Research Article

Multiobjective Optimization Tool for a Free Structure Analog Circuits Design Using Genetic Algorithms and Incorporating Parasitics

Yaser M. A. Khalifa, Badar Khan, and Faisal Taha

Electrical and Computer Engineering Department, State University of New York, New Paltz, NY 12561, USA

Correspondence should be addressed to Yaser M. A. Khalifa, khalifay@newpaltz.edu

Received 27 April 2007; Revised 7 March 2008; Accepted 26 June 2008

Recommended by Giovanni Squillero

This paper presents a novel approach for a free structure analog circuit design using genetic algorithms (GAs). A major problem in a free structure circuit is its sensitivity calculations as a polynomial approximation for the design is not available. A further problem is the effect of parasitic elements on the resulting circuit’s performance. In a single design stage, circuits that are produced satisfy specific frequency response specifications using circuit structures that are unrestricted and with component values that are chosen from a set of preferred values including their parasitic effects. The sensitivity to component variations for the resulting designs is performed using a novel technique and is incorporated in the fitness evaluation function. The extra degrees of freedom resulting from unbounded circuit structures create a huge search space. The application chosen is an RLC ladder filters circuit design.

Copyright © 2008 Yaser M. A. Khalifa et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Computer-aided design tools for analog circuits consist almost invariably of circuit analysis or simulation packages; no general-purpose analog circuit synthesis tools are available. Conventional numerical optimization techniques can be used to obtain combinations of component values that satisfy a given design specifications. In addition, conventional optimization techniques operate on circuits after having their desired responses approximated by a suitable polynomial, using ideal components with unrestricted values, and only operating on predefined circuit structures. The restrictions introduced by predefining the circuit structure waste extra degrees of freedom in the design process and can conceal many novel design structures that can satisfy the target response and at the same time enhance the design process.

In previous papers [1–5], evolutionary computation algorithms are applied to fixed structure circuit designs to satisfy certain specifications. As mentioned earlier, the predefinition of the circuit structure required in conventional design techniques is a limiting factor when considering a powerful search technique such as genetic algorithm. A few attempts have been made to include structure synthesis in genetically-based optimization techniques such as GA [6, 7] and genetic programming (GP) [8]. However, these techniques suffered huge delays and memory usage and needed to be used in conjunction with circuit analysis packages such as PSPICE [9, 10].

There is still a need, however, for more efficient algorithms that are faster, self-sufficient and can function independently. An efficient measure of performance based on the sensitivity for component variations for the resulting circuits and the incorporation of the parasitic effects associated with components is still needed. This is particularly important in the development of new generations of analog programmable analog arrays (APAAs) programming tools. In this paper, an efficient GA technique for free structure analog circuit design is described. The technique incorporates a sensitivity measure to compare between potential solutions and incorporates parasitic effects.
2. GENETIC ALGORITHMS

The field of genetic algorithms [11–13] was founded by Holland in the early 1970s. In [12], Holland emphasizes the ability of simple representations to encode complicated structures and the power of simple transformations to improve that structure. These representations are combined in what is called in biology a chromosome (also referred to as individuals). A number of these chromosomes will constitute a population. Syntactic operations are then used to alter and improve these coded solutions.

The standard genetic algorithms invented by Holland are the basic form from which other improved algorithms have been subsequently developed. As mentioned earlier, the problem to be optimized has to be encoded into a chromosome-like structure in which each gene or set of genes represents a significant parameter towards the desired solution. A population would consist of a prespecified number of these chromosomes.

Starting from an initial randomly-generated population, a selection mechanism is used to identify and select members that exhibit better characteristics. Those selected chromosomes are then acted upon by genetic operators, such as Mutation and Crossover, which are explained in later sections, to reproduce and generate the subsequent generations. The procedure continues until either a time limit is reached or is achieved.

2.1. Crossover operator

The Crossover operator is considered to be the most important operator in genetic algorithms. It is also a special characteristic of genetic algorithms over many other search techniques that are inspired by the process of biological optimisation.

In nature, Crossover occurs when two parents exchange parts of their corresponding chromosomes. In genetic algorithms, the same idea forms the basis of the Crossover operator, where useful parts of different parent chromosomes should be combined in order to produce an offspring that benefits from the advantageous bit combinations of both parents. Two-point Crossover is an operator that randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring.

Consider the following two parents selected for Crossover. The "|" symbols indicate the randomly chosen crossover points:

- Parent 1: 110|010|10.
- Parent 2: 001|001|11.

After interchanging the parent chromosomes between the crossover points, the following offspring are produced:

- Offspring 1: 110|001|10.
- Offspring 2: 001|010|11.

2.2. Mutation

Mutation is a genetic operator that is used to change individual bits in the chromosome at a specified probability.

The motivation for using this operator is that it helps to introduce genetic diversity to new generations. While it is desirable in nature to keep the level of mutation low so that an individual produced by the mutation does not differ too much from its ancestor, this is not always the case in applications of genetic algorithms. Chosen mutation rates typically can range from 0.001 to 0.05. Rates beyond these limits are not recommended in most genetic algorithms applications.

3. IMPLEMENTATION

3.1. Chromosome adaptation and structure

In [1–3], the predetermination of the circuit structures has helped in reducing the chromosome length. This was achieved by incorporating the order of the chromosome genes into the node connections information. However, in the case of free structure circuit synthesis this technique cannot be used.

In the case of free structure design, the solution space is much larger and the optimization process is composite. It involves the optimization of two different but highly correlated problems; the structure as well as the sizing of a circuit. Hence, all specifications of the electrical circuit including the structure and the sizing of all the components should be included in the chromosome representation.

The specifications of the electrical circuit include the number of components in the circuit, the type of each component, and a list of connections between the components. Circuit components can include a variety of different types of components, including resistors, capacitors, and inductors.

Given the above information, the chromosome for each circuit comprises a number M of groups of equal bit lengths, and an extra group of bits representing the number of nodes in the circuit; see Figure 1.

![Figure 1: Chromosome structure.](image)

Each gene in the chromosome specifies a component and contains four fields (i) type of component (R, C, etc.), (ii) a pointer to a menu containing component value, (iii) and (iv) circuit nodes to which the component is connected; see Figure 1. Two-point crossover rate of 100% and a mutation rate of 0.5% were used for the GA operators.
FOR population_member = 0 to population_size, DO
  Determine N, number of nodes
  FOR gene_location = 1 to M, DO
    Decode fields 1, 2, 3 and 4
    Descale fields 3 and 4 according to N
    Update NAM
  END FOR
  FOR frequency_test_point = 0 to maximum, DO
    LU decompose NAM
    Calculate response
    Compare with template and find error
  END FOR
END FOR

Figure 2: Pseudocode for a single generation evaluation.

It is generally desirable for a low-cost design to use the least number of components. For this reason a Null component, representing an open circuit, is introduced into the component type menu.

3.2. The design program

On entry the user is required to supply the following input data.

Type of filter (low pass, high pass, band pass)

(i) $\omega_1$, $\alpha_{\text{min}}$.
(ii) $\omega_2$, $\alpha_{\text{max}}$.
(iii) $N$, the maximum number of nodes. Optional and can be used to limit the search to a certain number of designs.
(iv) $M$, the maximum number of components (optional).

The execution of the program starts by calculating $N$, maximum number of nodes. Then, genes representing elements are then decoded one at a time, splitting them into four fields as explained in Section 3.1.

Upon determining type, value and connection nodes, the component value is inserted in the corresponding nodal admittance matrix (NAM) location [14]. When all elements are decoded, contents of the resulting entries in NAM are passed to the calculation function, in which lower-upper (LU) decomposition and solutions to the matrix equations are solved. Figure 2 shows a pseudocode for the program.

4. LEAST-SQUARED ERROR EVALUATION

The error measured is defined as the difference between the specified response constraint, $H_{\text{Upp}}(\omega)$ and $H_{\text{Low}}(\omega)$, denoted by the shading in Figure 3, where $H_{\text{Upp}}(\omega)$ is the upper boundary of the frequency response, and $H_{\text{Low}}(\omega)$ is the lower boundary, and the actual response, $H_a(\omega)$, from the design represented by GA chromosomes. The actual frequency response, $H_a(\omega)$, will hence be prescribed by its unique combination of component values. Therefore,

$$H(\omega, R_1, R_2, \ldots, R_m, C_1, C_2, \ldots, C_n, L_1, L_2, \ldots, L_v).$$

For simplicity, all types of components will be declared as $k$. Thus $H$ becomes

$$H(\omega, k_1, k_2, \ldots, k_q),$$

where $q = m + n + v$.

Hence, the error function will depend on the region it is applied for. For the passband the error function would be

$$\text{err}(\omega, k_1, k_2, \ldots, k_q) = H_{\text{Low}}(\omega) - H_a(\omega, k_1, k_2, \ldots, k_q),$$

while in the stopband the error function would be

$$\text{err}(\omega, k_1, k_2, \ldots, k_q) = H_{\text{Upp}}(\omega) - H_a(\omega, k_1, k_2, \ldots, k_q).$$

The least-squares error criterion $E$ is used as a performance index because of mathematical simplicity and ease of programming. The error is defined as

$$E = \int_{u}^{l} \text{err}(\omega, k_1, k_2, \ldots, k_q).$$

Substituting from (4) in (5), LSE would be

$$\text{LSE} = \int_{u}^{l} \left[H(\omega) - H_a(\omega, k_1, k_2, \ldots, k_q)\right]^2.$$

where $H(\omega)$ is the boundary response $H_{\text{upp}}(\omega)$ or $H_{\text{low}}(\omega)$. Since the calculations must be performed numerically, the integration in (6) will be replaced by summation. Therefore,

$$\text{LSE} = \sum_{i=1}^{m} [H(\omega) - H_a(\omega, k_1, k_2, \ldots, k_q)]^2,$$

(7)

where $m$ is equally spaced frequency points assumed over the range of integration. This is chosen because of ease in programming.

4.1. Specified weighted errors

Equation (7) gives equal weights for all frequency points within the range. This does not necessarily reflect the prescribed performance, for example, in the case of numerically small errors in the passband. This numerically small error in the passband will be more likely to affect the accuracy of a solution than a relatively larger error encountered in the stopband. Hence, a weight function can be inserted in (7) to enable us to emphasize or de-emphasize the error in the various regions of the frequency spectrum. Therefore, (7) can be modified to be

$$\text{LSE} = \sum_{i=1}^{m} \{w(\omega) \cdot [H(\omega) - H_a(\omega, k_1, k_2, \ldots, k_q)]\}^2.$$

(8)

The weight function $w(\omega)$ has been arbitrarily chosen to be two in the passband region and unity in the stopband region. Any other design weight functions can be easily applied instead.

Although the passband will be characterized by a specific passband ripple, the upper boundary of the response represents the maximum power transfer level, the claimed results of LC filter theory. Therefore, testing only the suitability of the lower limit of the response would be sufficient, as the upper limit will never be exceeded in the passband.

In the stopband region, only the upper limit of the specified attenuation level needs to be considered, since any lower value of attenuation meets the specifications. The transition region is considered as a “do not care” region. Thus, it is sufficient to test just the passband and the stopband for confirmations. The different frequency band boundaries are shown in Figure 3. The fitness is taken as the reciprocal of the error function.

4.2. Performance improvements by controlling connection fields

The number of nodes comprising the circuit is determined at the beginning of the decoding process to control the connection node numbers of the circuit components.

The fields determining the connection nodes are decoded in a way to produce node numbers within $N$. This is achieved by linear-scaling down the whole range of values of allowed node numbers to only that of $N$. This helps to avoid circuit structures that would have components connected to nodes that do not exist in the defined structure and which are hence unacceptable and would be a total waste of time to evaluate.

5. Sensitivity comparison

In [5], the author has developed a rapid feasible region estimation technique using GA. The technique is modified to provide a measure of comparison among the resulting circuit designs using the free structure GA technique described in the previous sections.

A hierarchical approach for circuit design using GA is implemented. Once convergence occurs and a family of solutions emerges, usually 4–6 in average, the structures of these circuits are fixed. For this stage, a uniform closely spaced distribution of component values is used to help determining estimates for near and far limits of component values within which the required circuit specifications are met. At the end of this stage, the least minimum $x_i^{\text{min}}$ and largest maximum $x_i^{\text{max}}$ boundaries for each of the parameters are taken to be the estimate of the feasible region for that specific parameter. A measure for the transfer function sensitivity to variations in component $x_i$ is

$$\Delta x_i = \frac{x_i^{\text{max}} - x_i^{\text{min}}}{x_i};$$

(9)

and for $m$, the total number of components in the design is

$$S^p = \sum_{i=1}^{m} \Delta x_i.$$

(10)

The new fitness function will result from multiplying $S^p$ by its current value.

6. Extension to include parasitic effects

Parasitics can include wire and winding resistances and inductors in lumped circuit components as well interlayer capacitances and node to ground voltages. It can also include resistances associated with polysilicon and doped semiconductor regions when used as conductors, and capacitances associated with any crossover, from any conductor to substrate, and with any depletion region in a reverse-biased pn junction, in integrated circuits. Careful layout design can eliminate some of these parasitics, but avoiding them all together is not feasible. Other factors which can also affect the amount of parasitics are the area of the circuit and the packaging density of the chip [15].

Parasitics can cause significant deviations in circuit performance and should be accounted for in early stages of the design, if their effects are to be minimized. In this chapter, GAs, as described earlier, are modified to include some additional information the designer might want to consider for each circuit component. Parasitic effects considered in this paper include inductor resistance as shown in Figure 4, and series and parallel resistances, and series inductance associated with a capacitor as shown in Figure 5.

To implement the parasitic effects into the circuit design optimisation problem using the GA, an additional parameter is added into the component menu and this represents the parasitic effect of the associated component and the quality factor $Q$; see Figure 6.
This method will avoid adding to the complexity of the structure and will make it possible to add or remove any effect from the component menu without having to alter the chromosome structure.

In the implementation considered in this work, arbitrary loss factors for components are assumed. The technique is based on the idea to perturb the element values of the lossy filter with the aim of making the magnitude function of the lossy filter proportional to that of the ideal filter at some discrete frequencies in the passband and the stopband. There is no restriction on the $Q$'s of the elements, provided that they are high enough, and the elements need not have identical $Q$'s.

However, the methods outlined need preknowledge of the values of the $Q$'s for each element, which might not be feasible in practical situations, where designers quite commonly depend on statistical distribution data for their $Q$ values. It is still feasible to specify classes of $Q$'s, for example, low, medium, and high; but these will reflect a band of randomly distributed values for each class rather than specific values.

It is very likely that the designer will decide on what class of $Q$'s is needed for a circuit design rather than specific $Q$ values. For this reason, the method used in this work assumes a random distribution of $Q$ values within each class of $Q$'s; see Figure 6.

### 6.1. Incorporation of parasitic effects in circuit designs

In implementing the parasitic effects into the circuit design optimisation problem using the GA, two methods can be applied. The first is to include whatever effects required to be considered in the chromosome structure itself; see Figure 7.

This has the following limitations, firstly it will add to the complexity of the gene structure and the GA operation. This will eventually put limits on the number of additional effects a designer might be able to consider. In addition to that, inserting or removing an effect will require a change in the chromosome structure. Last but not least, effects considered for each type of elements are different and hence different gene lengths will be required to represent each type of elements.

The second method to represent any parasitics into the GA implementation is to associate these effects within the menu listings of elements. Additional effects can then be introduced and any further calculations of these effects can then be made, for example, loss factors; see Figure 8.
This method will avoid adding to the complexity of the chromosome structure and will make it possible to add or remove any effect from the component menu without having to alter the chromosome structure.

7. RESULTS

The GA used has a crossover rate of 100% and a mutation rate of 5% throughout all runs. The GA was allowed to
run for one thousand generation in each run. However, convergence was reached after an average of one hundred and fifty generations. Each generation had fifty chromosomes or individuals. Each generation took 10 seconds on average in processing. In these implementations, the desired design template is the starting point of the design process, whereas in conventional designs polynomial approximation for the desired response is used [15]. This has the limitation of constraining the feasible solution to sets of design parameters that would satisfy the polynomial approximation. By designing directly from the template specifications, a designer would avoid extra degree of approximation a polynomial fitting would introduce. At the same time, it provides an extra degree of freedom which would enlarge the solution space considerably to include any design that would reside within the design template without necessarily being represented by a polynomial. Also the GA search starts from an arbitrary point in the solution space and needs not to have any prior knowledge of the feasible region of the required design.

7.1. Low-pass design

Different low-pass design specifications were considered. One had a passband ripple of 1 dB, minimum attenuation of $-100$ dB at a transition ratio of 10. An average group of five solutions emerged; the frequency response of one of these solutions is plotted in Figure 9. The circuit constructed by the GA is shown in Figure 10. It is noticed that well-known π and T ladder structures have also been generated by the GA.

A different design specification of 0.5 dB passband ripple and $-60$ dB stopband attenuation was tested. Figure 11 shows the resulting circuit design.

In a different run, two circuits that were designed to meet achieve a maximum attenuation of 1 dB in the passband and a minimum attenuation of $-100$ dB in the stopband; see Figure 12, Table 1. A comparison between the two circuits was conducted based on the tolerance rage to changes in component values. Figure 9 shows the two circuits used to test the GA program.

Even though the two circuits have a very close components values, the range generated by the GA program was different. The differences in ranges for each component in the circuits are shown in Table 2. We also note that C2 and C5 in circuit (a) are in parallel, which is equivalent to C2 in circuit (b).

As shown in Table 2, the two circuits vary with the difference at some components, but in general, circuit (a) has a higher tolerance range for its component values than circuit (b). This gives circuit (a) fitness leads over circuit (b).
Table 1: (a) Table shows components, nodes, ranges, and actual values for each component in circuit a. (b) Table shows components, nodes, ranges, and actual values for each component in circuit b.

<table>
<thead>
<tr>
<th>Component</th>
<th>Circuit 1</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual value</td>
<td>Nodes 1</td>
</tr>
<tr>
<td>C1</td>
<td>1.80</td>
<td>3</td>
</tr>
<tr>
<td>C2</td>
<td>2.20</td>
<td>2</td>
</tr>
<tr>
<td>C3</td>
<td>2.20</td>
<td>1</td>
</tr>
<tr>
<td>L1</td>
<td>2.20</td>
<td>1</td>
</tr>
<tr>
<td>L4</td>
<td>1.00</td>
<td>2</td>
</tr>
<tr>
<td>L3</td>
<td>1.00</td>
<td>2</td>
</tr>
<tr>
<td>C5</td>
<td>2.70</td>
<td>2</td>
</tr>
<tr>
<td>R1</td>
<td>1.00</td>
<td>3</td>
</tr>
<tr>
<td>L2</td>
<td>2.20</td>
<td>3</td>
</tr>
<tr>
<td>C6</td>
<td>1.00</td>
<td>3</td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>Component</th>
<th>Circuit 2</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual value</td>
<td>Nodes 1</td>
</tr>
<tr>
<td>L4</td>
<td>1.00</td>
<td>3</td>
</tr>
<tr>
<td>R1</td>
<td>0.10</td>
<td>3</td>
</tr>
<tr>
<td>C2</td>
<td>1.80</td>
<td>2</td>
</tr>
<tr>
<td>C1</td>
<td>1.20</td>
<td>3</td>
</tr>
<tr>
<td>C3</td>
<td>2.20</td>
<td>1</td>
</tr>
<tr>
<td>L2</td>
<td>2.20</td>
<td>2</td>
</tr>
<tr>
<td>L1</td>
<td>4.70</td>
<td>1</td>
</tr>
<tr>
<td>R2</td>
<td>0.15</td>
<td>3</td>
</tr>
<tr>
<td>L3</td>
<td>1.00</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2: Differences between circuits’ components.

<table>
<thead>
<tr>
<th>Component</th>
<th>Circuit b</th>
<th>Range</th>
<th>Difference</th>
<th>Circuit a</th>
<th>Range</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>L4</td>
<td>2.20</td>
<td>2.27</td>
<td>0.07</td>
<td>L4</td>
<td>2.20</td>
<td>2.72</td>
</tr>
<tr>
<td>R1∥R2</td>
<td>0.06</td>
<td>0.32</td>
<td>0.26</td>
<td>R1</td>
<td>0.48</td>
<td>1.40</td>
</tr>
<tr>
<td>C2</td>
<td>1.80</td>
<td>2.32</td>
<td>0.52</td>
<td>C2∥C5</td>
<td>1.20</td>
<td>1.47</td>
</tr>
<tr>
<td>C1</td>
<td>0.77</td>
<td>1.57</td>
<td>0.80</td>
<td>C1∥C6</td>
<td>0.35</td>
<td>0.89</td>
</tr>
<tr>
<td>C3</td>
<td>1.68</td>
<td>2.44</td>
<td>0.76</td>
<td>C3</td>
<td>2.20</td>
<td>2.72</td>
</tr>
<tr>
<td>L2</td>
<td>0.57</td>
<td>1.36</td>
<td>0.79</td>
<td>L2</td>
<td>1.00</td>
<td>1.52</td>
</tr>
<tr>
<td>L1</td>
<td>4.70</td>
<td>5.22</td>
<td>0.52</td>
<td>L1</td>
<td>1.68</td>
<td>2.65</td>
</tr>
<tr>
<td>L3</td>
<td>0.62</td>
<td>1.23</td>
<td>0.61</td>
<td>L3</td>
<td>0.50</td>
<td>1.09</td>
</tr>
</tbody>
</table>

Average difference 0.54 Average difference 0.61

7.2. High-pass design

The high-pass filter design is done directly without having to use a low-pass prototype and then performs frequency transformations. A design template of 1 dB passband ripple and attenuation of −60 dB at the stopband edge with a transition ratio of 10 is considered. A number of four solutions on average emerged two of these circuits are shown in Figure 13. The frequency responses of the resulting circuits are shown in Figure 14.

8. Conclusion

This paper has presented a novel circuit design tool by using the GA strong search abilities in both the structure and sizing domains of a circuit design problem. A successful design of the chromosome structure has been obtained to include
the full circuit specifications. A novel technique to compare circuit sensitivities to changes in component values among different circuit structures was developed.

It is noticed that well-known π and T ladder structures have also been generated by the GA. That is due to the low sensitivity such structures have, to small changes in component values. This feature was detected by the GA as a result of the sensitivity comparison technique developed in this paper.

So far, the technique described in this paper is limited either passive circuits, or active circuits that are based on passive models. A further extension to include active circuits and three terminal components is underway and will appear shortly in another paper.

REFERENCES


