
13. Lovay, M., Peretti, G., Romero, E., Marqués, C.: Fault-tolerant Filter based on an Evolvable Hardware Technique: a case study. In: 13th Argentine Symposium on Technology (AST), 42 JAIIO, pp. 73--84, ISSN 1850-2806. (2012)
14. Pac-Designer Software User Manual, Lattice Semiconductor Corporation. (2011)
15. Reeves, C., Rowe, J.: Genetic algorithms: principles and perspective. A guide to GA theory. Kluwer Academic Publishers (2002)
16. Roberts, G., Taenzler, F., Burns, M.: An introduction to Mixed-Signal IC Test & Measurement. Oxford University Press, Second Edition (2012)
17. Salem Zebulum, R., Pacheco, M., Vellasco, B. R.: Evolutionary electronics: automatic design of electronic circuits and systems by genetic algorithms. United States: CRC Press (2002)
18. Sivanandam, S., Deepa, S.: Introduction to Genetic Algorithms. Springer (2008)
19. Yu, X., Gen, M.: Introduction to Evolutionary Algorithms. Springer (2010)