Analog Circuit Design Based on Computational Intelligence Techniques

Gabriel Oltean, Sorin Hintea, and Emilia Şipos

Abstract:

This paper presents a new method for analog circuit design optimization. Our approach turns to good account the advantages offered by computational intelligence techniques. Design objectives can be expressed in a flexible manner using fuzzy sets. This way appears the possibility to consider different degrees for requirement achievements and acceptability degree for a particular solution. Neuro-fuzzy systems (universal approximators) are used to model the complex multi-variable and nonlinear circuit performances. These models satisfy two main requirements: high accuracy and low computation complexity. An efficient and robust genetic algorithm avoids local minima in its exploration of the large, multidimensional solution space in quest for the optimal solution.

Keywords: *analog circuit design, optimization, genetic algorithm, fuzzy sets, neuro-fuzzy systems.*

1. Introduction

The purpose of analog circuit design is to produce a sized circuit schematic starting from a set of circuit requirements. Given a circuit schematic and the circuit's performance specifications, the sizes and biasing of all devices have to be determined such that the circuit meets the specifications at some optimal cost. This is a difficult and critical step for several reasons: 1) most analog circuits require a custom optimized design; 2) the design problem is typically underconstrained with many degrees of freedom; and 3) it is common that many (often conflicting) performance requirements must to be taken into account, and tradeoffs must be made that satisfy the designer [1].

Optimizations tools appear, naturally, as the key factor for the tremendous effort of finding the design parameters, which satisfy a complex, high-dimensional, multi-objective and multi-constrained problem [2]. An optimization algorithm for analog circuit design has three key components: formulation of the optimization problem, performance evaluation engine, and optimization engine.

Research efforts on circuit synthesis involving a broad spectrum of computational intelligence (CI) techniques have begun to appear in the literature over the past few years. Fuzzy sets are used to formulate the objective functions, getting this way the possibility to consider different degrees for requirement achievements and acceptability degrees for a particular solution. One approach [3] - [6], is to consider that the membership degree μ represents the degree of fulfillment of the fuzzy objective. A value $\mu=1$ means that the objective is fully satisfied, while a value $\mu=0$ means that the objective is not satisfied at all. This method has a disadvantage in the case of equality requirement, no information being available regarding the relation between the requirement and the actual performance. An accurate estimation of the circuit performances requires the use of complex models leading to an excessively large computation time in the iterative optimization process. One way to reduce the computation time is to use more simple models of circuit performances. In order to satisfy both main requirements (accuracy and speed), many researches proposed several CI-based methods to evaluate circuit performances. For example least-square support vector machines are involved in [7] - [9]. Fuzzy systems are very useful to model the circuit performances because they imply just a few simple mathematical operations and can model any complex, multivariable and nonlinear function at any level of accuracy. Such fuzzy models are used in [10] - [13].

The optimization engine (the way to compute new parameter values) is the heart of the optimization algorithm. It should be chosen so that the optimization will converge to an optimal solution in a reduced number of iterations. This task is not an easy one due to complex relations between design parameters and circuit performances. A parameter affects more than one circuit performance at a time, so when a parameter is modified to improve a performance it can worsen another. To search the whole solution space, a powerful global optimization technique should be considered. Genetic algorithms (GA) are based on the Darwinian principle of natural selection and the concepts of natural genetics. GAs 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. GAs do not require initial guesses or derivative information and have often found non-intuitive solutions to engineering problems. Genetic algorithms have already been employed in many CAD applications [10], [14] - [17].

The objective of this work is to develop an efficient algorithm based on computational intelligence techniques for design optimization of analog circuits. Our approach tries to combine the best qualities of these techniques: flexibility in formulation the objective functions and a known range of their values using fuzzy sets; accuracy and low computational complexity of circuit performance models based on neuro-fuzzy systems; and a powerful global optimization engine based on

genetic algorithm.

The reminder of the paper is organized as follows. We begin in Section 2 with an overview of the CI-based optimization algorithm used in the design optimization of analog modules. The techniques used in the key phases of the algorithm are presented here. Section 3 focuses on the utilization of our proposed algorithm and results obtained for design optimization of a CMOS amplifier. In the end, in Section 4, some conclusions and further research directions are discussed.

2. CI-based Optimization Algorithm

Design optimization of an electronic circuit is a technique used to find the design parameter values (length and width of MOS transistors, bias current, capacitor values etc.) in such a way that the final circuit perfor-mances (dc gain, gain-bandwidth, slew rate, phase margin etc.) meet as close as possible the design requirements. As stated in [18] there are two basic modalities to deal with the analog design: knowledge based approaches and optimization based approaches. In the present paper we are centered on the last one.

2.1. Overview of the Optimization Algorithm

The optimization algorithm begins with the formulation of optimization objectives and optimization problem, followed by the initialization of the design parameters. During iterations an evaluation engine computes the actual circuit performances based on the actual design parameter values. If the objectives are fulfilled, the solution consists in the set (or sets - in the case of a real multiobjective optimization) of the actual design parameter values and the algorithm is stopped. If not, new design parameter values are to be computed by the optimization engine and the optimization loop is covered once again.

The novelty introduced in this paper is the utilization of different CI techniques in all phases of optimization algorithm, as it is shown in Fig. 1.

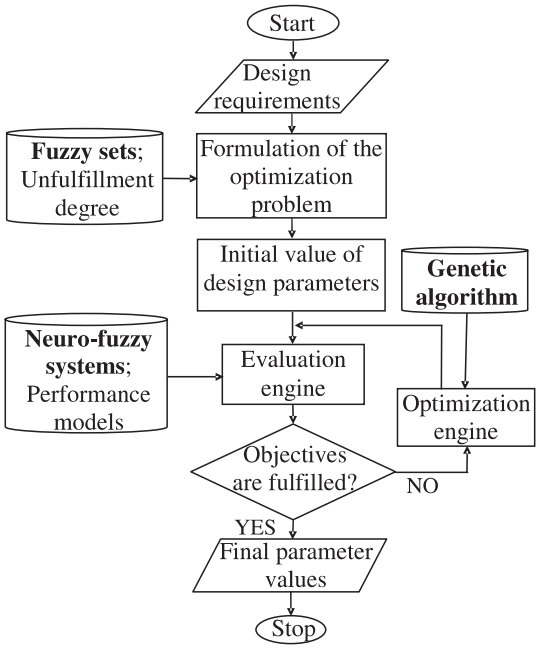

Fig. 1. CI-based optimization.

Fuzzy sets are used to define the objective function in formulation of the optimization problem. Neurofuzzy systems address the performance evaluation issue (evaluation engine). Finally, in the optimization engine, a genetic algorithm is responsible for the evolution of the population to finally produce the (near) optimum solution.

2.2. Formulation of the Optimization Problem

To solve a multiobjective optimization problem, as is the design optimization of analog circuits, the optimization problem can be formulated in one of the following two ways [14]:

1) As a single-objective, constrained optimization problem, where different performance objectives are combined to form a single scalar objective, and which produces one solution.
2) As a multiobjective optimization problem, where the concept of Pareto-optimality is used to produce multiple tradeoff solutions on a design decision surface.

Usually, for an analog circuit optimization problem with three or more objectives, the first approach appears to be computationally cheaper than the second approach. As a consequence this paper takes the single-objective approach.

To formulate the design objectives for a real design is not always a simple task. Often, it is not clear what precise values should be given to each objective. In fact, design objectives are often better expressed in real world terms than in precise numbers. The designers usually can accept a certain degree of fulfillment of the design objectives.

The fuzzy techniques used to define the optimization objectives suppose the fuzzification of the requirements, getting this way the possibility to consider different degrees for requirement achievements and acceptability degrees for a particular solution.

The authors proposed the utilization of fuzzy sets to define the objective functions in previous papers [19] and [20]. By contrast with the existing approaches where membership degree represents the degree of fulfillment, in our approach the membership degree μ represents the error degree in the fulfillment of the objective. A value $\mu=1$ means the objective is not satisfied at all, while a value $\mu=0$ means that the objective is fully satisfied. With this approach we know the range of possible values for objective functions as being [0, 1]. When the value of a certain objective function (unfulfillment degree - UD) is 0, we know that the corresponding requirement is fulfilled, no further effort being necessary to improve the associated performance.

The membership degree μ represents the error degree in the fulfillment of the fuzzy objective. A value $\mu=1$ means the objective is not satisfied at all, while a value $\mu=0$ means the objective is fully satisfied.

As an example, the requirements "greater or equal" $f_k(x) \geq f_k^r$ and "equal" $f_k(x) = f_k^r$ have the corresponding fuzzy objective functions presented in Fig 2. where:

x - the vector of the design parameters;

f_k - the k^{th} performance function;

f_k^r - the k^{th} requirements;
x^* - the current value of the design parameters vector.

The fuzzy objective functions are.

$$\mu_k(f_k(x)):\ D_{f_k} \rightarrow [0,1]\ . \quad (1)$$

where D_{f_k} is the range of possible values for $f_k(x)$. $\mu_k(f_k(x))$ indicates the error degree in accomplishing the k^{th} requirement, so we will call it "unfulfillment degree" (UD). A value $\mu_k = 0$ means full achievement of fuzzy objective, while a value $\mu_k = 1$ means that the fuzzy objective is not achieved at all. This occurs when $f_k(x)$ takes an unacceptable value. Fig 2 shows the corresponding value of the unfulfillment degree μ_k^* for the current value of the variables vector x^*. With this approach we know the range of possible values for objective functions as being [0, 1]. When the value of a certain objective function (UD) is 0, we know that the corresponding requirement is fulfilled, no further effort being necessary to improve the associated performance, as usually happens in typical minimization-type optimization.

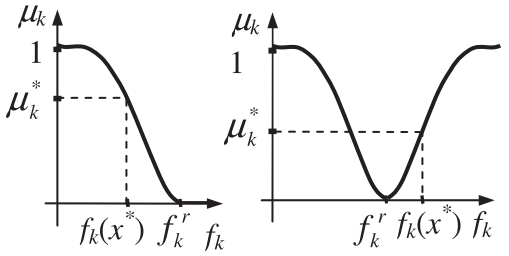

Fig. 2. Fuzzy objective functions: a) $f_k(x) \geq f_k^r$; b) $f_k(x) = f_k^r$.

The formulation of the multiobjective optimization problem became:

Find x that minimizes
$\{\mu_1(f_1(x)),\ \mu_2(f_2(x)),\ldots,\ \mu_n(f_n(x))\}$. (2)

where n is the number of requirements.

For our single-objective optimization approach, we have to combine the individual objective functions into a cost function by means of a weighted sum:

Find x that minimizes $F(x) = \sum_{k=1}^{n} w_k \mu_k(f_k(x))$. (3)

where w_k is the relative preference or weight associated with the k^{th} objective function. The requirement of the optimization, to satisfy all the objectives at the end of the optimization run, suggests that all the different objectives be weighted equally. On the other hand, for a given problem, some of the objectives may be more difficult to attain than some others. Thus, if a classification among the objectives is possible on grounds of relative difficulty of attainment, one would like to give higher numerical weights to the difficult objectives than the others.

2.3. Evaluation Engine

The design process of an electronic circuit is an iterative process (see Fig. 1) that requires a large number of performance evaluations. Analog circuits are difficult and time-consuming for a proper evaluation. Even in the case of basic characteristics of a simple circuit (amplifier gain, gain-bandwidth, slew rate etc) the performance in question can be a complex function of many parameters. In a realistic case, a performance model will be in general a non-linear function over a high dimensional space of circuit parameters [9].

Even a small cell requires a mix of ac, dc and transient analyses. An accurate estimation of the circuit performances requires the use of complex models leading to an excessively large computation time. One way to reduce the computation time is to use more simple models of circuits and devices.

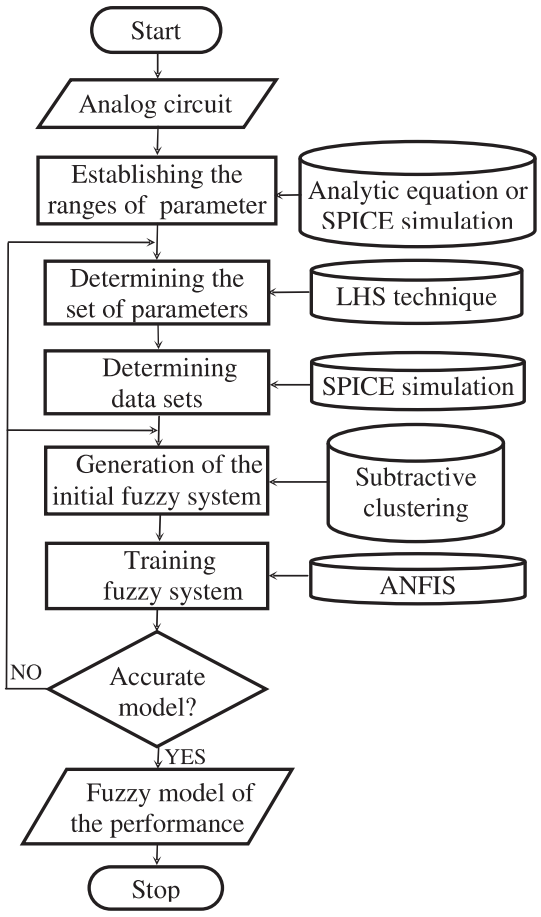

Fig. 3. Modeling procedure for circuit performance.

Fuzzy systems are very useful to model the circuit performances because they imply just a few simple mathematical operations and can model any complex, multivariable and nonlinear function at any level of accuracy. The author synthesized a method to build neuro-fuzzy models and used it for some analog modules [19], [21], [22]. These models are automatically built up using an input-output data set, using the ANFIS (Adaptive Neuro-Fuzzy Inference System) framework [23] to develop first order Takagi-Sugeno fuzzy systems. A common way to apply a learning algorithm to a fuzzy system is to represent it in a special artificial neuronal network (ANN) like architecture. In the ANFIS framework, six-layer architecture for ANN is used. ANFIS makes use of a mixture of back propagation to learn the premise para-

meters and least mean square estimation to determine the consequent parameters [23], [24].

The full modeling procedure is explained in Fig. 3. The ranges of the parameter values are established so that irrespective of the parameter values combinations, the circuit will operate in the desired region. For example, in an amplifier the transistors should be maintained in their active regions. The parameter set (the combination of the parameter values) should be chosen to be representative for the function to be modeled (covers the space of the parameters and embeds all the specific characteristics of the function).

For each input vector (one combination of the parameter values), the function value has to be found, in our case by SPICE simulation. Two data sets, a training set and a checking set are generated.

The training set is then subdued to a subtractive clustering procedure resulting an initial first order Takagi-Sugeno fuzzy system. Next, the initial fuzzy system is trained using ANFIS and the training and checking data sets.

The resulting neuro-fuzzy model is tested from the accuracy point of view. If the accuracy is unacceptable, the procedure must be resumed by generating a new initial fuzzy system or even by determining new data sets. If the accuracy is acceptable, the modeling procedure stops and provides the desired fuzzy model of that circuit performance.

The main advantage of fuzzy models is that there are no restrictions in the kind of functions that can be modeled, as far as neuro-fuzzy systems are universal approximators.

2.4. Optimization Engine

The heart of the whole algorithm is the optimization engine. A genetic algorithm (GA) is responsible for the exploration of solution space in quest of the optimal solution. Generally, the best individuals of any population tend to reproduce and survive, thus improving successive generations [25]. However inferior individuals can, by chance, survive and reproduce. In our case, the individuals consist of different versions (same topology, but different parameter values), which can evolve until a solution is reached (in terms of requirements accomplishment).

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. Basically, in a genetic algorithm each chromosome is potentially a solution of the optimization problem. Encoding defines the way genes are stored in the chromosome and translated to actual problem parameters.

A generic chromosome employed in our algorithm is shown in Fig. 4, where each gene represents a design parameter.

The "alphabet" used in the representation of genes can, in theory, be any finite alphabet. Thus, rather than use the binary alphabet of 1 and 0, one can use an alphabet containing more characters and numbers [26].

Fig. 4. Format of a generic chromosome.

Because the design parameters are real variables we chose real numbers to compose our alphabet. This way our population individuals are real valued vectors, rather than bit strings, thus simplifying the development of GA implementation.

The underlying procedure of our GA is shown in Fig. 5.

1. Population initialization
2. Fitness assignment
3. Selection
4. Recombination
5. Mutation
6. Reinsertion
7. Loop to step 2 until optimal solution is found

Fig. 5. GA procedure.

Seeding the population with random values, with a uniform probability, commonly does population initialization. It is sometimes feasible to seed the population with "promising" values that are known to be in the hyperspace region relatively close to the optimum [26]. Our implementation uses random at uniform initialization.

Each individual in the selection pool receives a reproduction probability depending on the own objective value and the objective value of all other individuals in the selection pool. This fitness is used for the actual selection step afterwards. Our implementation uses rank-based fitness assignment with linear ranking [27].

Consider $Nind$ the number of individuals in the population, Pos the position of an individual in this population (least fit individual has $Pos=1$, the fittest individual $Pos=Nind$) and SP the selective pressure. For example, in the case of a minimization-type optimization problem first position is allocated to the individual with highest value of the objective function. The fitness value for an individual is calculated as:

$$Fitness(Pos) = 2 - SP + 2(SP-1)\frac{Pos-1}{Nind-1} . \quad (4)$$

Linear ranking allows values of selective pressure in [1.0, 2.0].

For the selection our approach uses the roulette-wheel method. This is a stochastic algorithm and involves the following technique. For each individual a selection probability is computed as:

$$Selection\_probability(i) = \frac{Fitness(i)}{\sum_{i=1}^{N} Fitness(i)} . \quad (5)$$

The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its selection probability. A uniformly distributed random number is generated and the individual whose segment spans the random number is selected. The process is repeated until the desired number of individuals is obtained (called mating population). This technique is analogous to a roulette wheel with each slice proportional in size to the fitness.

Recombination produces new individuals in combining the information contained in two or more parents (parents - mating population). This is done by combining the variable values of the parents. Depending on the representation of the variables different methods must be used. For our real valued variables the intermediate recombination method was chosen. Offspring are produced according to the rule [27]:

$$Var_j^O = a_j Var_j^{P1} + (1 - a_j) Var_j^{P2}, \quad j = 1, 2, ..., Nvar \; . \tag{6}$$

where Var_j^O represent the j^{th} variable of the offspring, Var_j^{P1} represent the j^{th} variable of the first parent, while Var_j^{P1} represent the j^{th} variable of the second parent. The scaling factor a_j is chosen uniformly at random over an interval $[-d, 1+d]$ for each variable anew.

Intermediate recombination is capable of producing any point within a hypercube slightly larger than that defined by the parents.

By mutation, individuals are randomly altered. Mutation of real variables means that randomly created values are added to the variables with a low probability. Thus, the probability of mutating a variable (mutation rate) and the size of the changes for each mutated variable (mutation step) must be defined.

The probability of mutating a variable is inversely proportional to the number of variables (dimensions). The more dimensions one individual has, the smaller is the mutation probability. Different papers reported results for the optimal mutation rate. In [28] it is shown that a mutation rate of $1/n$ (n: number of variables of an individual) produced good results for a wide variety of test functions. That means that per mutation only one variable per individual is changed/mutated. Thus, the mutation rate is independent of the size of the population.

The size of the mutation step is usually difficult to choose. The optimal step-size depends on the problem considered and may even vary during the optimization process. It is known that small steps (small mutation steps) are often successful, especially when the individual is already well adapted. However, larger changes (large mutation steps) can, when successful, produce good results much quicker. Thus, a good mutation operator should often produce small step-sizes with a high probability and large step-sizes with a low probability.

Such an operator [27] was considered for our algorithm:

$$Var_j^{Mut} = Var_j + s_j r_j a_j, \quad j = 1, 2, ..., Nvar \tag{7}$$

$s_j \in \{-1, +1\}$ uniform at random

$r_j = r \cdot domain_i$, r - the mutation range (standard 10%)

$a_j = 2^{-uk}$, $u \in [0, 1]$ uniform at random, k - the mutation precision

The range of mutation is given by the value of the parameter r and the range of the variables. The parameter k (mutation precision) defines indirectly the minimal possible step-size and the distribution of mutation steps inside the mutation range. The smallest relative mutation step-size is 2^{-k}, the largest $2^0 = 1$. By changing these parameters (r and k) very different search strategies can be defined.

Our GA uses the pure reinsertion scheme: produce as many offspring as parents and replace all parents by the offspring. Every individual lives one generation only.

3. Design Optimization of a CMOS Amplifier

The proposed CI-based design optimization algorithm is implemented in the Matlab environment. It accepts three types of requirements: "greater than", "equal", and "smaller than". The user should provide numerical values, types and weights for all the requirements.

We used our algorithm for the design optimization of a CMOS simple operational transconductance amplifier (SOTA), shown in Fig. 6.

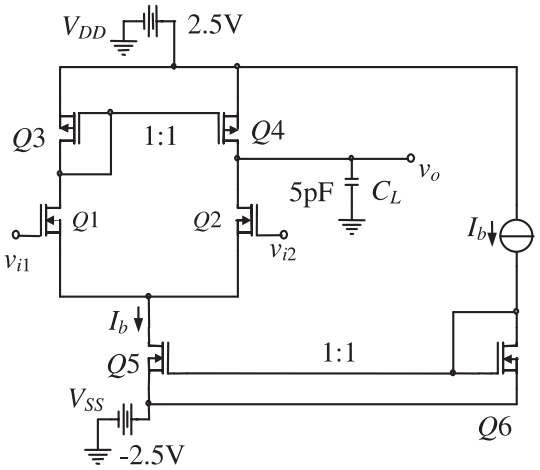

Fig. 6. Simple operational transconductance amplifier.

The design parameters of the circuit are the dimensions of the transistors (W/L) and the bias current I_b. The input transistors Q_1 and Q_2 must be identical, therefore $(W/L)_1 = (W/L)_2$ resulting the first parameter $(W/L)12 = (W/L)_{12}$. The transistors Q_3 and Q_4 (active load) must be paired, resulting $(W/L)_3 = (W/L)_4$, so our second parameter will be $(W/L)34 = (W/L)_{34}$. For the current mirror, Q_5 and Q_6, we consider the current (I_b) equal trough both transistors so $(W/L)_5 = (W/L)_6$. In order to keep a minimal area, we have taken $W = L$ so we obtained our third parameter $(W/L)56 = (W/L)_{56}$. The fourth and final parameter is I_b.

As performances, the important ones were considered: voltage gain A_{v0}, unity gain bandwidth GBW, and common mode rejection ratio $CMRR$.

Applying the previously described procedure, we built the neuro-fuzzy models of circuit performances with a set

of 850 data pairs (700 training pairs and 150 checking pairs).

The design optimization is illustrated here for the set of requirements presented in Table 1, for equal weighted objective functions. The optimization was run several times, for a population of 100 individuals. The algorithm proved to be a robust one, always a solution being found that fulfills all the requirements. Different number of iterations is necessary to search for the optimum solution depending on the initial population and on the evolution process. Table 1 gives the final performances of the circuit after four different optimization runs, while Table 2 shows the solutions (the values of the design parameters). The solutions appear to be slightly different from each other. At a closer look we can see that the values of the design parameters are calculated with two decimals. In practical implementations these values should be rounded toward some discrete values, so we can consider that our resulted solutions are in fact small variations around one solution - our solutions are near optimum solutions.

Table 1. Requirements and performances optimizing SOTA in four optimization runs.

Circuit functions	Requirements	Performances			
		run1	*run2*	*run3*	*run4*
A_{v0}	≥ 50	52.17	52.01	51.83	51.97
GBW [kHz]	$\geq 4\,600$	4 601	4 620	4612	4608
CMRR	$\geq 1\,000\,000$	1 011 413	1 008 935	1 004 844	1 003 761
Number of iterations		94	90	87	63

Table 2. Solutions optimizing SOTA in four optimization runs.

Design parameter	run1	run2	run3	run4
$W12$	39.28	39.73	39.33	39.07
$W34$	4.00	3.99	3.98	3.99
$W56$	6.90	6.86	7.21	7.23
I_b [μA]	96.33	97.22	94.74	94.00

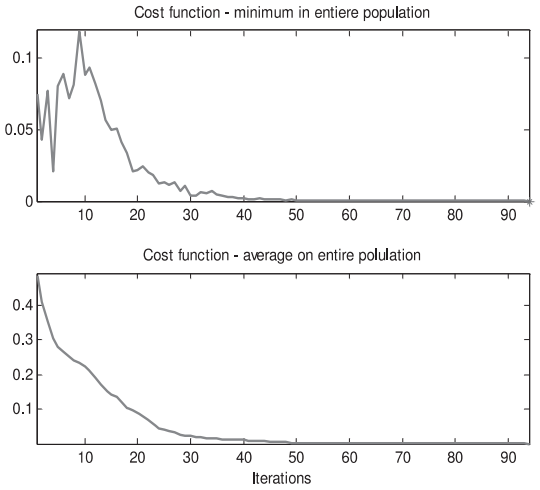

Fig. 7. Minimum and average cost function evolution for run1 .

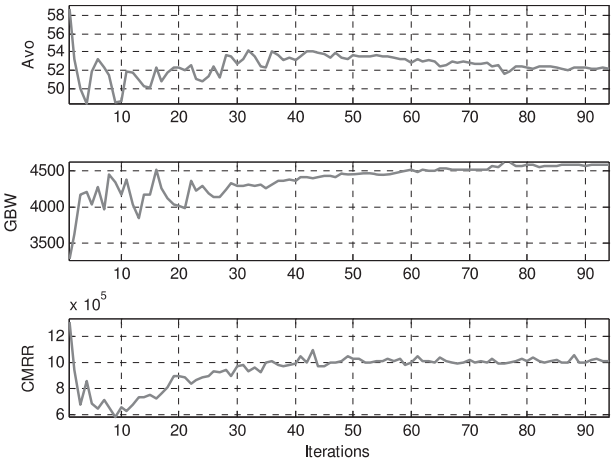

Fig. 8. Dynamic behavior of performances in run1.

For the first optimization run *(run1)*, the dynamic behavior of the algorithm is presented in Fig. 7. In the first iterations (up to 10), due to a high diversity of individuals, a new population does not contain always a fittest individual than in the previous population (minimum value of the cost function increases). On the other hand, the population as a whole is improved continuously, the average value on the entire population of the cost function decreasing in time. As the population improves during evolution, all individuals moves toward the optimal solution, decreasing both the minimum and average values of cost function.

The evolution of all performance functions during optimization is presented in Fig. 8.

4. Conclusions

A new computational intelligence-based optimization algorithm for analog circuit design was presented. The proposed algorithm was used to optimize the design of a CMOS simple transconductance amplifier, with very promising results. In a reduced number of iterations (63 to 94) it was able to always find an optimal solution, regardless the initial starting point (initial population), proving its robustness. The multiobjective optimization problem, specific to the analog circuit design, was reformulated as a single-objective optimization. For each optimization run the proposed algorithm produces one optimum solution. A further research direction is to use a real multiobjective optimization method to produce solutions on the Pareto front.

AUTHORS

Gabriel Oltean, Sorin Hintea*, and Emilia Şipos - Technical University of Cluj-Napoca, C. Daicoviciu Street, 15, Cluj-Napoca, Romania. E-mails: {Gabriel.Oltean, Sorin.Hintea, Emilia.Sipos}@bel.utcluj.ro.

* Corresponding author

References

[1] R.A. Rutenbar, G.G.E., Gielen, J. Roychowdhury, "Hierarchical Modeling, Optimization, and Synthesis for System-level Analog and RF Designs", In: *Proc. of the IEEE*, vol. 95, issue 3, 2007, pages 640-669.

[2] M. Barros, J. Guilherme, N. Horta, "GA-SVM Optimization Kernel Applied to Analog IC Design Automation". In: *Proc. of 13th IEEE International Conference on Electronics, Circuits and Systems*, ICECS06, 2006, pp. 486-489.

[3] B.Sahu, A.K. Dutta, "Automatic Synthesis of CMOS Operational Amplifiers: A Fuzzy Optimization Approach" . In: *Proc. of 15th Intl. Conf. on VLSI Design*, 2002, pp. 366-371.

[4] M. Fares, B. Kaminska, "FPAD: A Fuzzy Nonlinear Programming Approach to Analog Circuit Design", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no.7, 1995, pp. 785-793.

[5] B.R.S. Rodriguez, M.A. Styblinski, "Adaptive Hierarchical Multi-Objective Fuzzy Optimization for Circuit Design". In: *IEEE International Symposium on Circuits and Systems*, ISCAS '93, 1993, pp. 1813-1816.

[6] L. Fu, J. Fan, L. Yuanchun, T. Yantao, D. Yisong, "Study and Application of a Constrained Multi-objective Optimization Algorithm". In: *IEEE International Vehicle Electronics Conference*, IVEC 1999, pp. 305-307.

[7] X. Ren, T. Kazmierski, "Performance Modeling and Optimization of RF Circuits Using Support Vector Machines". In: *Proc. of 14th International Conference on Mixed Design of Integrated Circuits and Systems*, MIXDES07, 2007, pp. 317-321.

[8] F. De Bernardis, M.I. Jordan, A. Sangiovanni Vincentelly, "Support Vector Machines for Analog Circuit Performance Representation", *DAC 2003*, Anaheim, California, USA, 1-58113-688-9/03/0006, 2003, pp. 964-969.

[9] T. Kiely, G. Gielen, "Performance Modeling of Analog Integrated Circuits Using Least-Squares Support Vector Machines". In: *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, vol. 1, 2004, pp. 448-453.

[10] S. Balkir, G. Dundar, G. Alpaydin, "Evolution Based Synthesis of Analog Integrated Circuits and Systems". In: *Proc. of the IEEE NASA/DoD Conference on Evolution Hardware*, EH'04, 2004 , 0-7695-2145-2/04, pp. 26-29.

[11] G. Alpaydin, S. Balkir, G. Dundar, "An Evolutionary Approach to Automatic Synthesis of High-Performance Analog Integrated Circuits", *IEEE Transaction on Evolutionary Computation*, vol. 7, no.3, June 2003, pp. 240-252.

[12] A. Torralba, J. Chavez, L.G. Franquelo, "A Collection of Fuzzy Logic-Based Tools for the Automated Design, Modeling and Test of Analog Circuits", *IEEE Micro*, vol. 16, no. 4, 1996, pp. 61-68.

[13] N.D. Venkata, B.L. Evans, "An Automated Framework for Multicriteria Optimization of Analog Filter Design", *IEEE Trans. on Circuits and System-II: Analog and Digital Signal Processing*, vol. 46, no. 8, 1999, pp. 981-990.

[14] A. Somani, P.P. Chakrabarti, A. Patra, "An Evolutionary Algorithm-Based Approach to Automated Design of Analog and RF Circuits Using Adaptive Normalized Cost Functions", *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, 2007, pp. 336-353.

[15] T. Eeckelaert, R. Schoofs, G. Gielen, M. Steyaert, W. Sansen, "Hierarchical Bottom-up Analog Optimization Methodology Validated by a Delta-Sigma A/D Converter Design for the 802.11a/b/g standard". In: *Design Automation Conf.*, 43rd ACM/IEEE, 2006, pp. 25-30.

[16] K. Takemura, T. Koide, H.J Mattausch, T. Tsuji, "Analog-circuit-component optimization with genetic algorithm". In: *The 47th Midwest Symposium on Circuits and Systems*, vol. 1, 25th -28th July, 2004, pp. I - 489-92.

[17] M. Taherzadeh-Sani, R. Lotfi, H. Zare-Hoseini, O. Shoaei, "Design optimization of analog integrated circuits using simulation-based genetic algorithm". In: *International Symposium on Signals, Circuits and Systems*, SCS2003, 10th-11th July, vol. 1, 2003, pages 73-76.

[18] R.L. Gieger, P.E. Allen, N.R. Strader, *VLSI Design Techniques for Analog and Digital Circuits*, McGraw-Hill Publishing Company, 1990.

[19] G. Oltean, "FADO - A CAD Tool for Analog Modules". In: *Proc. of the International Conference on "Computer as a Tool"*, EUROCON2005, Belgrade, ISBN 1-4244-0050-3, IEEE catalog number: 05EX1255C, 2005, pp. 515-518.

[20] G. Oltean, "Multiobjective Fuzzy Optimization Method", Scientific Bulletin of the Politechnica University of Timisoara, *Trans. on Electronics and Communications*, vol. 49, issue 63, no. 1, ISSN 1583-3380, 2004, pp. 220-225.

[21] G. Oltean, S. Hintea, Doris, Lupea, "A Fuzzy Optimization Engine for Analog Circuit Design". In: *Proc. of the 8th International Conference MIXDES2001*, Zakopane, Poland, 21st-23rd June 2001, pp. 77-82.

[22] G. Oltean, C. Miron, M. Crasi, M. Carlugea, "Evaluation of the Analog Circuits Performances Using Fuzzy Models", Scientific Bulletin of the Politechnica University of Timisoara, *Trans. on Electronics and Communications*, vol. 47, issue 61, no. 1, ISSN 1583-3380, 2002, pp. 12-17.

[23] R.J.-S Jang, "ANFIS, Adaptive-Network-Based Fuzzy Inference System", *IEEE Transaction on System, Man, and Cybernetics*, vol. 23, no. 3, 1993, pp. 665-685.

[24] C. Tran, A. Abraham, L. Jain, *Decision Support Systems Using Intelligent Paradigms*, eprint arXiv:cs/0405052, 2004.

[25] Reis, Cecilia, J.A.T. Machado, J.B. Cunha, E.J.S. Pires, "Evolutionary computation in the design of logic circuits". In: *IEEE International Conference on Systems, Man and Cybernetics*, 7-10 Oct. 2007, pp. 1664-1669.

[26] R. Eberhart, Y. Shi, Computational Intelligence. *Concepts to Implementations*, Elsevier, Morgan Kaufman Publisher, ISBN 978-1-55860-759-0; 2007.

[27] H. Pohlheim, *GEATbx Introduction to Evolutionary Algorithms: Overview, Methods and Operators*, version 3.7 (November 2005), 2005, http://www.geatbx.com/.

[28] H. Mühlenbein, D. Schlierkamp-Voosen "Predictive models for the breeder genetic algorithm in continuous parameter optimization", *Evolutionary Computation*, vol. 1, issue 1, ISSN:1063-6560, 1993, pp. 25-49.