FUGA: A Fuzzy-Genetic Analog Circuit Optimization Kernel

Pedro Sousa, Nuno Horta

Instituto de Telecomunicações
IST – Torre Norte, AV. Rovisco Pais, 1
1049-001 Lisboa, Portugal
(351) – 218418093

ABSTRACT
The microelectronics market trends present an ever-increasing level of complexity with special emphasis on the production of complex mixed-signal systems-on-chip. Strict economic and design pressures have driven the development of new methods for automating the analog design process. However, and despite some significant research efforts, the essential act of design is still performed by the trial and error interaction between the designer and the simulator.

The proposed approach focuses on the development of a new strategy based on a Fuzzy Genetic Analog Circuit Optimization, in order to increase efficiency on the analog circuit and system design cycle. It combines Evolution Computation techniques with Fuzzy Logic Models, in order to be able to deal with multi-objective problems and to speed up the design cycle.

The work reported addresses the development of an efficient optimization tool FUGA. The resulting synthesis optimization kernel, simulation capabilities, architecture and the behavior, are presented. The improvement in design productivity and performance is demonstrated for a set of well known several circuits and analog designs.

Keywords

1. INTRODUCTION
Designing complex analog and mixed-signal circuits and systems is a complex and cumbersome task that requires extensive design expertise. Despite the evolution of design automation approaches in the last decades, most of the designer effort in the synthesis of an analog system is still dedicated to circuit sizing in order to satisfy a set of predefined performance specifications and parameters constraints. Therefore, the development of new design automation procedures to improve the design efficiency is mandatory [1, 7].

Recent approaches to analog design automation show an enormous potential of applying evolutionary computation techniques to both circuit/system level [6] and layout level [5], due to their capability of evolving solutions on large search spaces associated to either linear or non-linear design problems. Nevertheless, the introduction of design knowledge during the optimization process should allow optimization techniques to prune the search space and, therefore, attain the desired solution in a more effective way. The design knowledge may be introduced using several approaches, e.g., SVM Models, Neural Networks, Fuzzy Systems, etc. The Fuzzy approach has the advantage of using models, which are fully interpretable, in terms of design rules and physical parameters. Recent works include fuzzy descriptions alone or combined with others approaches to model knowledge at different levels including circuit’s performance equations [3], topology selection [2], measure the degree of fulfillment of the objectives and constraints [4].

This paper presents a new circuit level optimization approach based on an evolutionary computation kernel composed by a genetic algorithm combined with a fuzzy model, describing design knowledge. The fuzzy model is generated based on the interpretation of a set of samples obtained with a Design of Experiments (DOE) approach. The interaction between the fuzzy model and genetic algorithm, during the optimization process, is here, made at the mutation operator level. The effectiveness of the proposed approach is shown using well known circuit examples.

The paper is organized as follows: Section 2 describes a standard genetic algorithm (GA) approach applied to analog circuits/systems optimization. Section 3 presents the proposed fuzzy-genetic approach to boost the GA optimization kernel performance. Section 4 demonstrates with examples the effective gain of the proposed approach. Finally, in section 5 the conclusions are drawn.

2. GA Optimization Kernel
The proposed fuzzy-genetic optimization approach rises on a GA standard kernel implementation. Therefore, the implemented GA approach, which will serve as a benchmark to evaluate the new approach, will be summarized in this section.

First, the search space, a R^n space, is limited on each dimension by the range associated to each optimization variable, which may be discrete or continues. Naturally, each variable will be represented by a gene and the complete set of genes (variables) form the chromosome, which represents a point inside the n-dimensional search space.

Then, the implementation of each GA steps were intentionally made simple, once, the aim of this study is to evaluate the ability of the fuzzy model, representing design knowledge, to improve the optimization kernel. Moreover, as will be seen later, the proposed approach applies to any other GA implementation. Therefore, the GA steps are implemented as follows: the initial
population generation is random, the paring operator selects the pairs at random from the upper half of the population, the crossover operator uses a single point, the mutation operator randomly selects the genes to be mutated.

Finally, the cost function measures the absolute distance (error) of the solution of each chromosome to the desired performance goals.

The membership function is applied for all input variables (in our case, the input variables are the genes that compose the chromosome) and can be either triangular or Gaussian, decided by the user at this level. The membership function converts an element \( x \) in crispy values into fuzzy a standard value \( \mu(x) \) set between \([0, 1]\).

The fuzzy rules are defined based on the circuit behavior and their general form is given by if-then rules, as the following: “If input is low then output is max”, where low and max are membership function terms. For simple cases these rules can easily be described manually, however, for more complex structures it is clearly impossible to infer such information in a reasonable time. Therefore, in section 3.2, an approach to automatically generate the fuzzy rules will be discussed.

The fuzzy inference algorithm is applied to activate a subset of rules from the complete set of fuzzy rules, this, in agreement with the fuzzy values. There are many operators to implement the inference mechanism, e.g, MIN, MAX, SUM and PROD or a hybrid solution with more than one. Here, the inference is implemented based on a MAX-MIN strategy.

As mentioned before, the input variables are converted from crispy values to fuzzy values. Therefore, it is needed to defuzzify the result of the fuzzy inference. A defuzzification is a process to get a best possibility representation of the distribution of an inferred fuzzy action. Several methods can also be applied here, e.g., “center-of-gravity”, “center-of-area”, “first-of-maxima”, “last-of-maxima”, “middle-of-maxima”, “center-of-area for singletons”, etc. Here, the center-of-area approach (1) was selected.

\[
x_i = \frac{\sum x_j \mu(x_j)}{\sum \mu(x_j)}
\]

At last, the result returned by the defuzzification method is always a rated value. Thus, we must analyze this result to take a final decision.

The fuzzy model is defined in 5 steps, as follows:

Step 1) membership functions
Step 2) fuzzy rules
Step 3) Inference algorithm
Step 4) Defuzzification
Step 5) solution normalization

The fuzzy model here presented has the goal of incorporating design knowledge to allow a faster convergence of the optimization algorithm. The design knowledge is described in terms of the design variables and their qualitative contribution to the performance measures.

The fuzzy model is defined in 5 steps, as follows:

3. Proposed Fuzzy-Genetic Approach

In this section, the fuzzy model structure is briefly described, then, the process to automatically generate the model is discussed and, finally, the proposed fuzzy model integration on the GA flow is presented.

3.1 The fuzzy model

The fuzzy model here presented has the goal of incorporating design knowledge to allow a faster convergence of the optimization algorithm. The design knowledge is described in terms of the design variables and their qualitative contribution to the performance measures.

The fuzzy model is defined in 5 steps, as follows:

Step 1) membership functions
Step 2) fuzzy rules
Step 3) Inference algorithm
Step 4) Defuzzification
Step 5) solution normalization

The membership function is applied for all input variables (in our case, the input variables are the genes that compose the chromosome) and can be either triangular or Gaussian, decided by the user at this level. The membership function converts an element \( x \) in crispy values into fuzzy a standard value \( \mu(x) \) set between \([0, 1]\).

The fuzzy rules are defined based on the circuit behavior and their general form is given by if-then rules, as the following: “If input is low then output is max”, where low and max are membership function terms. For simple cases these rules can easily be described manually, however, for more complex structures it is clearly impossible to infer such information in a reasonable time. Therefore, in section 3.2, an approach to automatically generate the fuzzy rules will be discussed.

The fuzzy inference algorithm is applied to activate a subset of rules from the complete set of fuzzy rules, this, in agreement with the fuzzy values. There are many operators to implement the inference mechanism, e.g, MIN, MAX, SUM and PROD or a hybrid solution with more than one. Here, the inference is implemented based on a MAX-MIN strategy.

As mentioned before, the input variables are converted from crispy values to fuzzy values. Therefore, it is needed to defuzzify the result of the fuzzy inference. A defuzzification is a process to get a best possibility representation of the distribution of an inferred fuzzy action. Several methods can also be applied here, e.g., “center-of-gravity”, “center-of-area”, “first-of-maxima”, “last-of-maxima”, “middle-of-maxima”, “center-of-area for singletons”, etc. Here, the center-of-area approach (1) was selected.

\[
x_i = \frac{\sum x_j \mu(x_j)}{\sum \mu(x_j)}
\]

At last, the result returned by the defuzzification method is always a rated value. Thus, we must analyze this result to take a final decision.

3.2 Automatic Fuzzy Rules Generation

The generation of fuzzy rules implies determining how a change in an input variable may contribute to change a specific output or performance parameter. The generation of fuzzy rules can be done with several different approaches. However, the addressed design problem in our case range from simple to highly complex problems with large number of design variables leading to huge design spaces. Therefore, it is not feasible to use an exact analysis, so, in order to generate the desired fuzzy rules, capable of indicating in a qualitative form the right direction for the search engine, the design space must be efficiently sampled and those samples must be combined and analyzed to generate the rules. For this purpose, the Design of Experiments (DOE) technique was chosen. DOE techniques provide a mathematical basis to select a limited but “optimal” set of sample points needed to fit a black-box model [8]. Well-known and often-used sampling schemes range from full and fractional factorial design, over Plackett-Burman and Taguchi schemes, to Latin hypercube and even random design [9].
The simplest types of factorial designs involve only two factors of treatments, the “high” and “low” levels of a factor and are represented by $2^k$ factorial design, where $k$ is the number of input variables and $2^k$ represents the number of treatment combinations. If $k$ represents the total of input variables, it will correspond to the full factorial design (all set of points are present). As the number of factors in a $2^k$ factorial design increases, the number of runs, required for a complete sampling of the design space, rapidly outgrows the resources of most experiments. If the experimenter can reasonably assume that certain high-order interactions are negligible, information on the main effects and low-order interactions may be obtained by running only a fraction of the complete factorial experiment. In this case the number of combinations is represented by $2^{k-p}$ fractional factorial design, where $p$ represents the independent generators. It is important to select the $p$ generators for a $2^{k-p}$ fractional factorial design in such a way that we obtain the best possible alias relationships. A reasonable criterion is to select the generators such that the resulting $2^{k-p}$ design has the highest possible resolution.

Sometimes, the two factors of treatments become inappropriate for non-linear systems. In these cases, a higher number of factors of treatment are necessary to model an eligible relationship between input and output variables. Therefore, we can represent $b^k$ and $b^{k-p}$ to factorial design and fractional factorial design, respectively, where variable $b$ represents the number of factors of treatment.

The present model provides a set of five steps that is defined as follows:

Step 1) Convert the design space from $b^k$ to $b^{k-p}$ fractional factorial design space.

Step 2) Determine the new combination of sample points for the fractional factorial design.

Step 3) Evaluate the respective values for output variables.

Step 4) Evaluate the main effects of the control parameters (input variables) on each output.

Step 5) Generate the fuzzy rules.

Consider, for simplicity, that a first order Low-Pass filter is selected, i.e., one stage from the example of figure 10. The gain and pole expression and the respective domains for the input variables are defined in (2) and (3), respectively.

$$|A_0| = \frac{R_2}{R_1} \quad F_0 = \frac{1}{2\pi R_2 C} \quad R_1 (\Omega) = [800, 1200] \quad R_2 (\Omega) = [80000, 120000] \quad C (F) = [0.6 \times 10^{-12}, 1.0 \times 10^{-12}]$$  \hspace{1cm} (2)

The results of Gain and Frequency, described in table 1, can be used in a hypercube representation to compare the main difference between full and fractional factorial design space. This difference is shown in figure 2 and 3.

In step 1, a choice of a value for variable $p$ should be taken. For this example, the value of variable $p$ is 1. Consequently, the full factorial design will be placed for a $2^{k-p}$ fractional factorial design. With this choice, it only is necessary 4 runs for each output variables.

In step 2, a set of sample points is determined. If we are considering two factors of treatments, we assume that $R_1^\pm$ represents the “low” level and $R_1^+$ represents the “high” level (assuming a similar representation for $R_2$ and $C$). Notice that, the “low” and the “high” levels are always represented by the minimum and the maximum value that is defined in (3), respectively. In step 3, the sample points are evaluated. The set of sample points that will be considered, as well as, the respective values for output variables obtained from equations described in (2), are presented in table 1. Remember, that this is an intentionally simple example and for complex circuits, e.g., operational amplifiers, we will not be able to derive all the exact equations and, therefore, the sample points will be evaluated by electrical simulation.

The results of Gain and Frequency, described in table 1, can be used in a hypercube representation to compare the main difference between full and fractional factorial design space. This difference is shown in figure 2 and 3.

<table>
<thead>
<tr>
<th>Run 1</th>
<th>$R_1^-$</th>
<th>$R_1^+$</th>
<th>$C^+$</th>
<th>Gain [dB]</th>
<th>Frequency [MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 2</td>
<td>$R_1^+$</td>
<td>$R_2^+$</td>
<td>$C^+$</td>
<td>43.52</td>
<td>1.326</td>
</tr>
<tr>
<td>Run 3</td>
<td>$R_1^+$</td>
<td>$R_2^-$</td>
<td>$C^+$</td>
<td>36.48</td>
<td>1.989</td>
</tr>
<tr>
<td>Run 4</td>
<td>$R_1^+$</td>
<td>$R_2^+$</td>
<td>$C^-$</td>
<td>40.00</td>
<td>2.210</td>
</tr>
</tbody>
</table>

Figure 2. Projection of a $2^3$ full factorial design into three $2^2$ designs.
With this interaction effect it is clear which input variables would influence each output parameter. In tables 2 and 3, an example to evaluate the main interaction effect for the Gain and Frequency is presented, respectively.

Table 2. Input variable effect on the Gain output.

<table>
<thead>
<tr>
<th>Input</th>
<th>Interaction effect</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_1</td>
<td>(36.48 + 40.00) – (40.00 + 43.52)</td>
<td>-7.04</td>
</tr>
<tr>
<td>R_2</td>
<td>(43.52 + 40.00) – (40.00 + 36.48)</td>
<td>7.04</td>
</tr>
<tr>
<td>C</td>
<td>(40.00 + 40.00) – (43.52 + 36.48)</td>
<td>0</td>
</tr>
</tbody>
</table>

A graphical representation of the effect of each input variables on each output parameter is presented in figure 4 and 5.

Table 3. Input variable effect on the Frequency output.

<table>
<thead>
<tr>
<th>Input</th>
<th>Interaction effect</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_1</td>
<td>(1.989 + 2.210) – (3.316 + 1.326)</td>
<td>40.443</td>
</tr>
<tr>
<td>R_2</td>
<td>(1.326 + 2.210) – (3.316 + 1.989)</td>
<td>-1.769</td>
</tr>
<tr>
<td>C</td>
<td>(1.326 + 1.989) – (3.316 + 2.210)</td>
<td>-2.211</td>
</tr>
</tbody>
</table>

An illustrative example of the overall fuzzy rules generated for both output variables are shown in tables 4 and 5.

Table 4. Set of Fuzzy Rules for Gain.

<table>
<thead>
<tr>
<th>Rule</th>
<th>R_1</th>
<th>R_2</th>
<th>A_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1:</td>
<td>IF Low AND Low</td>
<td>THEN Med</td>
<td></td>
</tr>
<tr>
<td>Rule 2:</td>
<td>IF Low AND Mod</td>
<td>THEN Min</td>
<td></td>
</tr>
<tr>
<td>Rule 3:</td>
<td>IF Low AND High</td>
<td>THEN Min</td>
<td></td>
</tr>
<tr>
<td>Rule 4:</td>
<td>IF Mod AND Low</td>
<td>THEN Max</td>
<td></td>
</tr>
<tr>
<td>Rule 5:</td>
<td>IF Mod AND Mod</td>
<td>THEN Med</td>
<td></td>
</tr>
<tr>
<td>Rule 6:</td>
<td>IF Mod AND High</td>
<td>THEN Min</td>
<td></td>
</tr>
<tr>
<td>Rule 7:</td>
<td>IF High AND Low</td>
<td>THEN Max</td>
<td></td>
</tr>
<tr>
<td>Rule 8:</td>
<td>IF High AND Mod</td>
<td>THEN Max</td>
<td></td>
</tr>
<tr>
<td>Rule 9:</td>
<td>IF High AND High</td>
<td>THEN Med</td>
<td></td>
</tr>
</tbody>
</table>
3.3 Fuzzy Model Integration into GA Flow

The previously described fuzzy model was integrated in a GA flow in order to introduce design knowledge during the search process and improve the efficiency of the optimization kernel.

![Figure 6. Fuzzy Model applied to the mutation operator.](image)

In the present case the fuzzy model will be used to support the GA’s mutation operator. As mentioned before, the mutation replaces the value of a gene with a randomly-generated value from the domain defined for each input variable. But sometimes, those changes may move the individual to an undesired or unpromising search space region, especially when large domains are considered. The inclusion of the fuzzy model allows a fine control of the new gene randomly-generated by indicating a more promising orientation for the new gene value. The change on the mutation operator based on the fuzzy model is shown in figure 6, for the case of a mutation in variable 2.

In step 4, the results for Gain and Frequency are obtained by applying the defuzzification, considering all the input variables in the selected chromosome. The result is achieved using a triangular shape, three levels of subsets and is applying the MAX-MIN inference method. Under the center-of-area method, the result from defuzzification gives a 64.76% and 50.94% output for the Gain and Frequency, respectively. As the results show, the Frequency’s defuzzification result presents a more stable output then Gain’s result.

![Figure 7. Defuzzification’s result for Gain.](image)

![Figure 8. Defuzzification’s result for Frequency.](image)

The final step of fuzzy based mutation operator would be to analyze the defuzzification results and return one of three possible alternatives:

1) Increase the value of the input variable;
2) decrease the value of the input variable;
3) Gives a new random value between a small ranged around the present value.

When the conclusion process chooses the third alternative, it means that the present value (cell proposed to mutation) of input variable is already near to the best goal. For that reason, a small change around the present value must be defined and a new random value is calculated. It guarantees that the fitness of that
chromosome would not have large variations. Otherwise, when the conclusion process chooses the first supposition, the increase or decrease of the cell must be done respecting the limits of the respective domains.

For the working example, and considering the chromosome presented in figure 6, a graphical illustration of the population convergence is shown in figure 9. Moreover, the qualitative information obtained after the defuzzification allows the following conclusion, to be applied by the mutation operator: a mutation for R\textsubscript{1} will result on an increment of the present value, a mutation for R\textsubscript{2} will result on a decrement of the present value and, finally, for variable C the fuzzy model concludes that the present value is near the main goal, once the output from the defuzzification is stable, so, in this case, a random mutation will be considered, allowing only a small change around the present value.

4. Examples
In this section, an example of analog circuit synthesis is discussed, showing the viability of the proposed approach. The results here presented were all produced by a C platform implementation of the proposed approach and using an equation-based approach to evaluate the circuit performance.

All the present examples were executed for a population of 32 chromosomes, a random selection over the best 16 chromosomes for paring, a single crossover considered for mating and for the mutation operator either a random approach (5\%) or a fuzzy model approach.

4.1 Example 1: Active 2\textsuperscript{nd} Order LP Filter
The example shows the circuit optimization using a standard genetic algorithm kernel (GA) versus a fuzzy-genetic algorithm kernel which introduces the fuzzy rules, in the mutation step of the GA optimization process.

The proposed optimization approach is independent from the circuit, so, a well known circuit structure was selected in order to clearly illustrate the achieved performance gains.

Figure 9. Active second order Low-Pass Filter.

Considering the filter transfer function in (4) and the equivalent representation (5) in terms of the DC Gain, the cut frequency (\(\omega_0\)) and the quality factor (Q), the test is here performed with the goal described in (6) and considering the passive elements sizes (R\textsubscript{1} to R\textsubscript{4}; C\textsubscript{1},C\textsubscript{2}) as optimization variables.

\[
T(s) = \frac{1}{s^2 + s \left( \frac{1}{R_1 C_1} + \frac{1}{R_2 C_2} \right) + \frac{1}{R_2 R_4 C_1 C_2}} \tag{4}
\]

\[
T(s) = \frac{\omega_0^2}{s^2 + \frac{\omega_0^2}{Q}} \tag{5}
\]

\[
A_{0,\text{th}} = 70dB \pm 0.1
\]

\[
\omega_{0,\text{th}} = 3.6E6 rad s^{-1} \pm 1E3
\]

\[
Q_{0,\text{th}} = 0.48 \pm 0.02
\]

Finally, the cost function is given by the following expression (7) representing the sum of the normalized distances to each performance goal.

\[
Cost = \frac{|A_{\text{exp}} - A_{\text{th}}| + |F_{\text{exp}} - F_{\text{th}}| + |Q_{\text{exp}} - Q_{\text{th}}|}{A_{\text{th}} + F_{\text{th}} + Q_{\text{th}}} \tag{7}
\]

In figure 11, the results for the standard genetic algorithm approach are shown. For this example, a limit of 400 generations was imposed. The end of each run is reached only if the intended solutions are found or the limit of GA generations is reached. In figure 12, the results obtained with the fuzzy-genetic approach are presented and show that incorporating design knowledge to restrict the mutation change leads to impressive gains in terms of performance, as described in table 6. First, the success rate is increased, then, the number of required generations is reduced, and, finally, the number of cost function calls is also reduced. The cpu time is higher for the fuzzy-genetic approach, due to additional computation required to determine the mutation value, however, we must remember that we are discussing prototype implementation that will, in the future, use electrical simulation to determine the cost function value, and, therefore, due to the reduced fuzzy-genetic cost function calls it is expectable to also achieve a significant reduction in terms of execution time.
Table 6. The optimization kernel performance for the Fuzzy-GA and Standard GA approaches.

<table>
<thead>
<tr>
<th></th>
<th>Fuzzy-GA</th>
<th>Standard GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nº Runs</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Average (GA Gen. Num)</td>
<td>43.51</td>
<td>207.73</td>
</tr>
<tr>
<td>Std (GA Gen. Num)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Success Rate [%]</td>
<td>99</td>
<td>77</td>
</tr>
<tr>
<td>Cost Function Calls</td>
<td>289 090</td>
<td>1 233 990</td>
</tr>
<tr>
<td>CPU Time [seg]</td>
<td>8.71</td>
<td>1.38</td>
</tr>
</tbody>
</table>

4.2 Example 2: Generalized Test

In order to evaluate the generalization of the proposed approach, several tests were performed considering active filter structures with both a different number of optimization variables and a different number of performance specs, as illustrated in table 7. The results presented in table 8 show significant improvements when compared to the standard approach, particularly, the success rate is always superior, the number of required generations and the number of cost function evaluations is considerably reduced. These results show that the proposed approach can be generalized once it keeps or improves the benefits as both the number of optimization variables and performance specs increase.

5. Conclusion and Future Work

The proposed Fuzzy-Genetic approach to implement an optimization kernel for circuit level design introduces considerable gains, in terms of performance, by using design knowledge, represented by a set of fuzzy rules, in order to persuade the genetic algorithm to search in more promising directions. The examples also show the scalability of the approach allowing its application to more complex circuit structures. Preliminary results show a large reduction in terms of the number of fitness function evaluations and, also, the number of generations needed to reach the desired solution. The additional time required to reach the solution is not relevant when considering the time needed to perform the additional number of electrical simulation in the case of standard GA approach. Finally, the automatic generation of the design fuzzy rules represents an additional advantage of the proposed solution when optimizing highly complex structures.

For future work a fuzzy model to other GA operators is being planned. All the examples used an equation-based approach for sample evaluation. In the future the evaluation engine will be based on electrical simulation, with additional design constraints allowing, for instance, detailed corners validation.

6. REFERENCES


Table 7. Specifications for the set of 9 additional tests.

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Input Variables</th>
<th>Domain input variables</th>
<th>Output</th>
<th>Specification</th>
<th>Transfer Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1º Order Low-Pass (1)</td>
<td>R₁, R₂, R₃, C₁, C₂</td>
<td>Rₛ∈[200:2000e3] Cₛ∈[1e-13:1e-11] Aₛ∈[12500:17500]</td>
<td>GainDC, F₀</td>
<td>GainDC = 45dB ± 1 Fₛ = 2MHz ± 0.16E6</td>
<td>$T(s) = -\frac{R₃ / R₁}{s + \frac{1}{R₃C₁} + \frac{1}{R₄ / R₃}}$</td>
</tr>
<tr>
<td>1º Order Low-Pass (2)</td>
<td>R₁, R₂, R₃, C₁, C₂</td>
<td>Rₛ∈[200:2000e3] Cₛ∈[1e-13:1e-10]</td>
<td>GainDC, F₀</td>
<td>GainDC = 45dB ± 1 Fₛ = 2MHz ± 0.16E6 Q = 0.5 ± 0.2</td>
<td>$T(s) = \frac{1}{s² + \frac{1}{R₃C₁} + \frac{1}{R₄ / R₃} + \frac{1}{R₅ / R₄} / R₅}$</td>
</tr>
<tr>
<td>2º Order Sallen-Keyes</td>
<td>R₁, R₂, R₃, C₁, C₂</td>
<td>Rₛ∈[10:2000e3] Rₛ∈[10e2:2000e4] Cₛ∈[2e-13:2e-8]</td>
<td>GainDC, F₀</td>
<td>GainDC = 20dB ± 1 Fₛ = 2MHz ± 0.16E6 Q = 0.4 ± 0.3</td>
<td>$T(s) = \frac{(R₅ + R₆) / R₆}{s² + \frac{1}{R₄ / R₃} + \frac{1}{R₄ / R₃} / R₅}$</td>
</tr>
<tr>
<td>Low-Pass Noch (1)</td>
<td>R₁, R₂, R₃, R₄, C₄, C₅, C₆</td>
<td>Rₛ∈[10000:6000e6] Cₛ∈[1e-13:1e-8]</td>
<td>GainDC, F₀</td>
<td>GainDC = 15dB ± 3 Q = ±0.2</td>
<td>$T(s) = \frac{s² + R₄ / (R₄C₆ + R₃R₄ / R₃)}{R₆}$</td>
</tr>
<tr>
<td>Low-Pass Noch (2)</td>
<td>R₁, R₂, R₃, R₄, C₄, C₅, C₆</td>
<td>Cₛ∈[1e-13:1e-11]</td>
<td>GainDC, F₀</td>
<td>GainDC = 48dB ± 1 Fₛ = 4.2kHz ± 0.25kHz Fₛ = 60kHz ± 1kHz</td>
<td>$T(s) = \frac{s² / R₄}{s² + \frac{1}{R₃C₆ + R₃R₄ / R₃} + \frac{1}{R₄ / R₃} / R₄}$</td>
</tr>
</tbody>
</table>

Table 8. Results, extracted from 100 runs executions, for the set of 9 additional tests.

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Total Fuzzy</th>
<th>N° Max of Generations</th>
<th>Average (GA Gen. Num)</th>
<th>Success Rate [%]</th>
<th>Cost Function Calls</th>
<th>CPU Time [seg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1º Order Low-Pass (1)</td>
<td>SGA</td>
<td>18</td>
<td>300</td>
<td>88</td>
<td>93</td>
<td>464 917</td>
</tr>
<tr>
<td>1º Order Low-Pass (2)</td>
<td>SGA</td>
<td>54</td>
<td>300</td>
<td>77</td>
<td>96</td>
<td>161 101</td>
</tr>
<tr>
<td>2º Order Low-Pass (1)</td>
<td>SGA</td>
<td>162</td>
<td>300</td>
<td>161</td>
<td>79</td>
<td>910 757</td>
</tr>
<tr>
<td>2º Order Low-Pass (2)</td>
<td>SGA</td>
<td>171</td>
<td>500</td>
<td>208</td>
<td>77</td>
<td>1 233 990</td>
</tr>
<tr>
<td>2º Order Sallen-Keyes (1)</td>
<td>SGA</td>
<td>90</td>
<td>500</td>
<td>231</td>
<td>94</td>
<td>1 337 303</td>
</tr>
<tr>
<td>2º Order Sallen-Keyes (2)</td>
<td>SGA</td>
<td>333</td>
<td>500</td>
<td>29</td>
<td>98</td>
<td>166 310</td>
</tr>
<tr>
<td>Low-Pass Noch (1)</td>
<td>SGA</td>
<td>972</td>
<td>500</td>
<td>126</td>
<td>97</td>
<td>793 934</td>
</tr>
<tr>
<td>Low-Pass Noch (2)</td>
<td>SGA</td>
<td>1 002</td>
<td>500</td>
<td>233</td>
<td>75</td>
<td>1 486 540</td>
</tr>
<tr>
<td>Low-Pass Noch (3)</td>
<td>SGA</td>
<td>7 563</td>
<td>600</td>
<td>258</td>
<td>69</td>
<td>1 622 188</td>
</tr>
</tbody>
</table>
