Design Automation of Analog Filters via Evolutionary Algorithms

Robert Buckingham, Lisa Myers, Paul Giampaolo, Joseph Misiti
University of Pittsburgh, School of Engineering
Department of Electrical and Computer Engineering
Pittsburgh, PA 15261

ABSTRACT

In this paper, we propose the utilization of evolutionary algorithms (EA) for automating the design process for analog filters. The design of analog filters is a time consuming and expensive process, which is constrained by traditional design methodologies. EAs are a tempting choice as they have shown great potential as an optimization tool in nature. They also have the advantage of requiring no circuit knowledge. This allows the algorithm to explore options that would not typically arise in traditional design techniques.

1 Introduction

It has been shown that evolutionary algorithms are an effective design tool for optimization of component values within a predefined, fixed topology [1,2]. The approach of allowing component values to evolve in parallel with topology is attractive for various reasons. This allows the algorithm to evolve the circuit without any bias from traditional design methodologies. This, in turn, allows for the possibility of better solutions. Traditional circuit design generally uses a top-to-bottom approach. Restricting EAs to use traditional topologies may hinder them if the topologies are not easily improved through evolution's bottom-to-top design approach [3].

In this paper we present a method for storing information to describe an arbitrary circuit in genetically modifiable form and two genetic algorithms for accomplishing circuit evolution. Key aspects of these algorithms are their abilities to evolve topologies appropriate to solution criteria and to explore further topologies to meet the criteria with a reduced amount of components.

By automating the design process of analog filters in this fashion, time and money can be saved. Not only can the design time be reduced, but the resulting evolved circuit could produce a more efficient design, both financially and electronically.

The remainder of the paper is structured as follows: Section two outlines the overall flow of the evolutionary design process being used. Section three details the two specific EAs and the theory behind them. Section four discusses convergence issues. Section five covers our methods of element reduction. Section six presents results derived from these algorithms. Finally, our conclusions are presented in Section seven.

2 Evolutionary Design Process

The design process is divided into five phases. Each phase contains a specific aspect of the overall functionality of the process. The interaction between these phases is shown in Figure 1.

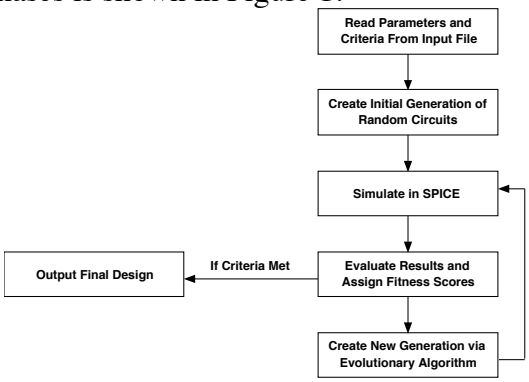

Figure 1: Evolutionary Process Flow

2.1 Creation of Initial Generation

In order to create our initial generation of circuits we must first have a method with which we can represent any given circuit. This method should also be conducive to genetic manipulation. The method we have chosen is for each circuit to be represented by a collection of elements. Each element contains four fields, three discrete and one real (see Figure 2). The first field is the element type. This is represented discretely and in addition to the specific element types it can code for a 'null' element, meaning that it codes for nothing. The second field is real valued and contains the value of the component. The final two fields are both discrete and contain the node connections for the element. The ability to have the type field code for a null element is advantageous from an algorithmic point of view as it allows each circuit to contain the same number of element objects yet still have a different number of actual elements within the circuit.

Type	Data	Node 1	Node 2

Figure 2: Element Representation

The initial generation is filled with random circuits. However, merely randomizing elements is insufficient to create a viable initial generation. Before a circuit is declared valid, it has to meet certain criteria. These criteria are used to ensure that the circuit is valid and can be simulated in SPICE.

2.2 Fitness Calculations

After circuit simulations are complete, the fitness of the circuits is calculated. This calculation is done by comparing the simulation results with the design goals. This calculation can be seen in Figure 3.

$$\frac{\sum |R - C|}{N}\left[1 + .005E\right]$$

Figure 3: Fitness Equation

With this formula, the fitness score is calculated by taking the summation of the difference of the Results[R] and the Criteria[C] and dividing them by the number of data points. This result is multiplied by a factor which is directly proportional to the number of elements [E] in the circuit. This factor is used to give circuits with a smaller number of elements better fitness scores.

2.3 Reproduction

The next phase that occurs is Reproduction. This phase involves mating two different circuits to generate a new circuit, which may contain attributes from both of its parents. Reproduction is very useful because mating two good parent circuits can result in an even better child circuit.

3 Algorithms

Our program uses two EAs: globalEvolution and oneDRegionalEvolution. The core of each algorithm (recombination and mutation) is the same. The differences occur in their approach to selection of parents and reinsertion.

3.1 globalEvolution

Our first EA, globalEvolution derives its name from its method of parental selection. The pool of potential parents consists of the entire generation. It uses tournament selection to select the two parents. Tournament selection is the process of randomly selecting two potential parents out of the entire generation and selecting the circuit with the lower fitness score. The equation showing the probability of being a parent based on rank within the population is shown in Figure 4.

$$p(X) = \frac{1 + 2(n - X)}{n^2}$$

Figure 4: Tournament Selection

After the parents have been selected and have mated, reinsertion occurs. This is the method by which circuits are inserted into the next generation. globalEvolution uses a method known as Elitism. This technique involves copying a certain percentage of the performers unchanged from the previous generation. This guarantees that the best performers of a given generation are at least as good as those from the previous generation. In globalEvolution, the top 9% are copied to the following generation.

3.2 oneDRegionalEvolution

oneDRegionalEvolution is the second EA used. This algorithm differs from globalEvolution in the method for selection of parents and reinsertion. The parental selection in this algorithm uses regional modeling. Specifically, this means that the circuits can be viewed as taking up discrete locations. These locations are spread across a one-dimensional field. The possible parents for a given offspring can originate only from within a certain range of the location the offspring is to be produced at. In oneDRegionalEvolution, this range is 5% of the generation size to either direction. Once the pool of parents is identified, the actual parents are chosen using tournament selection in the same way as that used in globalEvolution.

The reinsertion phase in oneDRegionalEvolution uses regional elitism. This means that if a circuit is the top performer in its local region, it gets copied unchanged to the next generation.

3.3 Recombination

Once the parents are chosen there are three possible methods of recombination. The first traverses through the parent's genome one element at a time with three possible options: copy parent 1's element to the offspring, copy parent 2's elements to the offspring, or create the offspring's element by crossover between the parents' elements. This is illustrated in Figure 5.

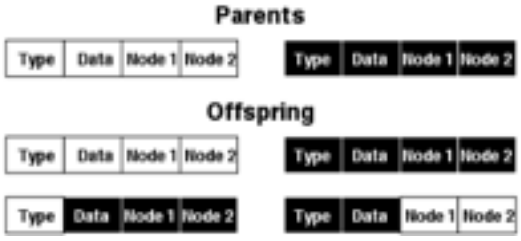

Figure 5: Recombination Method 1

The second method used is similar to the first. In this method the offspring's genome is created one field at a time. The field can

either be one of the parents', or result from crossover between the parents' fields. This is illustrated in Figure 6.

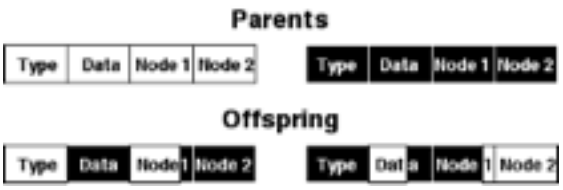

Figure 6: Recombination Method 2

The final recombination method differs significantly from the previous two. This method will randomly choose a node in each of the parents and then create a new circuit by simply tying the two parents together at the chosen nodes. This was done to facilitate improvement by providing a method by which evolved 'modules' could be reproduced. This ability was shown to be advantageous in [6]. This method is illustrated in Figure 7.

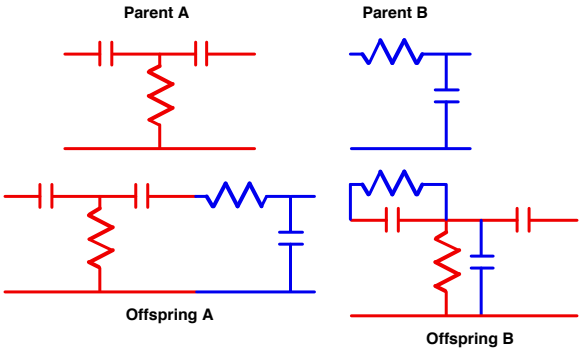

Figure 7: Recombination Method 3

3.4 Mutation

After an offspring is created by recombination, mutation can occur. The mutation rate is a per element rate. If mutation occurs for a given element, it will affect one of its four fields. The field affected is chosen randomly. Once the field to be affected is known, the type of mutation is then randomly determined. Using two types of mutations was shown to be advantageous in [4]. The first type, a small mutation, has 2/3 chance of occurring. A small mutation involves bit-flipping for discrete fields and a small percentage change for real valued fields. The second mutation method is used for large mutations and has a 1/3 chance of occurring. This method simply involves setting the affected field to a random value.

4. Convergence Issues

Often when using EAs, the issue of convergence to local minima becomes problematic. When this occurs the generation's genetic diversity quickly plummets. As a result, all members of the population become virtually identical. Consequently, sexual reproduction, which is the core of the genetic algorithms, ceases to be an effective method of optimization. Three methods have been employed in an effort to counteract this problem.

1. Dynamic Mutation Rate

Once the population has begun to converge, mutations account for the bulk of the genetic diversity. The normal mutation rate is not intended as a primary means of increasing diversity. A larger mutation rate will raise variability in the population. However, it will also hinder the efficacy of sexual reproduction. Therefore, a large mutation rate is only beneficial as a tool to stimulate a stagnating population. In an effort to reap the benefits of both a low and high mutation rate, the dynamic mutation rate was introduced as in [5].

After a certain number of generations pass without any improvement by the top performer, the mutation rate begins to increase in a piecewise linear fashion until it reaches a threshold of 20 time the normal mutation rate. Increasing the multiplier

higher than this raises the mutation rate to a point where it effectively randomizes all offspring. Figure 8 illustrates how the mutation rate changes as a function of the number of generations without improvement.

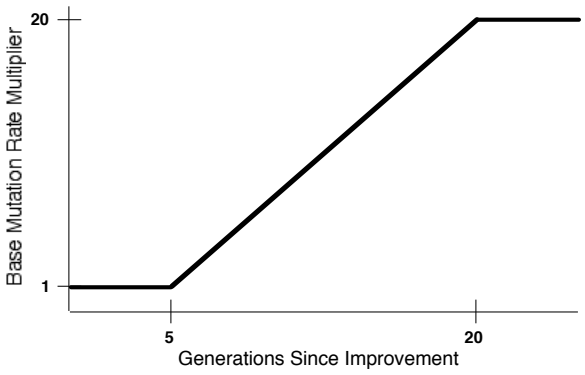

Figure 8: Dynamic Mutation Rate

2. Random Circuit Injections

Although the dynamic mutation rate is effective, it cannot always prevent the convergence to a local minimum. If over a set number of generations, the top performer fails to improve by a specified amount, it can be concluded that the dynamic mutation rate is insufficient to prevent stagnation of the population in this instance. At this point, the population is assumed to have very little remaining diversity.

Random circuit injections are used to reintroduce genetic variability. This process inserts randomly generated circuits into the population in place of any circuit which would normally be produced by reproduction. The circuits that are considered elite by the EAs are still copied unchanged in the usual fashion. This has a net effect of instantly escalating the population's genetic diversity.

3. Simulation Resetting

While the previous two methods often solve the convergence problem, occasionally a more radical approach is required. If enough generations have passed to allow for multiple injections and the top performer has yet to make any substantial improvement, the simulation is reset. In this case *all* circuits are replaced with randomly generated ones, including ones deemed elite. Although this is a drastic approach, and must be used sparingly, it will allow the population to again utilize sexual recombination as an optimization tool.

5. Element Reduction

While the EAs will eventually provide a solution, they are only concerned with the user specified criteria. Often they will produce a circuit that, although it meets the criteria, is much more complex than necessary. Reduction of the number of elements is required if this method is to produce circuits that are of use in a commercial environment. Two methods have been implemented to facilitate element reduction.

5.1. Fitness Penalty

The fitness penalty reduces elements continuously throughout the simulation. It accomplishes this by applying a penalty to a circuit's fitness score proportional to the number of elements it uses. This is shown in the fitness score calculation in Figure 3. If a circuit improves, but uses more elements, it is not considered superior unless it improves by at least .5% per element added. Conversely, a circuit that is a worse performer can be considered an improvement if it reduces the number of elements used.

5.2. Tuning Generations

Tuning generations attempt to reduce the number of elements used once a circuit has been produced that meets all user specified criteria. The tuning generations are simply a

number of generations that continue after a solution has been found. During these generations, circuits that meet all specified criteria are ranked according to the number of elements they contain, with the circuit using the fewest being ranked highest. This method expands the criteria to include number of elements as a major factor.

6. Results

Utilization of EAs for analog filter synthesis has proven itself as a viable design tool. The EAs, in conjunction with the element reduction techniques discussed in section 5, have been able to successfully produce filters meeting various criteria with reasonable numbers of components. They have also proven themselves useful for designing filters that are difficult to produce using traditional methodology.

6.1. Low Pass Filter

This is a simple low pass filter design. The filter requirements specified a pass band up to 1kHz with +/- 1dB of ripple and a stop band starting at 10kHz with at least –40dB of attenuation. The resulting filter and it's bode plot are shown in Figure 9 and Figure 10.

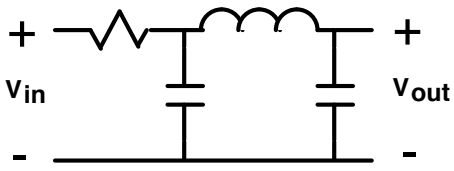

Figure 9: Low Pass Filter

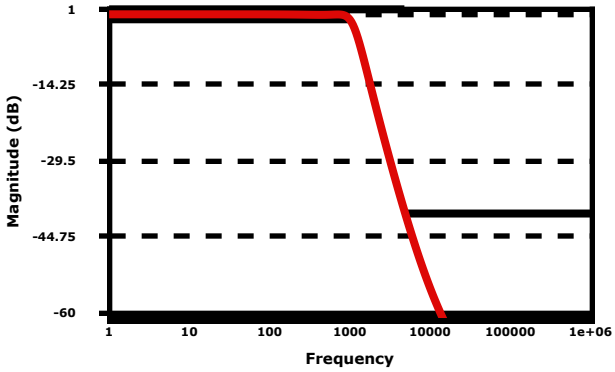

Figure 10: Low Pass Filter Bode Plot

6.2. High Pass Filter

This is a simple high pass filter design. The filter requirements specified a stop band up to 1kHz with at least 80dB attenuation and a pass band starting at 10kHz with no more than +/- 1dB of ripple. The resulting filter and it's bode plot are shown in Figure 11 and Figure 12.

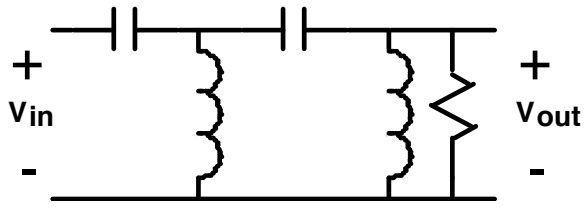

Figure 11: High Pass Filter

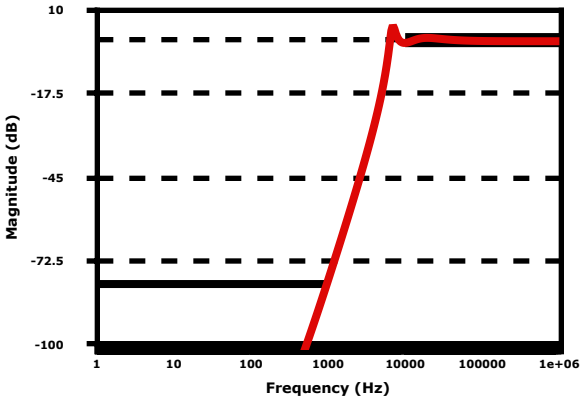

Figure 12: High Pass Filter Bode Plot

6.3. Band Stop Filter

This is a straightforward band stop filter. The stop band is from 10kHz – 100kHz and requires at least 40dB of attenuation. The

transitions regions are a decade wide to either side and +/- 1dB of ripple is allowed in the pass bands. The circuit and it's bode plot are shown in Figure 13 and Figure 14.

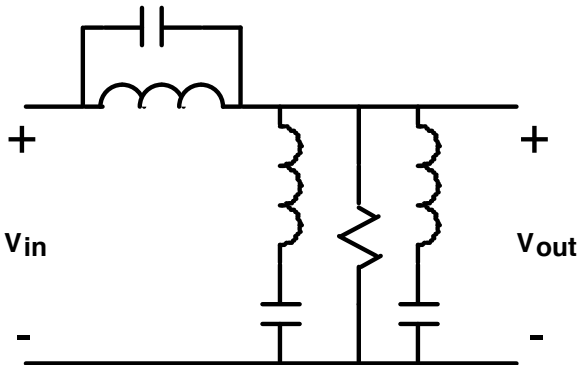

Figure 13: Band Stop Filter

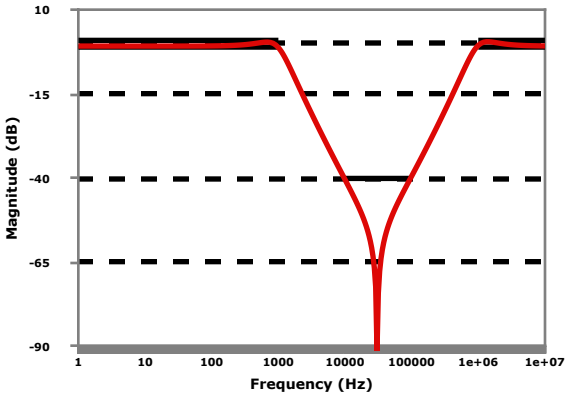

Figure 14: Band Stop Filter Bode Plot

6.4. Asymmetric Band Stop Filter

The next filter developed is an asymmetric band stop filter. This filter has a pass band to 10 Hz with +/- 2dB of ripple allowed. The stop band is from 100Hz – 10kHz with at least 20dB of attenuation. The second pass band is an attenuated pass band. It starts at 100kHz and is allowed to be between –8dB and –12dB. The filter and its associated bode plot are shown in Figure 15 and Figure 16.

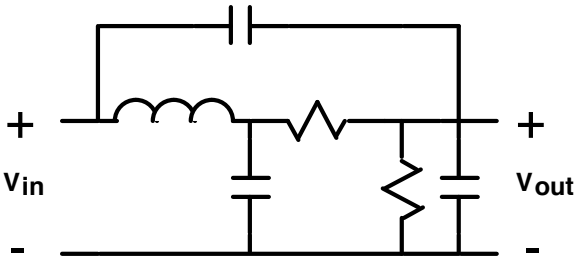

Figure 15: Asymmetric Band Stop Filter

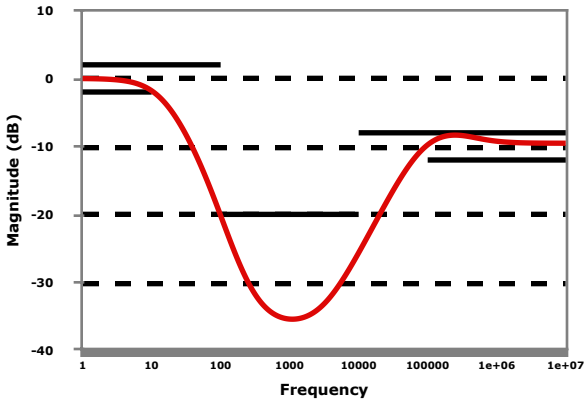

Figure 16: Asymmetric Band Stop Filter Bode Plot

7. Conclusions

The ability for EAs to develop topology in conjunction with elements values greatly increases their utility. This allows for new and potentially improved designs that would not otherwise be explored. The EAs have shown the ability to develop appropriate topologies with realistic amounts of elements.

Based upon the results of our work it seems feasible that EAs will be useful for other aspects of analog design. Based on our results, we feel the ability of these algorithms to scale upward to more complex problems, including the incorporation of three terminal devices, is worth exploring.

References

[1] David H. Horrocks and Mark C. Spittle, "Component Value Selection for Active Filters Using Genetic Algorithms," Proceedings of the

IEE Workshop on Natural Algorithms in Signal Processing, vol. 1, no. 1, pp. 131-136, 1993.

[2] Kenneth V. Noren and John E. Ross, "Analog Circuit Design Using Genetic Algorithms," Second Online Symposium For Electronics Engineers, Summer 2001.

[3] A. Thompson and P. Layzell, "Analysis of Unconventional Evolved Electronics," Communications of the ACM, vol. 42, no. 4, pp. 71-79, 1999.

[4] J. E. Grimbleby, "Automatic Analogue Circuit Synthesis Using Genetic Algorithms," IEE Proceedings: Circuits, Devices, and Systems, vol. 147, no. 6, pp. 319-323, 2000.

[5] S. Uyar, G. Eryigit, and S. Sariel, "An Adaptive Mutation Scheme in Genetic Algorithms for Fastening the Convergence to the Optimum," The Third Asia Pacific International Symposium on Information Technology, pp. 461-465, 2004.

[6] John R. Koza, Martin A. Keane, and Matthew J. Streeter, "The Importance of Reuse and Development in Evolvable Hardware," Nasa/ DoD Conference on Evolvable Hardware, pp. 33-42, 2003.