Fault-tolerant Filter based on an Evolvable Hardware Technique: a case study.

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 work addresses the problem of providing fault tolerance to an eighth order low-pass filter, employing the principles of evolvable hardware. The filter under study is implemented through the cascade connection of four biquadratic filters in a field programmable analog device. The reconfiguration process of the filter involves the execution of a genetic algorithm (GA) in an external computer, after a fault detected. To perform the test of the filter, we assume that the method of transient analysis is applied. The GA performance is evaluated by fault simulation, employing a parametric fault model. The fault simulation results show that GA finds solutions that meet the established restrictions and presents relatively short run times.

Keywords: Evolvable hardware, adaptive filter, genetic algorithm, programmable devices, multi-objective optimization.

1 Introduction

Analog filters are one of the most employed analog blocks, being essential parts of the analog front end of many electronic systems, like communication systems, medical instruments, and signal processors.

When a filter is operating in harsh environments, there may be certain agents that potentially could deteriorate its performance. If the application is critical, the system can require characteristics of safe operation, adaptation to a changing environment or ability for compensating degradations in its own circuitry.

Providing adaptive characteristics requires configurable hardware sections and a reconfiguration methodology. Particularly, evolvable hardware (EHW) is a methodology that offers self-adaptation by combining reconfigurable hardware with evolutionary algorithms. In EHW, the designer establishes performance goals and usually a genetic algorithm (GA) [1] searches the possible hardware configurations for reaching them [2-3].

In this work, we use an EHW technique for reconfiguring an eighth order low-pass filter. In this way, the filter can achieve adaptation to changes in the operating environment or ability for compensating degradations in its own circuitry. We employ

a field programmable analog array (FPAA) for implementing the filter under test, particularly the ispPAC10 device.

For establishing if a filter reconfiguration is necessary, we consider that the filter is periodically tested during in-field operation. For this case study, we assume that Transient Analysis Method (TRAM) [4] is applied. If the test strategy detects that the system does not meet the specifications, then a GA, which runs in an external computer, evolves the configurable parameters values of the filter. The evolved values are loaded into the hardware for continuing the normal operation.

2 System Description

ispPAC10 is a FPAA device from Lattice Semiconductor [5]. Figure 1 shows the device block diagram. The FPAA has four analog cells (only two cells are shown in the figure), which are composed by an operational amplifier (OA), a resistor, a programmable capacitor, and two input amplifiers (IA) that are connected to the operational amplifier through resistors. In the figure, the shaded area represents the internal resources required for programming connections between the cells. The device also has a reference system, a self-calibration circuitry, and a configuration memory.

The resistance in the feedback loop of the OA has a fixed value and can only be connected or disconnected. Each capacitor can adopt 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 resistors that connect the IA with OA are fixed.

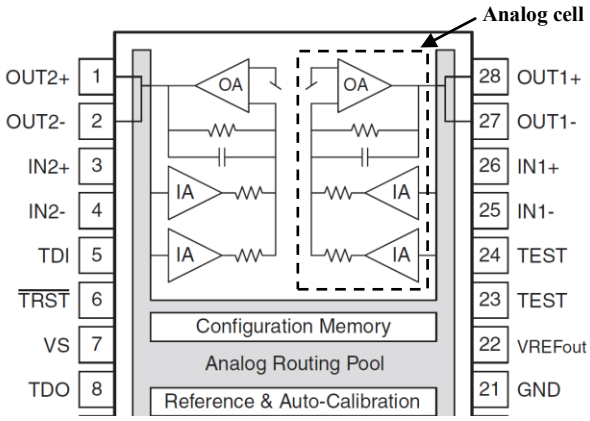

Figure 1. Block diagram of ispPAC10

The design is performed using Pac Designer [6], 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 Butterworth filter, with DC gain equal to one, as a cascaded of four biquadratic sections. This filter is chosen as a starting-point design, and it can be modified according to the application needs.

As each ispPAC10 chip can implement up to two biquadratic filters, the overall system requires two chips. Figure 2 shows the filter frequency response characteristic, corresponding to the so-called nominal response. This characteristic has to be maintained (within tolerance limits) during in-field operation.

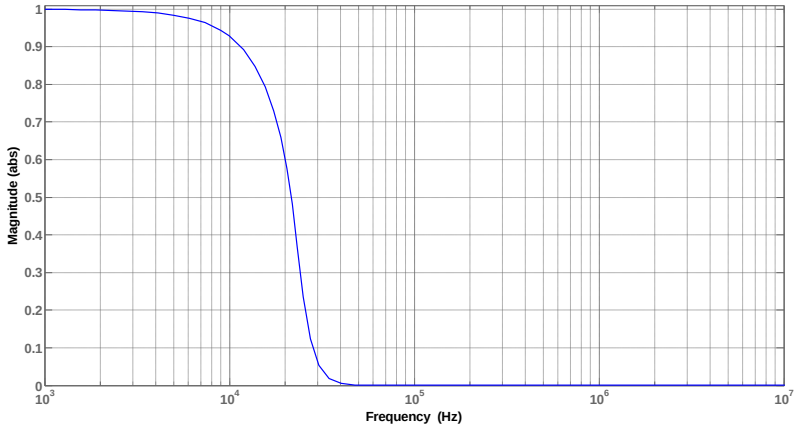

Figure 2. Nominal response of the designed filter

3 Filter testing

To perform the test of the designed filter, we assume that TRAM [4], [7], is applied to each biquadratic section. The capacity of this method for the detection of faults using different fault models has been previously studied [8-12], demonstrating its feasibility. TRAM is a simple test strategy proposed for filters implemented as a cascade of second-order sections. In this strategy, each section is excited with a step, ramp or parabola input signal. It is assumed that a fault in the filter will produce a change in the time occurrence of the peak (T_P) or in the overshoot (OS) of the output transient response. Figure 3 shows the transient response parameters evaluated by TRAM.

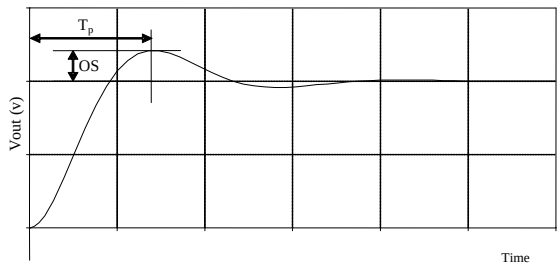

Figure 3. Transient response parameters

By analyzing the transient response, the coefficients of the transfer function can be obtained. T_p and OS are related with the parameters of interest, the undamped natural frequency (ω_n) and the damping factor (ξ) as follows:

$$T_p = \frac{\pi}{\omega_n \cdot \sqrt{1-\zeta^2}} \tag{1}$$

$$OS = e^{-\pi\left(\frac{\zeta}{\sqrt{1-\zeta^2}}\right)} \tag{2}$$

Eqs. (1) and (2) allow establishing ξ and ω_n. In addition, by measuring the output voltage when the signal is stable the gain value can be obtained. In this way, the transfer function of each biquadratic stage and the overall one can be obtained. This function is compared with the nominal function (obtained in the design calculations), with the aim of determining if it is necessary to initiate the reconfiguration process.

The feasibility of application of TRAM to the studied device has been widely demonstrated in previous work. For this reason, it is not analyzed here the implementation of TRAM. Instead, it is assumed that the values of T_p, OS and gain are available after the test routine. The reconstruction of the transfer function is made in the external computer.

4 GA Overview

The overall search space is 7.21E +28 different filter implementations, due to each biquadratic filter use three IAs and two capacitors. When the search space becomes so large, an exhaustive search is no longer possible because the computational cost. In this case, it is recommended the use of a specific technique to find the optimal solution. GAs are one of these methods, which combine direction and chance in the search, allowing simultaneous exploration and exploitation of the search space [13].

GA finds the gain values (k_i) of the IAs and the values (c_i) of the capacitors for each biquadratic stage, with the goal of maintaining the filter response within specifications in the pass band, the cutoff point and the attenuation band. For this reason, the GA has to solve a multi-objective optimization problem [14-15].

In our case, 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 and in the attenuation band. This fact allows using the so-called a priori method, which transform a multi-objective problem into a single-objective one.

Two a priori methods are the ε-Constraint Method (CM) and the Weighted Sum Method (WSM) [14-17]. The first one performs the optimization considering one objective while transforms the others into restrictions. The second method combines the different objective functions into a single one, generally in a linear way. The use of both methods allows employing traditional GAs, as the described in [1], [2], [18].

This work presents a hybrid GA (HGA), which uses the two methods above mentioned in an integrated way, with the aim of improving the efficiency for solving the problem. HGA incorporates the relevant aspects of each method because it

generates the objective function using WSM and transforms all the objectives in restrictions using CM.

Figure 4 shows a flowchart of HGA. For the sake of clarity, in the following the paragraph numbers are related to the corresponding block in the flowchart.

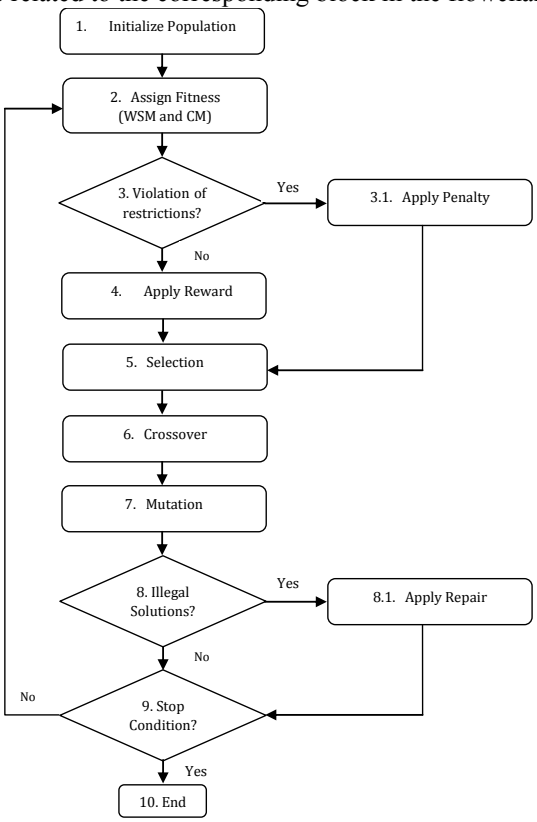

Figure 4. Flowchart of HGA

1. The algorithm starts by randomly generating (with uniform probability) an initial population of individuals that are possible solutions to the problem.
2. For every evolutionary step, known as generation, the individuals in the current population are evaluated according to some predefined quality criterion called the fitness function (objective function constructed using WSM and CM). 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.
3. HGA takes into account the restrictions by the use of a penalty technique (3.1 in the flowchart). By using this technique, the solutions violating the restrictions are modified in their fitness values (based on the violation degree) in order to decrease their chances of being selected.
4. HGA uses a technique of rewards, by which the solutions that meet the restrictions are modified in their fitness values (based on the compliance degree) in order to increase their chances of being selected.

5. The selection of the individuals for the next generation is performed through the method of the rotating roulette. The probability of an individual to be selected for crossover is proportional to its fitness.
6. The crossover operator selects two or more individuals, called parents, and exchanges parts of their information to form two or more new individuals, called offspring. The parents who do not undergo the crossover operation are copied unchanged to the new pool of individuals.
7. 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, at random with some probability.
8. If the individual generated does not represent a legal solution due to the nature of the encoding technique, then it is repaired (8.1 in the flowchart).
9. This new generation goes through the process described above, from the fitness evaluation to the mutation step. The cycle repeats until a stop criterion is met, such as a maximum number of generations is reached or a desired solution is found.

5 GA Implementation

After performing an analysis of the magnitude response of nominal filter (Fig. 2), we selected five evaluation points. At each point, HGA evaluates the transfer function of each individual (possible values of k_i and c_i determining a filter design), compares it with the nominal filter response and assigns the fitness. Figure 5 shows the location of the points.

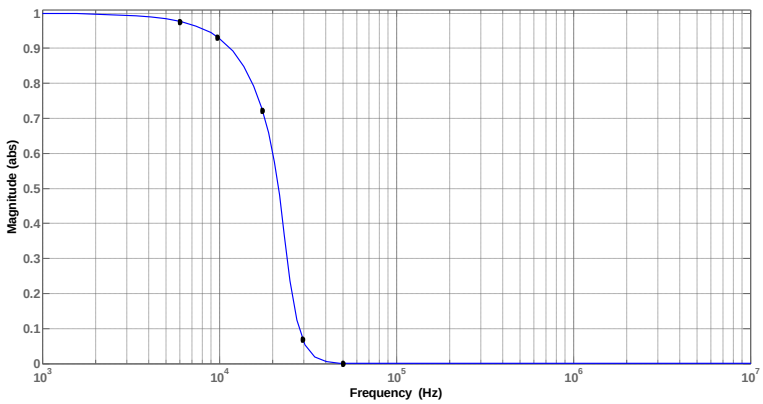

Figure 5. Location of points with which the GA performs the evaluation

As shown in the figure, we consider five points: two of them correspond to the pass band, one to the cutoff point and two at the attenuation band. For each individual (*y*), the algorithm determines:

1. ***Relative error in magnitude* ($REM_P(y)$):** represents the error in magnitude between a given individual in the population ($M_P(y)$) and the nominal filter ($M_P(NF)$) at a point P.

$$REM_P(y) = \left|\frac{M_P(y) - M_P(NF)}{M_P(NF)}\right| \quad (3)$$

2. ***Relative error in frequency* ($REF_P(y)$):** represents the error in frequency between a given individual in the population ($F_P(y)$) and the nominal filter ($F_P(NF)$) at a point P.

$$REF_P(y) = \left|\frac{F_P(y) - F_P(NF)}{F_P(NF)}\right| \quad (4)$$

Figure 6 shows the graphic interpretation of the above mentioned errors.

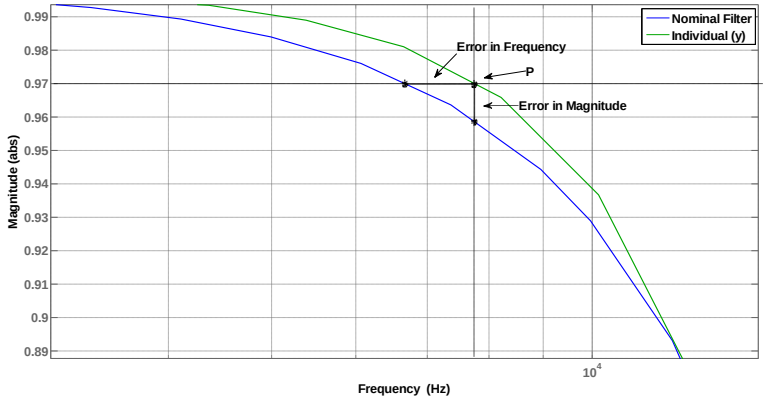

Figure 6. Graphic interpretation of the two errors considered

The function that assigns a fitness value to each individual is developed considering:

- The relative errors in magnitude and in frequency obtained at the cutoff point (REM_1 and REF_1).
- The *maximum* relative errors in magnitude and in frequency obtained from the evaluation of the points corresponding to the pass band (REM_2 and REF_2).
- The *maximum* relative errors in magnitude and in frequency obtained from the evaluation of the points corresponding to the attenuation band (REM_3 and REF_3).

As previously stated, HGA uses WSM and CM in hybrid form. The resulting fitness function (f) to assign a fitness value to each individual y is:

$$f(y) = B - \sum_{i=1}^{3} w_i \left[REM_i(y) + REF_i(y)\right]$$

Subject to: (5)

$$REM_i(y) \wedge REM_i(y) \leq \varepsilon_i$$

In (5), i adopts the values 1, 2 and 3, for the cutoff point, the pass band and the attenuation band respectively. The values of w_i are called weights, representing the

degree of importance assigned to the error in each band. The values of ε_i represent the maximum tolerable error defined for each band. B is a constant added for avoiding negative numbers.

In order to apply the restrictions, the algorithm penalizes the individuals with errors above ε_i. Consequently, these individuals adopt a lower fitness (Lf), according to the degree of constraint violation and the degree of importance assigned to the error in each band (w_i), as follows:

$$Lf(y) = \sum[w_i.(f(y) - f(y) \cdot \frac{(RE_i(y) - \varepsilon_i)}{\varepsilon_i})] \quad (6)$$

In (6), the penalty is applied considering only bands where restrictions are not met.

HGA increases the f value to the individuals that present errors less than or equal to ε_i in the *three* bands as follows:

$$Hf(y) = \sum_{i=1}^{3}[w_i.(f(y) + f(y) \cdot \frac{(\varepsilon_i - RE_i(y))}{\varepsilon_i})] \quad (7)$$

In (6) and (7), the value of RE_i is the higher of REM_i and REF_i.

HGA employs real coding and uses Simplex Crossover Method (SPX) and uniform random mutation method [16].

The algorithm is implemented using as parameters an initial population of 200 individuals, a crossover probability of 1 and a mutation probability of 0.05. These values are chosen using previous experimental guidelines [18-19].

The selection of the individuals for the crossover is performed through the method of the rotating roulette. The algorithm stops when it reaches 100 generations, or when it finds a solution that meets the restrictions previously mentioned.

6 Experimental Results

6.1 Fault Free Operation

We set the value of the maximum tolerable error for each band as follows: ε_1=10%, ε_2=20% and ε_3=40%, for demonstration purposes. The values assigned to the weights used in (5) are w_1=0.46, w_2=0.46 and w_3=0.08. 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.

As GA is a stochastic process, its results could change according to the statistical distribution of the initial population. In order to see how the results could be affected by the setting of the initial population, we change the seed of its random generation and consider that all the values associated with the capacitors and the IAs are available for the algorithm.

Figure 7 shows the relative error in frequency for HGA 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 three bands, the relative error satisfies the performance criteria established.

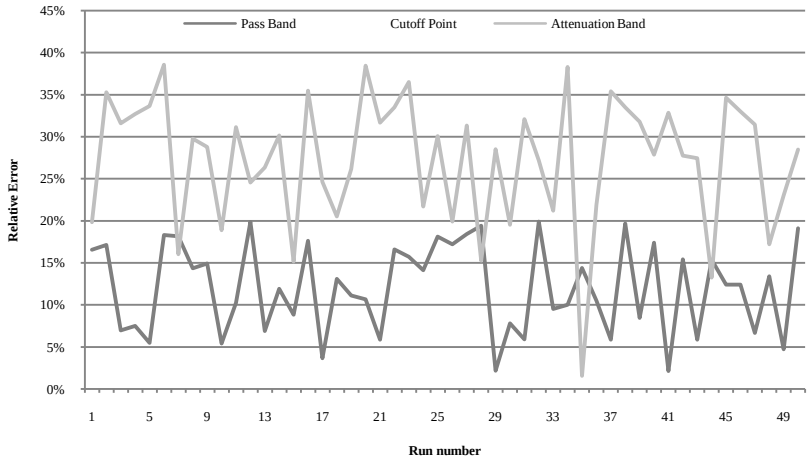

Figure 7. Errors for HGA. Fault-free operation

Table 1 shows a characterization of the relative error in frequency 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. HGA maintains the maximum relative errors within the criteria established in the three bands.

Table 1. Relative error characterization under fault-free conditions.

Band	Median (%)	Minimum Error (%)	Maximum Error (%)
Cutoff Point	3.81	0.17	8.35
Pass Band	12.44	2.15	19.92
Attenuation Band	28.65	1.57	38.56

On the other hand, in all simulations, HGA completes its execution at most in 77 generations, with a median of 33 generations. That is, the algorithm always finds a good solution before reaching the established maximum number of generations (100). The maximum run time of HGA is 677.69 seconds, with a median of 286.61 seconds.

Figure 8 shows the magnitude response for the nominal filter, the best and the worst design obtained from the 50 simulations performed with values provided by HGA.

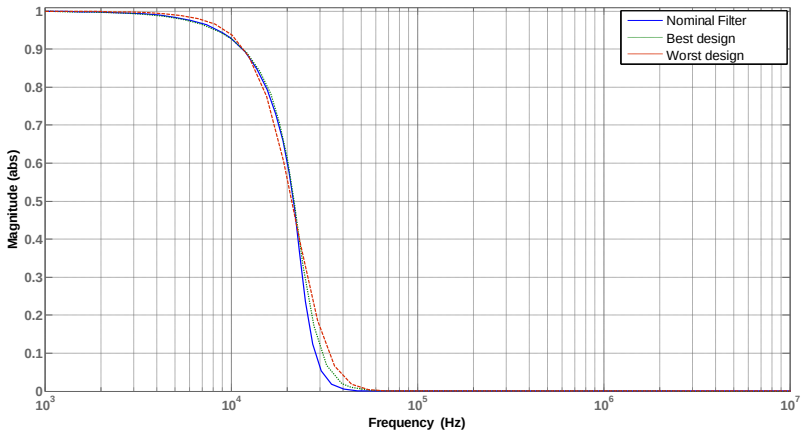

Figure 8. Magnitude response under fault-free condition

6.2 Operation under fault condition

The performance of the fault tolerance scheme with HGA is evaluated by means of fault injection. The fault model used is a parametric, one single fault, and assumes that there is a deviation in the capacitors of each biquadratic filter. Particularly, we consider that the capacitors deviate their values in a percentage of their nominal values, ± 10%, ±20%, ±30%, ±40% and ±50%.

Figure 9 depicts the relative errors in frequency in each of the bands for each deviation value in the capacitors. From the simulation results, it is observed that HGA is able to maintain system performance within specifications.

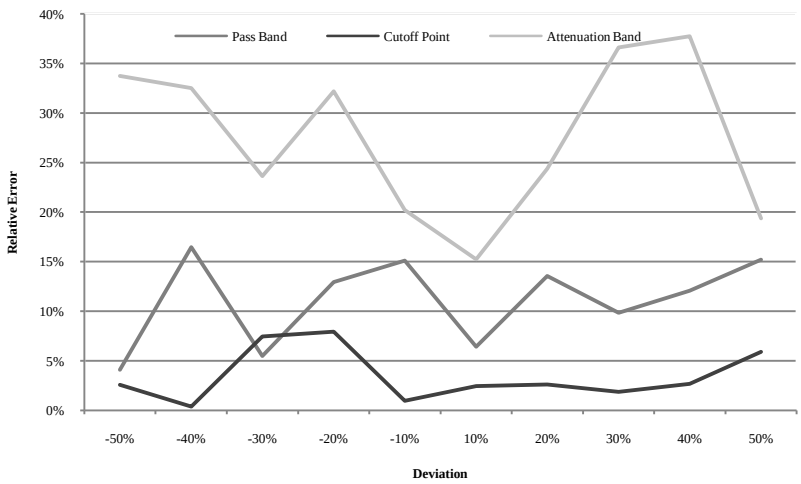

Figure 9. Errors for HGA under fault condition

Table 2 summarizes the effects of deviation faults. Comparing the normal (Table 1) and deviation fault conditions (Table 2), the faulty system presents an increase in the minimum errors and a decrease in the maximum errors. The median also decreases, excepting the pass band.

Table 2. Relative error characterization of HGA under fault condition.

Band	Median (%)	Minimum Error (%)	Maximum Error (%)
Cutoff Point	2.58	0.36	7.92
Pass Band	12.50	4.09	16.45
Attenuation Band	28.29	15.23	37.74

Regarding to the number of generations, the algorithm presents a slight increase in the maximum number of generations reached (83) and in the median (39), with respect to normal operation. For this reason the maximum run time (684.17 seconds) and the median time (308.19 seconds) also present a small increase.

7 Conclusions and future works

We present a high order filter based on programmable devices, with fault tolerance characteristics obtained through evolvable hardware. GA presented uses two a priori methods in an integrated way. This algorithm 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 faults. In addition, the genetic algorithm presents relatively short run times, even under fault conditions. In future works, will analyze the performance of GA using other fault models, such as catastrophic faults in capacitors and IAs.

References

1. Goldberg D., “Genetic Algorithm. Search, optimization and machine learning”, Addison-Wesley, 1989
2. Salem Zebulum R., Pacheco M., and Vellasco M., “Evolutionary electronics: automatic design of electronic circuits and systems by genetic algorithms”, CRC Press, 2002
3. Ji Q., Wang Y., Xie M., and Cui J., “Research on fault-tolerance of analog circuits based on evolvable hardware”, in Proceedings 7th international conference on Evolvable systems: from biology to hardware, pp. 100-108, 2007
4. Calvano J., Alves V., and Lubaszeswski M., “Fault detection methodology for second order filters using compact test vectors transient analysis”, 3rd. International Workshop on Design of Mixed-Mode Integrated Circuits and Applications, 18-24, 1999
5. Designing Higher-order Filters, Lattice Semiconductor Corporation, 2002
6. Pac-Designer Software User Manual, Lattice Semiconductor Corporation, 2011

7. Calvano J., Alves V., and Lubaszeswski M., “Fault detection methodology and BIST method for 2nd order Butterworth, Chebyshev and Bessel approximations”. Proceedings 18th IEEE VLSI Test Symposium, 319-324, 2000
8. Peralta J., Peretti G., Romero E., and Marqués C., “A new performance characterization of transient analysis method”. International Journal of Electronics, Communications and Computer Engineering, 3:12-19, 2009
9. Peralta J., Lovay M., Peretti P., Romero E., Marqués C, “Evaluación de la capacidad del test basado en transitorio para detectar fallas de degradación”, Mecánica Computacional Vol. XXVIII, pp. 2943-2953. November 2009
10. Peralta J., Peretti G., Romero E., and Marqués C., “Evaluation of circuit test strategies using statistical fault models: a case study”, Mecánica Computacional, vol. 26, pp. 2007-2015, Spanish, 2007
11. Peralta J., Peretti G., Romero E., Demarco G., and Marqués C., “Quality Assessment of Transient Response Analysis Method for Detecting Radiation-Induced Faults”, International Journal of Quality, Statistics, and Reliability, Volume 2011 (2011)
12. Peralta J., Peretti G., Laprovitta A., Romero E., and Marqués C., “Evaluation of the transient analysis method ability for detecting deviation faults in space environments”, Argentine School of Micro-Nanoelectronics Technology and Applications (EAMTA), 2011
13. Sivanandam S., Deepa S., “Introduction to Genetic Algorithms”, Springer, 2008.
14. Collette Y., and Siarry P., “Multiobjective optimization: principles and case studies”, Springer, 2003
15. Branke J., Deb K., and Miettinen K., “Multiobjective optimization: interactive and evolutionary approaches”, Springer, 2008
16. El-Ghazali Talbi, “Metaheuristics From Design to Implementation”, Wiley, 2009
17. Yu X., Gen M., “Introduction to Evolutionary Algorithms”, Springer 2010
18. Reeves C., and Rowe J., “Genetic algorithms: principles and perspective. A guide to GA theory”, Kluwer Academic Publishers, 2002
19. Hereford J., “Fault-tolerant sensor systems using evolvable hardware”, IEEE Trans. Instrum. Meas, vol. 55, n°3, pp. 846-853, 2006