Analog Circuit Design Using Genetic Algorithms

Kenneth V. Noren[1]
John E. Ross[2]
knoren@uidaho.edu
johnross@johnross.com
[1]**Department of Electrical and Computer Engineering**
University of Idaho
Moscow Idaho 83844-1023
[2]**John Ross & Associates**
Salt Lake City, Utah 84116

Abstract

This paper proposes that genetic algorithms (GA's) are viable candidates as tools for design automation of analog circuits (DAAC). The principles, strengths, and weaknesses of GA's applied to sizing circuit parameters to meet performance specifications are discussed and a software implementation is introduced. Designs of BJT and CMOS operational amplifiers, a CMOS operational transconductor, and a matching network are presented as examples.

1 Introduction

The design of the analog portion of a mixed-signal integrated circuit often requires a large fraction of the overall design time, despite the fact that the analog circuit is often a relatively small portion of the overall circuit. Interestingly, much of the analog design time occurs after a candidate topology, having the potential to meet required specifications, has been selected as a building block for the system. This is due to the need to manually iterate or "tweak" circuit parameters (e.g. component values and transistor sizes) to meet specifications. This iteration process is particularly challenging when specifications must include temperature and process corners. Moreover, designs using short-channel MOSFET's further complicate the design process due to a lack of manageable equations relating circuit parameters to specifications that result. If design automation of analog circuits (DAAC) is to be achieved, a solution to automating this phase of the design process must be developed. The problem may be classified as a global optimization problem.

While there have been efforts to develop optimization tools for analog circuit design, they have generally been based on gradient search methods. Unfortunately, these methods require initial guesses, tend to get stuck in local minima and have difficulty with discrete variables and non-linear constraints. This paper asserts genetic algorithms (GA's) [1, 2, 3, 4] may be a better alternative for global optimization tools for DAAC.

GA's are based on the Darwinian principle of natural selection and the concepts of natural genetics. GA's have many desirable characteristics and offer significant advantages over traditional methods. They are inherently robust and have been shown to efficiently search large solution spaces containing discrete or discontinuous parameters and non-linear constraints, without being trapped in local minima. GA's do not require initial guesses or derivative information and have often found non-intuitive solutions to engineering problems.

While GA's have been applied extensively to many areas of engineering, there are relatively few references in the analog circuit literature [5, 6, 7, 8, 9, 10] that report the application of genetic algorithms. In most instances only linear circuits were considered (a relatively easy problem compared to non-linear circuits) or the optimizations were based on derived circuit equations. Koza [11] has performed non-linear circuit designs using genetic programming, but emphasized topology generation. While his results are very exciting, he has not yet demonstrated their effectiveness in generating circuits suitable for production in an industrial environment.

This paper illustrates the application of GA's to select component values and transistor sizes for a given circuit topology to provide practical design solutions for a given set of performance specifications. Since this phase of analog circuit design is time intensive, significant time savings can be achieved if a software tool can perform this task. The software discussed in this paper represents an important step in that direction.

2 The Genetic Algorithm

This section introduces some of the concepts of genetic algorithms and their implementation in GA-Spice. It is provided only as a guide and is not intended as a substitute for more detailed expositions on GA's available in the literature.

2.1 Chromosomes

Central to all genetic algorithms is the concept of the chromosome. The chromosome contains all information necessary to describe an individual. In nature, chromosomes are composed of DNA. In a computer, a long binary or character string is used. Chromosomes are composed of genes for the various characteristics to be optimized and can be any length depending on the number of parameters to be optimized.

2.2 Encoding

Encoding defines the way genes are stored in the chromosome and translated to actual problem parameters. A possible encoding scheme for a hypothetical circuit using a 16 bit binary chromosome is shown below:

11101	11	000110	101
Gene for bias resistor	Gene for output resistor	Gene for transistor width	Gene for capacitance

Figure 1. Encoding Example

2.3 Fitness

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 GA-SPICE program has several basic fitness metrics that can be applied to any Spice output variable. The basic metrics are described in terms of the output variables c_i, user specified response s_i, and user specified point-by-point weighting values w_i. All of the basic metrics are 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

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

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

$$F = \sum_{m=1}^{M} W_m \quad f_m, \tag{2}$$

where W_m is the weighting applied to each of the basic fitness metrics. The adjusted fitness, A, is defined by Koza [4] as

$$A = \frac{1}{1+F}, \tag{3}$$

and is used to rescale the fitness to the range 0 to 1 with better individuals having the larger values. Subsequently, the adjusted fitness of each individual is normalized by the total adjusted fitness of the M members of the population.

$$N_i = \frac{A_i}{\sum_{k=1}^{M} A_k} \tag{4}$$

The normalized fitness is used in determining the probability of a particular individual being selected for mating.

2.4 Crossover

Crossover is a method of exchanging genetic material between two parents to produce offspring. The operation is simple, a cross-over point is chosen at random and the genetic information to the left of the cross-over point in parent A is combined with the genetic information to the right of the cross-over point in parent B to produce new offspring. A second offspring can be created by using the information from the right of parent A and the left of parent B. The operation is illustrated with the example below:

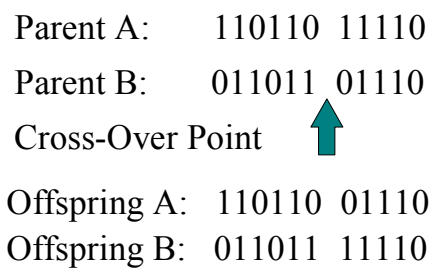

Figure 2. Single point crossover is a widely used to simulate the sharing of genetic information that occurs during mating.

A simple GA is described in this section. While more sophisticated forms of the GA exist, they are all extensions of this basic algorithm. The simple GA provides excellent performance for a wide range of problems.

The simple GA begins with the creation of an initial random population of individuals. The size of the population depends on the problem size. Population size ranges from as few as 20 to tens of thousands. Values of several hundred are commonly used.

The next step is to evaluate the fitness of each individual according to the predefined criteria. After the evaluation, the individuals are ranked according to fitness. Individuals are selected for mating based on fitness. Fitter individuals have a higher probability of mating and passing on genetic information to subsequent generations while less fit individuals have a non-zero probability of mating to preserve diversity.

Mating is simulated by applying the crossover operation to the chromosomes of two individuals selected based their fitness. Mutation is simulated by randomly changing a few bits in the chromosome of the offspring. Mutation provides a mechanism for exploring new regions of the solution space and prevents premature convergence to local minima. Finally, the fitness of the new generation is evaluated and the process is repeated for a specified number of generations or until a desired fitness is attained. A block diagram of the simple GA is shown below:

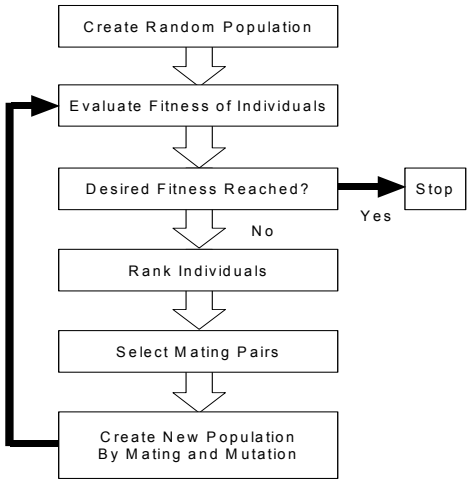

Figure 3. The simple GA is adequate for many engineering problems.

3 The GA-Spice Program

As part of this research, the GA-Spice program was developed. The program allows any parameter in a Spice netlist to be encoded as an optimization variable. This includes component values, node numbers, model types, process parameters, etc. Fitness functions can be constructed using any output variable available from Spice. Analysis types can include AC, DC, transient and noise. Optimization variables and fitness functions can be constructed easily via a point & click interface. The program is compatible with existing Spice tools for schematic capture and output visualization. The general nature of GA-Spice means that it is not limited to specific circuit topologies like equation based methods.

GA-Spice runs under the Microsoft Windows 9x/NT operating systems. Since the GA is inherently a coarse grain parallel operation it is possible to reduce optimization time by

employing parallel computer architectures. The current version of GA-Spice can make use of up to 4 processors in an symmetric multiprocessing system. A more powerful multi-processor version based on the Parallel Virtual Machine is under development.

4 Practical Applications of GA-Spice

The application of GA-Spice for the solution of several practical analog circuit applications is presented in this section.

4.1 BJT Operational Amplifier

The specifications for the BJT op-amp depicted in Figure 5 were a differential gain of at least 600V/V, an input offset voltage of 0V, and an output signal swing of at least at least ±13V. Several circuit designs produced by GA-Spice met the specifications. One of the circuits was built and tested in the laboratory to confirm that the design was practical and met specifications.

Interestingly, this circuit was also given as a design problem to junior electronics students. It required at least two weeks (three weeks or more for many) for the students to obtain a design that met specifications. Using GA-Spice, the design be performed in under 30 minutes with a 200 MHz Pentium notebook computer.

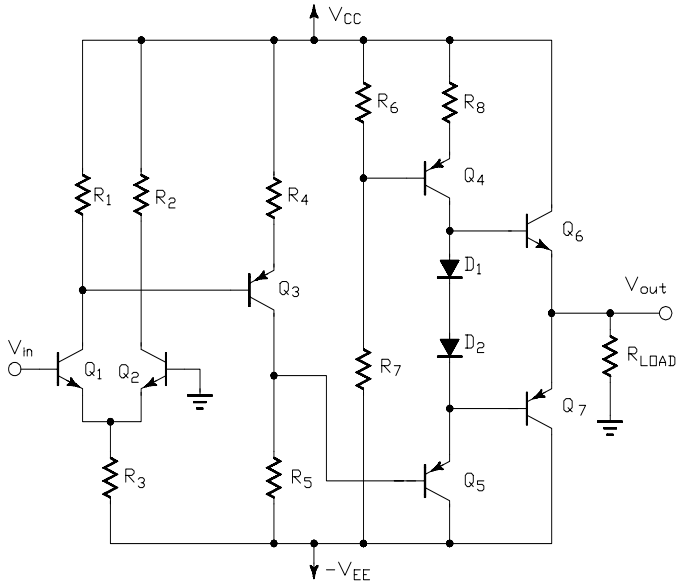

Figure 5 BJT Operational Amplifier

4.2 CMOS Operational Amplifier

The specifications for the CMOS Op-amp depicted in Figure 6 were to maximize gain and minimize the input offset voltage. The results from GA-Spice gave designs that had gains in excess of 2kV/V and input offset voltages of 0V. Transistor widths were fixed at 5µm, except for M_6 and M_7. These were fixed at 2µm. GA-Spice required one hour on a 200 MHz Pentium notebook computer to complete the design.

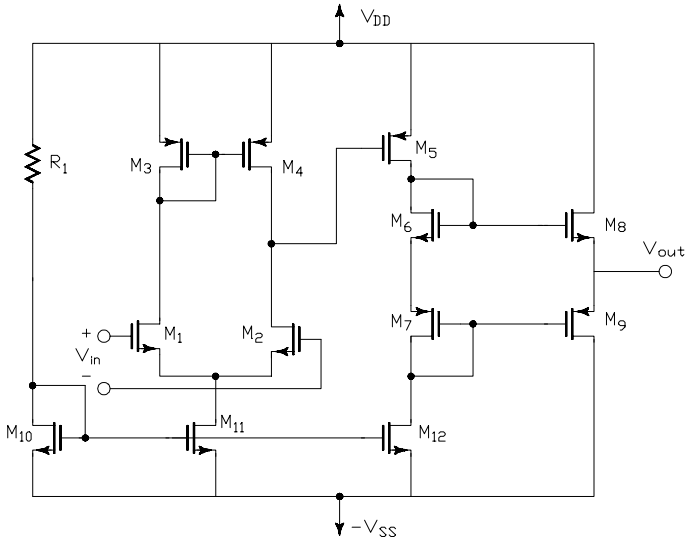

Figure 6 CMOS Operational Amplifier

4.3 Operational Transconductance Amplifier (OTA) Design

The topology of the OTA is shown in Figure 7. GA-Spice was used to determine the transistor widths and resistor value for R_1 required to properly bias the circuit and provide a specified transconductance. For this problem, the transistor widths were fixed at 2 um. The chromosome length was 76 bits, yielding 7.5×10^{22} possible designs. Desipite this enormous solution space, the program located an acceptable solution in approximately 15 minutes on 200 MHz Pentium notebook computer. The specifications and the resulting circuit performance are summarized in the table below.

Design Specifications	**Best GA-Spice Result**
DC Bias Points 1.1V < V(9) < 1.3V 3.7V < V(6) < 3.9 V 3.0V < V(10) < 3.2 V 3.0V < V(11) < 3.2V 3.7V < V(13) < 3.9V 0.8V < V(1) < 1.0V Gm: 95uA/V < Gm < 105uA/V	DC bias points: V(12) = 1.44V V(9) = 1.18V V(6) = 3.88V V(10) = 3.13V V(11) = 3.13V V(13) = 3.90V V(1) = 0.99V Gm: Gm = 100.7uA/V

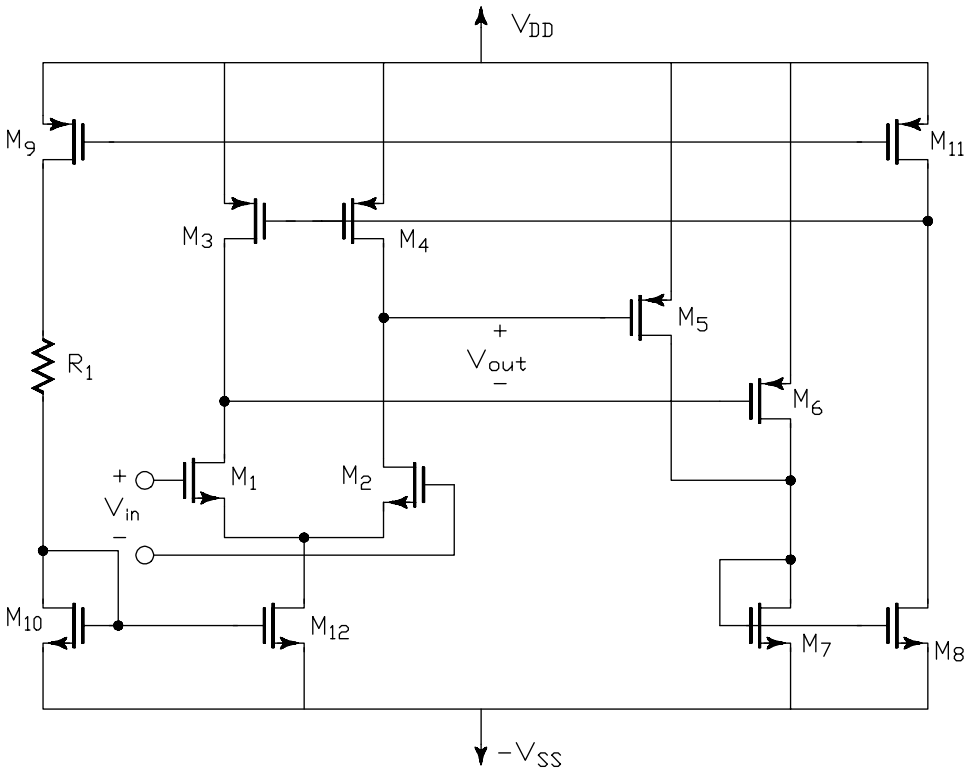

Figure 7. Schematic diagrom of OTA optimized by GA-Spice

W1=W2=6.96µm, W3=W4=101.5µm, W5=W6=4.6µm, W7=W8=27.4µm, W9=112.6µm, W10=3.23µm, W11=375.8µm, W12=7.56µm, and R_1 = 90.374kΩ. The spice model for the NMOS used VTO=0.7V, Kp=156µA/V, and lambda=30m V^{-1} and for the PMOS was VTO=-0.95V, Kp=48µA/V, and lambda=4.2µV^{-1}

4.4 Input Impedance Matching Network for an LNA

A schematic for a model for an input stage of a low noise amplifier is depicted in Figure 8. C_1 and C_2 are used to tune the input impedance, Z_{in}, to 50 Ω at 2.45GHz. L_{BOND} models bondwire inductance, C_{pad} models pad and trace capacitance, C_{par} models trace capacitance, R_3 and C_3 model the input impedance of a biasing circuit, R_{in} and C_{in} model the impedance looking into the gate of a MOSFET, R_{S1}, R_{S2}, C_{S1}, C_{S2}, and R_{lin} model parasitics of an integrated square spiral inductor.

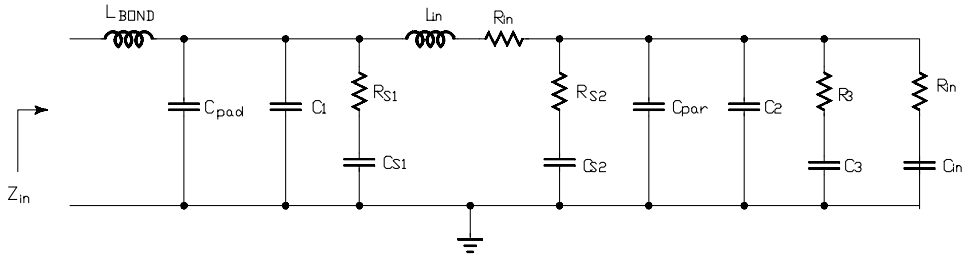

Figure 8. Schematic for a model for the input stage of a low-noise amplifier.

The results from GA-Spice provided $Z_{in} = 50.75 - j\ 3.5\ \Omega$ at 2.45GHz. GA-Spice provided a solution after running about 15 minutes on a notebook computer with a 200MHz Pentium processor.

The values for the components are $L_{BOND} = 3nH$, $C_{pad} = 500fF$, $C_{par} = 12.56fF$, $R_3 = 1.796k\Omega$, $C_3 = 72.22fF$, $R_{in} = 107.5\Omega$, $C_{in} = 265.7fF$, $R_{S1} = 52.7\Omega$, $R_{S2} = 79.6\Omega$, $C_{S1}=113fF$, $C_{S2}=93.8fF$, and $R_{lin} = 21.9\Omega$. C_1 and C_2 were selected by GA-Spice to be 306fF and 20fF.

5 Discussion

Four circuits whose component values have been selected using a genetic algorithm are presented in Section 4. Each of the these circuits met a set of simple specifications in a very reasonable amount of computer time.

The input matching circuit was an actual design problem in an industrial setting. Realistic values for the components were used. Linear circuits appear to be very easy for GA-Spice to optimize.

For the BJT and CMOS operational amplifiers and the CMOS operational transconductor amplifier, initial designs provided by GA-Spice proved unrealistic. For example, we found that GA-Spice had biased several of the Gate-Source voltages (VGS's) of several MOSFET's at levels of mere millivolts above the threshold voltage. This is very impractical. Thus, the fitness functions for the transistor circuits must be defined to prevent this. For example, we specified the overdrive voltages of all MOSFET's had to be between 0.2V and 0.5V for threshold voltages of 0.7V. The DC biasing for the BJT op-amp required similar constraints.

The user also must recognize when transistors in a design must be matched and properly link the parameters of the second transistor to the first using a symbolic link option provided by the program. For example, in the CMOS OTA, M3 and M4, M5 and M6, and M7 and M8, are MOSFET pairs used this symbolic link encoding scheme to provide proper OTA operation. An attempt to use a genetic algorithm without proper linkage produced a design that met specifications, however it resulted in a poor overall design due to the circuit symetry being destroyed.

For the CMOS op-amp and OTA, the MOSFET model used was a basic square-law model and not the current industry standard BSIM3v3 model. This helped speed up simulation times and still demonstrates the ability of the program to design analog circuits where equations are not available.

One *measurement* used in this paper is the simulation time of the GA-Spice. It has been stated that for simple circuits, this time is relatively small even on a slow notebook computer and could be comparable to the design time of an engineer. However, as GA-Spice is applied to larger circuits, perhaps the measurement will be the amount of time saved by the engineer. Even if a GA cannot design a circuit as quickly as an engineer, if it saves an engineer hundreds of man-hours, it is still an effective tool.

In an industrial environment, specifications and the MOSFET models will not be so simple. An effort to show the successful application of GA's to industrial strength analog circuits is presently underway. Future research also includes determining the types of genetic algorithms most suitable for analog circuit design, determining the best types of encoding schemes, and developing optimal fitness functions for the different circuit types.

6 Conclusions

This paper demonstrates that the applications of GA's are viable candidates for use in DAAC by successfully designing several circuits. The methods used are general and can be applied

to all analog circuits. It has been observed that GA-Spice provides a number of different solutions to a problem in a much shorter time than trial and error. GA-Spice can help provide new insights into circuit operation and produce novel designs. It appears GA's can play a role for future tools for DAAC. At a minimum, they can speed up or eliminate a considerable amount of the trial and error engineering being done by many organizations today.

References

[1] John H. Holland, "Genetic Algorithms - Computer programs that evolve in ways that resemble natural selection can solve complex problems even their 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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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

[11] 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