International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:2, 2008

Analog Circuit Design using Genetic Algorithm: Modified

Amod P. Vaze

Abstract**—**Genetic Algorithm has been used to solve wide range of optimization problems. Some researches conduct on applying Genetic Algorithm to analog circuit design automation. These researches show a better performance due to the nature of Genetic Algorithm. In this paper a modified Genetic Algorithm is applied for analog circuit design automation. The modifications are made to the topology of the circuit. These modifications will lead to a more computationally efficient algorithm.

Keywords**—**Genetic Algorithm, Analog circuits, Design.

I. Introduction

ANALOG circuits form an important part of mixed-signal integrated circuit and remain very important in high-speed applications such as communications. Analog circuit synthesis is very challenging, and has traditionally been performed by specialists who have a wealth of experience and intuition. It takes a lot more time than the easier chunks of circuit. The analog circuit building involves selecting a candidate topology meeting the requirements. This is due to the need to manually iterate circuit parameters (e.g. component values and transistor sizes) to meet specifications. There has been development in automating analog circuit design using optimization algorithms. A particular optimization algorithm that has been applied to the task of automating analog circuit synthesis is the Genetic Algorithm.

This problem can be classified as an optimization problem consisting of many parameters. Efforts with gradient search methods result in the need of initial guesses and may get stuck in local maxima. This paper asserts genetic algorithms (GA's) [1, 2, 3, 4] may be a better alternative for global optimization tools for automated design of analog circuits.

The goal of this paper is to use Genetic Algorithm to design analog circuits. The current algorithm is modified with respect to population size and the generation of valid circuits only. Current literature [5, 6, 7, 8] in the analog circuit design field looks only at linear circuits. The methods may generate invalid topologies and hence increase the computations. Also, higher population sizes are required. Koza [9] has performed non-linear circuit designs using genetic programming but emphasized topology generation. In this paper, simple modifications are introduced which reduce the computations required. Hence, we acquire a circuit in lesser generations.

The rest of the paper is organized as follows: in the next section we will review the related work on Genetic Algorithm and its application in analog circuit design automation. In the third section we state our algorithm for using Genetic Algorithm in circuit design and in the fourth section we will evaluate our method and some remarks about our method will be explored in last section.

Amod P. Vaze is a final year student of Bachelors degree Program in Electronics Engineering at the K. J. Somaiya College of Engineering, Vidya Vihar (Mumbai University), Mumbai 400077 India (phone: 91-9869111061; e-mail: amvaz2006@ yahoo.com).

II. Related Work

A. Genetic Algorithm

Genetic algorithms are heuristic optimization methods whose mechanisms are analogous to biological evolution [10]. A good general introduction to genetic algorithms is given in [3]. In Genetic Algorithm, the solutions are called chromosomes. The initial population is generated randomly. Selection and variation function are executed in a loop until some termination criterion is reached. Each run of the loop is called a generation. The selection operator is intended to improve the average fitness of the population by giving individuals of higher fitness a higher probability to be copied into the next generation. The quality of an individual is measured by a fitness function.

B. Usage of Genetic Algorithm in Analog Circuit Design Automation

As already stated, Genetic Algorithms have been used extensively in analog circuit design automation [5,6,7,8]. The basic principle of their application to analog circuit design involves representing the circuits as chromosomes and with component parameters as genes. The topology is created with the help of a skeleton topology kept ready on which the other components incorporate themselves. Koza [9] uses Genetic Programming to incorporate topology generating and modifying functions. The skeleton is kept very primitive here and more emphasis on self-generation is given importance. The invalid circuits generated are checked for and pruned. Fitness functions involve the usage of simple weighing functions, the weights deciding on which parameter is to be improved and which one to be suppressed. Selection and variation functions are used to generate better circuits and after the required fitness levels are achieved, the process is halted.

III. Genetic Algorithm Application in analog circuit design

A. Problem Description

Analog Circuit design using Genetic Algorithms involves representing the circuit completely. An analog circuit can be completely defined with the components and their values stated. Also, the nodes between which they are connected is essential. None of the circuit component should be floating else it will result in invalid circuit leading to a waste of that chromosome. So, the problem involves encoding the above said parameters into the chromosome. Each gene represents a component of circuit or any parameter. This chromosome, hence, represents our possible solution to the stated requirements. The fitness value of the circuit has to be measured. The fitness evaluation requires the usage of simulation software which can give us the required values for measuring its fitness.

B. Modification to the Topology

A modification to the existing topology has been made which guarantees valid circuits. This modification involves the connection of all nodes except ground i.e. first node connected to second node and second to third and so on. These nodes are connected with a resistor of very large value. The input node is connected to the source with source resistance and output is taken across a load resistor. This takes care of connection with the ground. Hence, now there are no floating nodes in the circuit. This leads to 100% efficiency of the population and also we eliminate the need to check for any invalid circuit. Thus, we speed up the algorithm i.e. the circuit with required fitness is built in lesser number of generations. Lesser number of generations result because the population does not consist of invalid circuits.

C. Fitness Function

Fitness is a single numerical quantity describing how well an individual meets predefined design objectives and constraints. Fitness can be computed based on the outputs of multiple analyses using a weighted sum. The definition of good fitness functions is highly problem dependent. The SPICE program is used to evaluate the fitness of each chromosome. The function is defined such that the superior individuals have the lowest fitness values. Using these definitions, the raw and standard fitness defined by Koza [4] are identical.

A raw fitness metric for minimizing an output variable c_i computed at N points can be defined as

$$f = \sum_{i=1}^{N} | c_i |$$

Other metrics can also be defined to maximize an output variable or to measure of the quality of a match of the calculated responses to a specified response on either a relative or absolute basis. Constraints are implemented by imposing a large penalty whenever the constraint is violated. Metrics for various functions can be combined to yield a combined fitness for different output variables, analysis types or circuit configurations. The total raw fitness F is then calculated using

$$F = \sum_{i=1}^{M} W_m f_m$$

where W_m is the weighting applied to each of the basic fitness metrics. This total fitness is used for getting the share of a particular chromosome in the total fitness. This helps in the selection of good individuals.

D. Genetic Operators

The genetic algorithm uses crossover and mutation operators to generate the offspring of the existing population. Before genetic operators are applied, parents have been selected for evolution to the next generation. We use the crossover and mutation operators and produce next generation. The probability of deploying crossover and mutation operators can be changed by user.

E. Halt Condition

Genetic Algorithm needs an Halt Condition to end the generation process. If we have no sufficient improvement in two or more consecutive generations; we can stop the Genetic Algorithm process. In other cases, we can use time limitation as a criterion for ending the process. We can also keep a desired fitness value within some percentage of accuracy as our Halt Condition.

F. Algorithm

Having looked at the above sections, we can now implement our algorithm which is as follows:

1. [Start] Generate a random population of n chromosomes (the format will be as stated in section A)
2. [Fitness] Evaluate the fitness f(i) of each chromosome i in the population with the fitness function (section C).
3. [New population] Create a new population by repeating the steps 4, 5 and 6 until the new population is complete.
4. [Selection] Select two parent chromosomes from a population according to their fitness. Roulette Wheel Selection or Thresholding or any method suitable for the above problem can be used.
5. [Crossover] With a crossover probability cross over the parents to form new offspring. This is analogous to reproduction and gives rise to a hybrid chromosome.

6. [Mutation] With a mutation probability mutate new offspring at each locus.

7. [Accept] Place new offspring in the new population for a further run of the algorithm.

8. [Replace] Use new generated population for a further run of the algorithm

9. [Test] If the halt condition (section E) is satisfied, we stop and return the best solution in current population, else go to step 2.

IV. Experiments and Results

We tested our improved algorithm for building an analog circuit. The analog circuit, which we built here for testing is a low pass filter with typical specifications provided. This circuit has been built earlier.

A threshold was fixed to generate a reasonable circuit and we ran both algorithms on the specific problem with different population sizes. Note that the crossover and mutation rates have been kept same for all the readings. So, inspite of increasing population size, there might be a discrepancy in the time required to generate the circuit. But, our goal is achieved because what we are looking for is a comparison of the duration required for the two algorithms. So, the operator effects are nullified.

The result is addressed in the following table:

TABLE I
Comparison Results of Previous and New Algorithm

Population Size	Time required	
	Modified GA	Previous GA
50	2 min 36 sec	3 min 42 sec
100	3 min 1 sec	4 min 36 sec
200	4 min 24 sec	9 min 12 sec

As the results show, the modified algorithm generates circuit in lesser time consistently. As population size increases, we see that the improvement is significant.

V. Conclusion

As mentioned earlier, Genetic Algorithm has been used for analog circuit design but the improved version shows significant improvements with higher population sizes. Therefore, it can be stated that the new algorithm is more efficient than the previous algorithm.

References

[1] John H. Holland, "Genetic Algorithms - Computer programs that evolvein ways that resemble natural selection can solve complex problems eventheir creators do not fully understand," Scientific American, pp. 66-72,July 1992.
[2] Holland, J.H. "Adaptation in natural and artificial systems", Ann Arbor: The University of Michigan Press, 1975.
[3] Goldberg, David E. "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.
[4] Koza, John R., "Genetic Programming - on the programming of computers by means of natural selection", MIT Press, 1992.
[5] Roberto Menozzi, Aurelio Piazzi and Fabrizio Contini, "Small-Signal Modeling for Microwave FET Linear Circuits Based on a Genetic Algorithm," IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, pp. 839-847, Vol. 43, No.10, October 1996.
[6] W. Druiskamp and D. Leenaerts, "Darwin: Analogue Circuit Synthesis Based on Genetic Algorithms," International Journal of Circuit Theory and Applications, Vol 23, 1995, pp.285-296.
[7] J. Stoffels and C. van Reeuwijk, "A design strategy for the synthesis of high-performance instrumentation amplifiers," Delft University of Technology Computational Physics Report Series, Report Number CP-96-002, 1996.
[8] J. B. Grimbleby, "Automated Synthesis of Active Electronic Networks using Genetic Algorithms," IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, Styrathclyde, September 1997, pp. 103-107
[9] John R. Koza, F. Bennett, D. Andre, M. Keane, and F. Dunlap, "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," IEEE Transactions on Evolutionary Computation, Vol 1, No. 2, July 1997, pp. 109-128
[10] M. Mitchell, An Introduction to Genetic Algorithm, MIT Press, 1996.