We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter 13

Simulated Annealing to Improve Analog Integrated Circuit Design: Trade-Offs and Implementation Issues

Lucas Compassi Severo, Alessandro Girardi, Alessandro Bof de Oliveira, Fabio N. Kepler and Marcia C. Cera

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772//45872

1. Introduction

The design of analog integrated circuits is complex because it involves several aspects of device modeling, computational methodologies, and human experience. Nowadays, the well-stablished CMOS (Complementary Metal-Oxide-Semiconductor) technology is mandatory in most of the integrated circuits. The basic devices are MOS transistors, whose manufacturing process is well understood and constantly updated in the design of small devices. Detailed knowledge of the devices technology is needed for modeling all aspects of analog design, since there is a strong dependency between the circuit behavior and the manufacturing process.

Contrary to digital circuits, which are composed by millions (or even billions) of transistors with equal dimensions, analog circuits are formed by tens of transistors, but each one with a particular geometric feature and bias operation point. Digital design is characterized by the high degree of automation, in which the designer has low influence on the resulting physical circuit. The quality of the CAD (Computer-Aided Design) tools used for circuit synthesis is much more important than the designer experience. These tools are able to deal with a large number of devices and interconnections. Digital binary circuits have robustness characteristics in which the influence of non-linearities and non-idealities are not a major concern. Furthermore, mathematical models of devices for digital circuits are relaxed and computationally very efficient.

On the other hand, analog design still lacks from design automation. This is a consequence of the problem features and the difficulty of implementing generic tools with high design accuracy. Thus, the complex relations between design objectives and design variables result in a highly non-linear n-dimensional system. Technology dependency limits the design automation, since electrical behavior is directly related to physical implementation. In

INTECH
open science | open minds

©2012 Girardi et al., licensee InTech. This is an open access chapter distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

addition, the large number of different circuit topologies, each one with unique details, makes modeling a very difficult task.

In general, the traditional analog design flow is based on the repetition of manual optimization and SPICE (Simulation Program with Integrated Circuit Emphasis) electrical simulations. For a given specification, a circuit topology is captured in a netlist containing devices and interconnections. Devices sizes, such as transistors width and length, or resistors and capacitors values, are calculated manually. The verification is performed with the aid of SPICE models and technology parameters in order to predict the final performance in silicon. Specific design goals such as dissipated power, voltage gain, or phase margin are achieved by manual calculation and then re-verified in simulation. Once the final performance is met, the design is passed on to a physical design engineer to complete the layout, perform design rule checks, and layout versus schematic verification. The layout engineer passes the extracted physical design information back to the circuit designer to recheck the circuit operation on the electrical level. When physical effects cause the circuit to miss specifications, several more iterations of this circuit-to-layout loop may be required. This process is repeated for each analog block in the circuit, even for making any relatively simple specification change. The amount of time and human resources used can vary, depending on the design complexity and the designer experience. However, even for a large and most skilled design team, the short time-to-market and strict design objectives are key issues of analog designs. Improvements in the analog design automation can save design time and effort.

In this chapter we analyze the Simulated Annealing (SA) meta-heuristic applied to adjust circuit parameters in transistor sizing automation procedure at electrical level. Previous works have been done in the field of analog design automation to enable fast design at the block level. Different strategies and approaches have been proposed during the evolution of analog design automation, such as simulation-based optimization [5, 9, 17], symbolic simulation [10], artificial intelligence [6], manually derived design equations [4, 21], hierarchy and topology selection [11], geometric programming [12, 16] and memetic algorithms [15]. The main difficulty encountered for wide spread usage of these tools is that they require appropriate modeling of both devices (technology dependent) and circuit topologies in order to achieve the design objectives in a reasonable processing time.

Moreover, the option of choosing different circuit topologies is also difficult to implement in a design methodology or tool, since most approaches work with topology-based equations, limiting the application range. The possibility of adding new block topologies must also be included in the methodology, since it is critical to the design. The usage of optimization algorithms combined with design techniques seems to be a good solution when applied to specific applications. This is because a general solution most often proves to have short comings for fully exploiting the capabilities of the analog CMOS technology. The key requirements of an analog synthesis tool are: interactivity with the user, flexibility for multiple topologies and reasonable response time. The interface with an electrical simulator and with a layout editor is also convenient [8].

The remaining of this chapter is organized as follows. Section 2 explains the Simulated Annealing meta-heuristic, its parameters, and functions. Circuit modeling, as well as the parameters and functions involved, are described in Sec. 3. Afterward, Sec. 4 presents a basic circuit used to explain the usage of SA, how the searches occur, and the results achieved. In Sec. 5, Simulated Annealing is used to seek solutions to a more complex circuit, in which we could analyze the impact of SA parameters and functions as a mean to automate circuit design. Finally, Sec. 6 conclude this chapter with our final remarks and future works.

2. Simulated annealing

The Simulated Annealing (SA) is a well known random-search technique that exploits an analogy with the way a metal heat and slowly freezes into a minimum energy crystalline structure, the so called annealing process. In a more general system, like an optimization problem, it is used for searching the minimum value of a cost function, avoiding getting trapped in local minima. The algorithm employs random searches which, besides accepting solutions that decrease (i.e. minimize) the objective cost function, may also accept some that increase it. The latter are called "indirect steps", and are allowed in order to escape from local optima.

The SA algorithm uses a *cooling function* $T(t)$, which maps a time instant t to a temperature T, decreasing T as t increases. At each iteration, new steps are randomly taken, based on a *probabilistic state generation function* $g(X)$, leading to new states in the solution space. In this context X is a vector of d parameters, where d is the dimensionality of the solution space. If a step leads to a state with a worse solution, it is only effectively taken, i.e. the new state is accepted, with a probability less than 1. States with better solutions are always accepted. This probability is given by an *acceptance function* $h(\Delta F)$:

$$h(\Delta F) = \frac{1}{1 + exp(\Delta F/T)} \tag{1}$$

Here, $\Delta F = F_{t+1} - F_t$ represents the variation of the cost function calculated at two consecutive times steps F_{t+1} and F_t.

The algorithm is able to reach an optimal solution on the choice of the *cooling function* and *probabilistic state generation function*. If the temperature in *cooling function* decreases too fast, the search will run faster, but the SA algorithm is not guaranteed to find the global optimum anymore [13]. This may be acceptable if a solution is needed in a small amount of time and the solution space is well-know or presents high dimensionality. This is called Simulated Quenching (SQ) [1], and is useful when an approximate solution is sufficient. There are some common sets of options to choose from when implementing an SA algorithm. They are described below.

2.1. Boltzmann annealing

The Boltzmann annealing is the classical simulated annealing algorithm, using physics principles to choose the *probabilistic state generation function* in order to ensure convergence to a global minimum. It employs a Gaussian distribution for generating new states:

$$g_{Boltz}(X) = \frac{1}{(2\pi T(t))^{d/2}} \exp\left(-\frac{(\Delta X)^2}{2T(t)}\right) \tag{2}$$

Here, $\Delta X = X - X_0$ and d is the number of dimensionality of the search space. The Boltzman *cooling function* is described as:

$$T_{Boltz}(t) = \frac{T_0}{\log(t)} \tag{3}$$

where T_0 is the initial temperature, and t is the time step.

Geman and Geman in the classical paper [7] have proved that using Gaussian distribution to generate new states (Eq. 3) with the Boltzman *cooling function* (Eq. 2) is sufficient to reach global minimum of an optimization function at infinite time.

2.2. Fast annealing

Fast Annealing is a variant of the Boltzmann Annealing [20] that uses as *probabilistic state generation function* the Cauchy distribution:

$$g_{Fast}(X) = \frac{T}{(\Delta X^2 + T(t))^{(d+1)/2}} \tag{4}$$

One advantage of the Cauchy distribution over the Gaussian distribution is its fatter tail. When the temperature decreases, the Cauchy distribution generates new states with a lower dispersion than states generated by a Gaussian distribution. In this way the converge using Cauchy distribution becomes faster.

However, in order to guarantee that the algorithm reaches the global minimum, a special *cooling function* is used:

$$T_{Fast}(t) = \frac{T_0}{t} \tag{5}$$

Where T_0 is the initial temperature, and t is the time step. It is important to show that the *cooling function* used in Boltzmann Annealing (equation 3) decreases more slowly than the *cooling function* used in Fast Annealing (equation 5). This characteristic turns the convergence of Fast annealing faster than Boltzmann annealing.

2.3. Reannealing

The reannealing method [14] raises the temperature periodically after the algorithm accepts a certain number of new states or after a given number of iterations. Then the search is restarted with a new annealing temperature. The reannealing objective is to avoid local minima, which presents interesting results when applied in nonlinear optimization problems.

2.4. Simulated Quenching

Simulated Quenching (SQ) [1], described before, is useful when an approximated solution is sufficient and there is a need of faster execution time. An example of the function that can be used to decrease the temperature faster is the exponential *cooling function* shown below.

$$T_{Exp}(t) = T_0 \cdot 0.95^t \tag{6}$$

Using this *cooling function* with Boltzmann *state generation function* (Eq. 2) or Fast *state generation function* (Eq. 4) will turn the optimization faster, but without convergence guarantee.

3. Circuit modeling

In order to design an analog integrated circuit with Simulated Annealing optimizations it is necessary to develop a cost function describing the analog circuit behavior. There are two ways to analyze a circuit behavior. One is based on simplified equations as cost functions, which represent the circuit. This is the faster alternative, but has low precision and limits the solutions in some regions of circuit operation. The other way is to use an external SPICE electrical simulator to evaluate the circuit with a complete model. This alternative provides better accuracy, but demands more computational power.

In this work the second alternative is used, with the electrical simulation performed by Synopsys HSpice ®. In the optimization procedure of analog integrated circuit design, the heuristic parameters are the MOSFET transistor sizes W (channel width) and L (channel length), voltage and current sources bias, and capacitors and resistor values. The design flow using Simulated Annealing proposed in this chapter is shown in Figure 1 .

The proposed methodology has three specification structures as inputs:

- *Design constraints* that represent all functions of circuit specifications and variable bounds;
- A *technology file* containing simulation model parameters for the MOSFET transistors; and
- *SA Options* for the configuration of the SA heuristic, such as temperature function, annealing function and stop condition.

The methodology starts with the initial solution generation that is provided by random generated numbers according to the variables bounds values. The circuit specifications of the generated solution are then evaluated by the cost function, which uses the external electrical SPICE simulator. Thereafter, the SA temperature parameter is initialized with the value specified in the SA options.

Thereafter, a new solution is generated by the SA state generation function (see Section 2), and evaluated by the cost function by means of electrical simulations. The new solution is compared with the current solution and, if it has a lower cost function value, it replaces the current solution. Otherwise, a random number is generated and compared with a probability parameter: if it is greater, the current solution is replaced by the new solution; if smaller, the new solution is rejected.

Finally, the stopping conditions are verified and, if satisfied, the optimization process ends. If not satisfied, the temperature parameter is reduced by the cooling function and the procedure continues. The stopping conditions usually include a minimum value of temperature, a minimum cost function variation, and a maximum number of iterations.

For analog design automation, a multi-objective cost function is necessary to aggregate different - and sometimes conflicting - circuit specifications. A typical multi-objective cost function can be:

$$f_c(X) = \sum_{i=1}^{n} S_i(X) + \sum_{j=1}^{m} R_j(X) \tag{7}$$

In this function, the first sum represents optimization specifications (design objectives) and the second one the design constraints. $S_i(X)$ is the i^{th} circuit specification value and $R_j(X))$ is the j^{th} constraint function. Both are functions of the vector X of design parameters and are normalized and tuned according to the desired circuit performance.

$R_j(X)$ is a function that is dependent on the specification type: minimum required value ($R_{min}(X)$) or maximum required value ($R_{max}(X)$) [3]. These functions are shown in Fig. 2, where a is the maximum or minimum required value and b is the bound value between acceptable and unacceptable performance values. Acceptable but non-feasible performance values are that points between a and b. They return intermediate values for the constraints functions in order to allow the exploration of disconnect feasible design space regions. These functions return additional cost for the cost function if the performance is outside the desired range. Otherwise, the additional cost is zero.

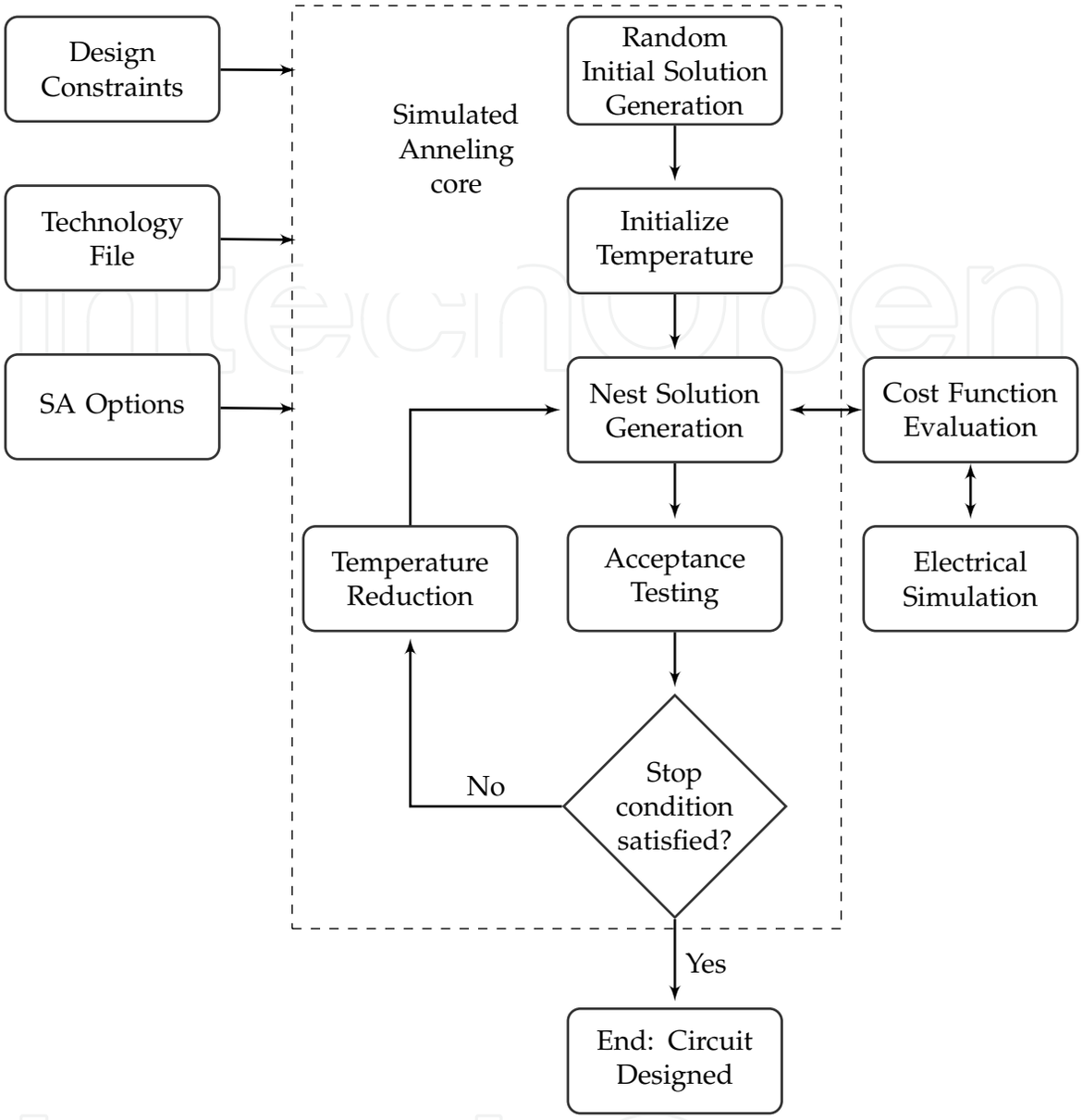


Figure 1. Analog integrated circuit sizing with Simulated Annealing flow.

4. Basic analysis of the search space

This section presents a simple case study, a differential amplifier, to introduce and explain the usage of Simulated Annealing to automate the design of analog integrated circuit. Section 4.1 describes the features of the differential amplifier. Sec. 4.2 explains the modeling of the differential amplifier that allows its simulation and the usage of the SA. Finally, to improve the automation process, some optimization options on SA are applied and their results are discussed in Sec. 4.3 .

4.1. Case study: Differential amplifier

A differential amplifier is a basic analog building block used in general as the input stage of operational amplifiers. Perhaps its simplicity, it is very useful as a first voltage

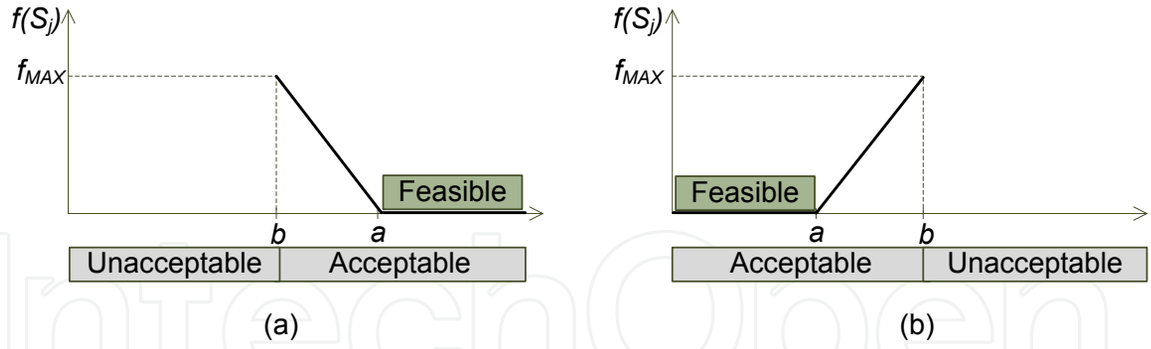


Figure 2. Cost function performance metrics: (a) minimum required value specifications and (b) maximum required value specifications.

amplification stage of many electronic devices and has become the dominant choice in today's high-performance analog and mixed-signal circuits [19]. Ideally, it amplifies the difference between two voltages but does not amplify the common-mode voltages. An implementation of the differential amplifier with CMOS transistors and active load is shown in Fig. 3 . It is composed by a differential pair formed by two input transistors ($M1$ and $M2$), an active current mirror ($M3$ and $M4$) and an ideal tail current source I_{ref}. The output voltage V_{out} depends on the difference between the input voltages V_{in1} and V_{in2}. For a small difference between V_{in1} and V_{in2}, both $M2$ and $M4$ are saturated, providing a high gain. Otherwise, if $|V_{in1} - V_{in2}|$ is large enough, $M1$ or $M2$ will be off and the output will be stuck at $0V$ or at V_{DD}.

The output voltage of the differential amplifier can be expressed in terms of its differential-mode and common-mode input voltages as

$$V_{out} = A_{VD}(V_{in1} - V_{in2}) \pm A_{VC}\left(\frac{V_{in1} + V_{in2}}{2}\right) \tag{8}$$

where A_{VD} is the differential-mode voltage gain and A_{VC} is the common-mode voltage gain. An ideal operational amplifier has an infinite A_{VD} and zero A_{VC}. Although practical implementations try to find an approximation to these values, the implementation of physical circuits insert some non-idealities that limit A_{VD} and A_{VC}.

Another important characteristic of a differential amplifier is the input common-mode range ($ICMR$). We can estimate ICMR by setting $V_{in1} = V_{in2}$ and vary input common-mode voltage (DC component of V_{in1} and V_{in2}) until one of the transistors in the circuit is no longer saturated [2]. The highest common-mode input voltage ($ICMR^+$) is

$$ICMR^+ = V_{DD} - V_{SG3} + V_{TN1} \tag{9}$$

Here, V_{SG3} is the source-voltage of transistor $M3$ and V_{TN1} is the threshold voltage of $M1$. The lowest input voltage at the gate of $M1$ (or $M2$) is found to be

$$ICMR^- = V_{SS} + V_1 + V_{GS2} \tag{10}$$

The voltage at node 1 (V_1) is determined by the physical implementation of the current source I_{ref}, which in general is a single transistor whose drain current is controlled by its gate voltage. V_{GS2} is the gate-source voltage of transistor $M2$.

The small-signal properties of the differential amplifier can be accomplished with the assistance of the simplified model shown in Fig. 4, which ignores body effect. In this figure, gm is the gate transconductance given by the derivative of the drain current in relation to gate-source voltage:

$$gm = \frac{\partial I_D}{\partial V_{GS}} \tag{11}$$

The series resistance rds is the inverse of the output conductance gds and can be estimated in small-signal analysis as

$$\frac{1}{rds} = gds = \frac{\partial I_D}{\partial V_{DS}} \tag{12}$$

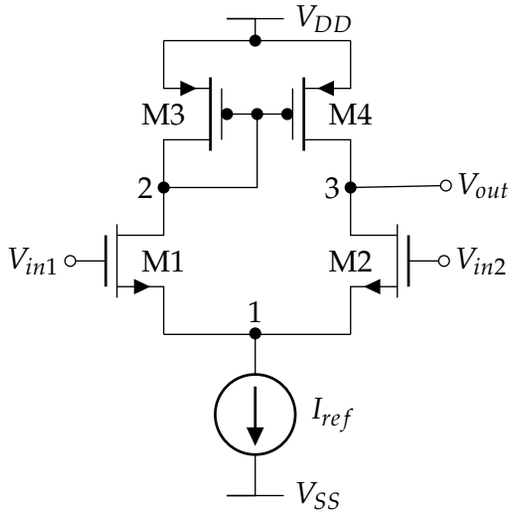


Figure 3. Schematics of a CMOS differential amplifier.

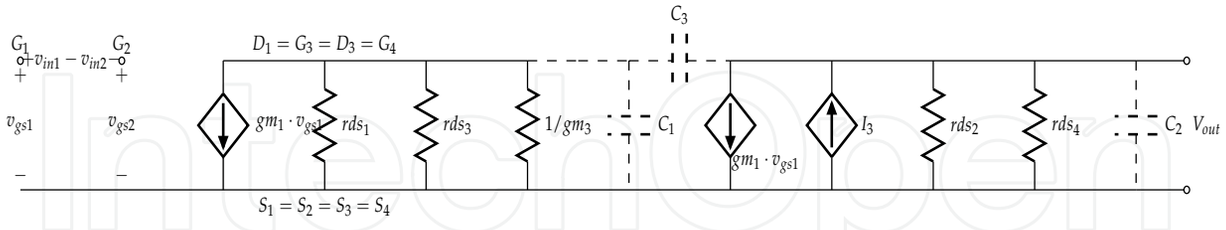


Figure 4. Simplified small-signal model for the CMOS differential amplifier.

The small-signal voltage gain A_{vo}, *i.e.*, the relationship between V_{out} and the differential input voltage $V_{in1} - V_{in2}$, can be estimated in low frequencies by

$$A_{vo} = \frac{gm_1}{gds_2 + gds_4} \tag{13}$$

For higher frequencies, the voltage gain is modified due to the various parasitic capacitors at each node of the circuits, modeled by C_1, C_2 and C_3, which are calculated as follows:

$$C_1 = C_{gd1} + C_{bd1} + C_{bd3} + C_{gs3} + C_{gs4} \tag{14}$$

$$C_2 = C_{bd2} + C_{bd4} + C_{gd2} + C_L \tag{15}$$

$$C_3 = C_{gd4} \tag{16}$$

Considering C_3 approximately zero, the voltage-transfer function can be written as

$$V_{out}(s) \cong \frac{gm_1}{gds_2 + gds_4}\left[\left(\frac{gm_3}{gm_3 + sC_1}\right)V_{gs1}(s) - V_{gs2}(s)\right]\frac{\omega_2}{s+\omega_2} \tag{17}$$

where ω_2 is given as

$$\omega_2 = \frac{gds_2 + gds_4}{C_2} \tag{18}$$

The pole ω_2 determines the cut-off frequency of the amplifier and is also called as ω_{-3dB}. Assuming that

$$\frac{gm_3}{C_1} \gg \frac{gds_2 + gds_4}{C_2} \tag{19}$$

then the frequency response of the differential amplifier reduces to

$$\frac{V_{out}(s)}{V_{in1}(s) - V_{in2}(s)} \cong \left(\frac{gm_1}{gds_2 + gds_4}\right)\left(\frac{\omega_2}{s+\omega_2}\right) \tag{20}$$

This first-order analysis leads to a single pole at the output given by $-(gds_2 + gds_4)/C_2$. Some zeroes occur due to C_{gd1}, C_{gd2} and C_{gd4}, but they can be ignored in this analysis. The gain-bandwidth product (GBW), which is the equal to the unity-gain frequency, can be expressed as

$$GBW = A_{vo} \cdot \omega_{-3dB} \tag{21}$$

The slew-rate (SR) performance of the CMOS differential amplifier depends the value of I_{ref} and the capacitance from the output node to ac ground and is given by

$$SR = \frac{I_{ref}}{C} \tag{22}$$

where C is the total capacitance connected to the output node (approximated by C_2 in our analysis).

Other important specifications for the electrical behavior of the differential amplifier includes power dissipation $P_{diss} = I_{ref} \cdot (V_{DD} - V_{SS})$ and total gate area, calculated as the sum of the product of gate width and lenght of all transistors that compose the circuit:

$$Area = \sum_i W_i \cdot L_i \tag{23}$$

All analog design has a target fabrication technology and a device type, in which the set of transistor model parameters is unique. These parameters determines the electrical characteristics - such as drain current, gate transconductance and output conductance - of the active devices that are part of the circuit. The specifications described before are function of these parameters, together with W and L. Since the parameters are fixed for a given fabrication

technology, the designer has as free variables only the gate sizes. Gate sizing is, in effect, the task of analog design.

4.2. Modeling the differential amplifier for automatic synthesis

The modeling of the differential amplifier of Fig. 3 for automatic synthesis is straightforward. Using a simulation-based approach, the circuit specifications are calculated by SPICE electrical simulations. As an example, let us consider the multi-objective design of a differential amplifier that must be optimized in terms of voltage gain A_vo and positive input common-mode range $ICMR^+$. Also, there is a list of constraints containing a series of specifications that must be met hardly. Table 1 summarizes the design objectives and constraints for this problem.

Specification	Value
A_v	maximize
$ICMR^+$	maximize
$Area$	$< 120\mu m^2$
PM	$> 70^\circ$
GBW	$> 100MHz$

Table 1. Design specifications and constraints for the differential amplifier of Fig 3 .

The cost function $f_c(X)$ is than formulated as a sum of design specifications and constraints in terms of the vector of the design variables X:

$$f_c(X) = 3 \cdot \frac{ICMR^+(X)}{ICMR^+_{ref}} + \frac{A_{vo}(X)}{A_{vo(ref)}} + R(X) \tag{24}$$

The specifications are calculated for a given X and normalized by a reference value. In this example, $ICMR^+_{ref} = 1.3V$ and $A_{vo(ref)} = 20dB$. The ponderation of each specification can be implemented with individual weights which indicate the relative importance of the parameter. In this example, we choose a weight of 3 for $ICMR^+$ and 1 for A_vo. $R(X)$ is a constraint function which is also a function of X, calculated as follows:

$$R(X) = \frac{R_{max}(Area(X), Area_{ref})}{4} + R_{min}(PM(X), PM_{ref}) + R_{max}(GBW(X), GBW_{ref}) \tag{25}$$

Here, $R_{max}(S(X), S_{ref})$ and $R_{min}(S(X), S_{ref})$ are constraint functions of maximum and minimum, respectively, in terms of the specification $S(X)$ and the reference value S_{ref}. For example, the constraint of gate area is related to $R_{max}(S(X), S_{ref})$, because it can not be larger than a reference value of $Area_{ref}$. The same occurs for GBW, which can not be smaller than GBW_{ref}, whose constraint is modeled by the function $R_{min}(S(X), S_{ref})$. Both constraint functions insert a penalty value in the cost function $f_c(X)$ if the specification is outside the expected range. Otherwise, they return zero. The following equations show how the constraint functions are implemented:

$$R_{max}(S(X), S_{ref}) = \begin{cases} 0 & \text{if } S(X) \leq S_{ref} \\ \frac{S(X) - S_{ref}}{S_{ref}} & \text{if } S(X) > S_{ref} \end{cases} \tag{26}$$

$$R_{min}(S(X), S_{ref}) = \begin{cases} 0 & \text{if } S(X) \geq S_{ref} \\ \frac{S(X)-S_{ref}}{S_{ref}} & \text{if } S(X) < S_{ref} \end{cases} \quad (27)$$

We used in this example the constraint reference values shown in Tab. 1 . In order to simplify the analysis, we consider that all transistors of the circuit are of the same size. It is not a practical approach, since transistor $M1$ must be equal to $M2$, but not necessarily equal to $M3$ and $M4$. However, this simplification allows the 2-D visualization of the problem and can be used to explain design trade-offs and automatic optimal search, providing an intuitive notion of the problem. So, we will consider in this analysis two free variables: $L = L_1 = L_2 = L_3 = L_4$ and $W = W_1 = W_2 = W_3 = W_4$. In this case, $X = [W \ L]$.

The design space for Eq. 24 was fully mapped by electrical simulation varying W and L from $1\mu m$ to $100\mu m$ with a step of $1\mu m$. The target technology node was $0.35\mu m$ $3.3V$ CMOS. Fig. 5 shows the plotted design space as a function of W and L. It is possible to note the highly non-linear nature of the generated function and the existence of a valley in which is localized a minimum value. The optimal solution for this sizing problem, *i.e.*, the minimum value of the design space, is known exactly in this case and is located at $W = 8\mu m$ and $L = 20\mu m$, with the value of -1.9623.

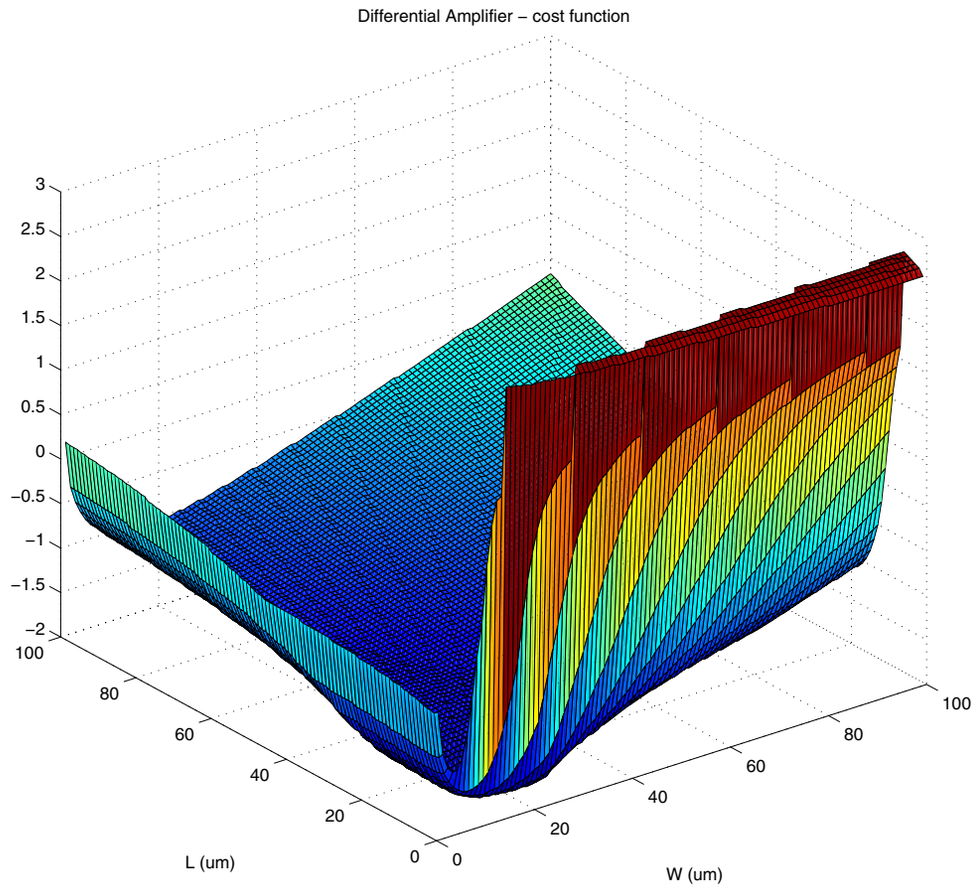


Figure 5. Two-variables design space for a differential amplifier. The minimum is at $W = 8\mu m$ and $L = 20\mu m$, with the value of -1.9623.

4.3. Optimization of a differential amplifier

For the analysis of Simulated Annealing options and the influence over the automatic sizing procedure of analog basic blocks, we will explore different configurations of temperature schedule, state generation function and reannealing for global optimization and further local optimization. Due to the random nature of some parameters of SA, an statistic analysis is needed to understand the search behavior. We performed 1000 optimization runs for each temperature schedule function described before: Boltzman, Exponential and Fast. The state generation function was kept fixed as $g_{Boltz}(X)$ (Eq. 2). Each execution started with a different seed for the random number generator function. The same parameters were used for the three functions, including the same random number vector for a fair comparison. A MATLAB script was implemented and the native SA method (`simulannealbnd`) was used as the main bound constrained optimization function.

4.3.1. Global optimization

Table 2 shows the mean of the optimal cost function found after 1000 executions of the optimization procedure for each temperature schedule function. The Boltzman schedule achieved the best values, with a mean final cost $\overline{f_c^*}$ of -1.960306, right near the optimal global solution of -1.9623. It means that most of the solutions provided by the procedure with Boltzman are near the global optimum. Exponential and Fast temperature schedules demonstrate worst results in terms of cost. Boltzman result was obtained at expense of a higher execution time and total number of iterators.

Temperature schedule	$\overline{f_c^*}$	Execution time (s)	Iterations
Boltzman	-1.960306	16.32	1777.44
Exponential	-1.834720	9.57	1043.89
Fast	-1.579269	12.99	1416.36

Table 2. Mean values of differential amplifier global optimization procedure for different temperature schedule functions after 1000 executions.

The free variables W and L found by the three temperature schedule functions are shown in Tab. 3. Again, Boltzman demonstrates the best results, with the mean values near the optimal solution. Fast schedule presents the worst results in this configuration.

Temperature schedule	$\overline{W}$ (μm)	$\overline{L}$ (μm)
Boltzman	8.07	20.57
Exponential	11.47	23.00
Fast	34.46	26.32
Optimum value	8.00	20.00

Table 3. Mean W and L values achieved by global optimization procedure of the differential amplifier after 1000 runs for each different temperature schedule function.

Fig. 6 shows a graph comparing the 3 temperature schedules, considering only the optimum solutions obtained in relation to the optimization time. It is possible to notice the very attractive results for Boltzman, which achieved 400 optimum solutions (over a set of 1000 executions) in about 25 seconds of execution time. After this time, the number of optimal solutions does not grow considerably, saturating in 430 at 37 seconds. The same saturation behavior happens with Exponential and Fast temperature schedules, but with a very lower number of optimal solutions and at early execution time.

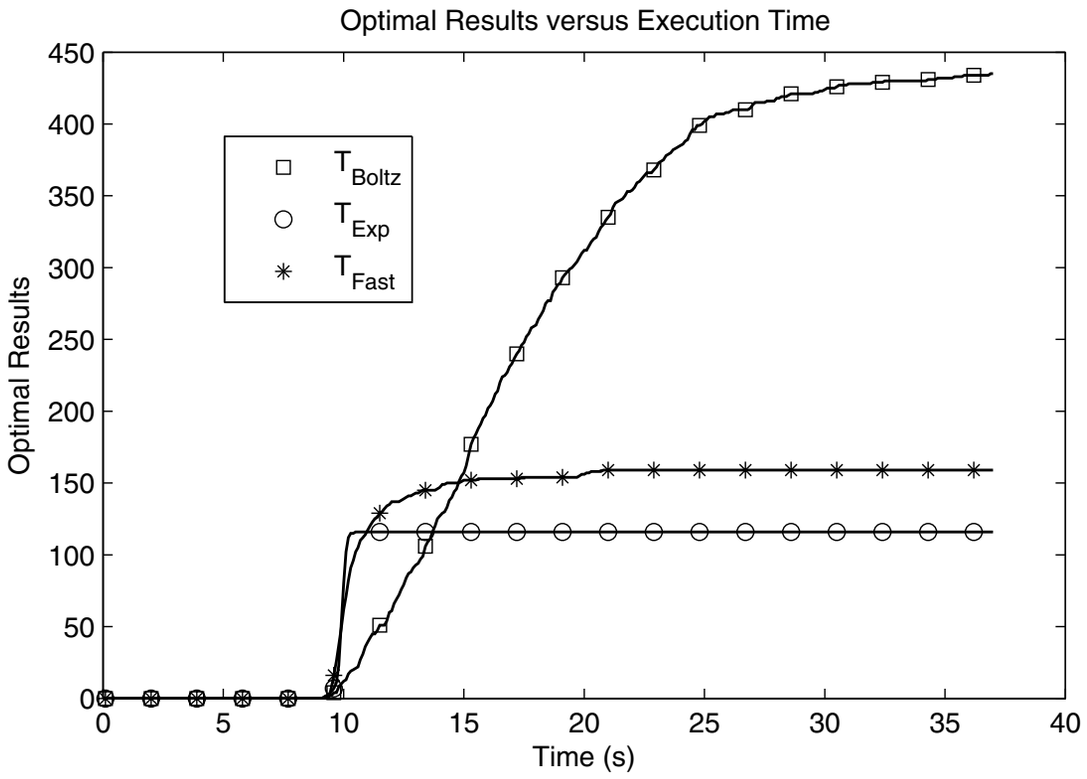


Figure 6. Optimal results versus execution time for the global optimization of a differential amplifier, considering 3 different temperature schedule functions.

4.3.2. Global followed by local optimization

In order to improve the results obtained by global optimization with Simulated Annealing, we apply a local optimization algorithm over the previous set of solutions generated by SA with the three temperature schedule functions. We choose the interior point algorithm [18], which is suitable for linear and non-linear convex design spaces. We suppose that the design space region near the solution provided by the global optimization and evolving the global optimum solution is convex and can be explored by this method. The algorithm was implemented by using the MATLAB native function `fmincon`. The results can be seen in Tab. 4 . It is clear the improvement obtained by the local optimization. The mean final cost of the 1000 executions for the three temperature schedules are close to the known global optimum of -1.960306. The total execution time (including global followed by local execution times) was increased by about 50%, but it is still in a reasonable value, near 20 seconds.

Temperature schedule	$\overline{f_c^*}$	Execution time (s)
Boltzman	-1.961759	24.40
Exponential	-1.927647	18.51
Fast	-1.786238	22.43

Table 4. Mean values of differential amplifier after local optimization over the results obtained by global optimization shown in Tab. 2.

The mean values found for the free variables after the local search are shown in Tab. 5 . Comparing to the previous values provided by the global optimization, it is possible to note the great improvement of the Exponential temperature schedule, whose mean W and L approached very near to the global optimum.

Temperature schedule	$\overline{W}$ (μm)	$\overline{L}$ (μm)
Boltzman	8.07	19.99
Exponential	9.68	20.30
Fast	20.12	21.59
Optimum value	8.00	20.00

Table 5. Mean W and L values achieved by local optimization procedure of the differential amplifier over the results obtained by global optimization shown in tab. 3.

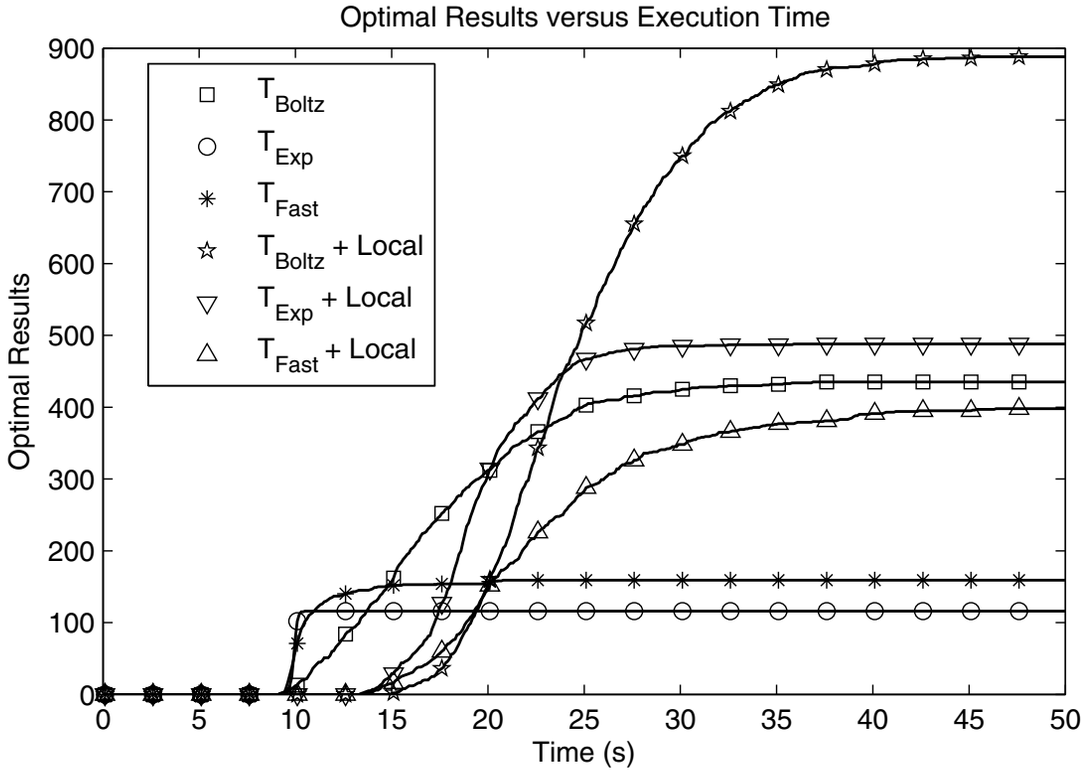


Figure 7. Optimal results versus execution time for the global optimization of a differential amplifier, considering 3 different temperature schedule functions - global and local.

In terms of the number of optimal solutions found over the 1000 executions, the local search also demonstrate an improvement. Fig. 7 shows the results obtained, in which we can see that, for Boltzman schedule, almost 90% of the final solutions are optimal, an improvement of more than 50% over the global optimization. The same occurs for the other temperature schedules.

We can observe the improvement in the number of optimal solutions with local search in Fig. 8, which presents the frequency histogram of the resulting final cost provided by global search (Fig. 8(a)) and global search followed by local search (Fig. 8(b)) for the 3 different temperature schedules. Besides the increase in the number of optimal solutions found, the inclusion of local search after global search also approximated the remaining non-optimal solutions in the direction to the best known value.

4.3.3. Global optimization with reannealing

For the analysis of the influence of reannealing in the optimization process, we performed some experiments executing Simulated Annealing with reannealing intervals of 200, 450, 700 and 950 iterations. Again, 1000 executions were done in order to guarantee a statistical analysis for the three temperature schedule functions described before.

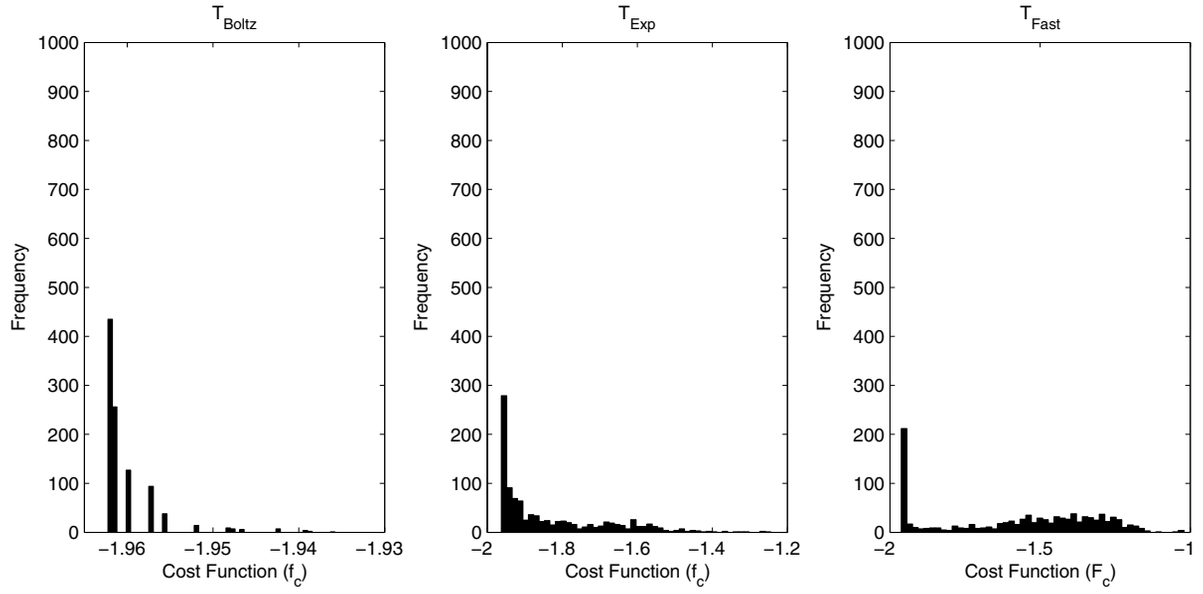
Fig. 9 shows the relation between the number of optimal solutions found by Boltzman schedule function versus the execution time for reannealing intervals from 200 to infinite (*i.e.*, no reannealing). Reannealing interval affects the number of optimal solutions in this case. As the interval decrease, the number of optimal solutions decrease too. The best configuration is with no reannealing, demonstrating that it is not interesting to use reannealing with T_{Boltz}. It happens because the temperature decreases slowly at the beginning of the annealing process. With the reannealing, the temperature increases for higher values before the search in the design space reaches a path trending to the optimal solution.

When the temperature schedule function is modified to Exponential, the behavior is opposite. As the reannealing interval decreases, more optimal solutions are found. Fig. 10 shows the relation between optimal solutions found and execution time for this temperature schedule configuration.

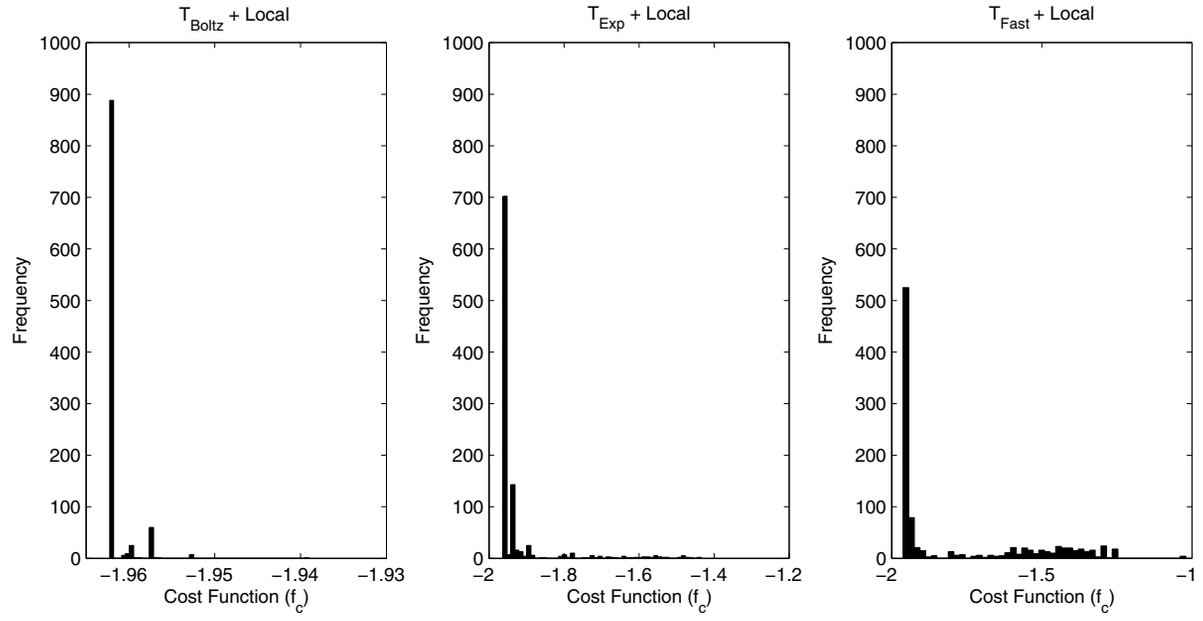
The same occurs for the Fast temperature schedule function, shown in Fig. 11. As the reannealing interval diminishes, the number of optimal solutions increases. This behavior is maintained for ever small intervals. A high improvement in the number of optimal solutions is obtained for reannealing intervals in the order of 100 iterations, as shown in Fig. 12. As the temperature decreases very fast, the reannealing allows to avoid local minima. Thus, it increases the chances of finding the correct path to the optimum solution. Also, we can observe the existence of an optimum value for the reannealing interval which returns the maximum number of optimal solutions.

4.3.4. Analysis of state generation function

The variation of the state generation function is also a factor that can change the convergence of the Simulated Annealing algorithm. Two of these functions are analyzed here: Boltzman and Fast. The combinations of temperature schedule function and state generation function produce distinct results for the synthesis of the differential amplifier. Fig. 13 shows



(a) Global search with Simulated Annealing.



(b) Global search with Simulated Annealing followed by local search with Interior Point Algorithm.

Figure 8. Frequency histograms of the final cost found by the optimization process for three different temperature schedule functions: Boltzman, Exponential and Fast. Obs.: x-scales are different in each chart for better visualization purpose.

the number of optimal solutions returned by the algorithm after 1000 executions for 6 combinations.

We can notice that there are a great improvement in the quality of the solutions using Boltzman temperature schedule together with Boltzman state generation function. This is the best combination, according to that was theoretical predicted in Section 2.

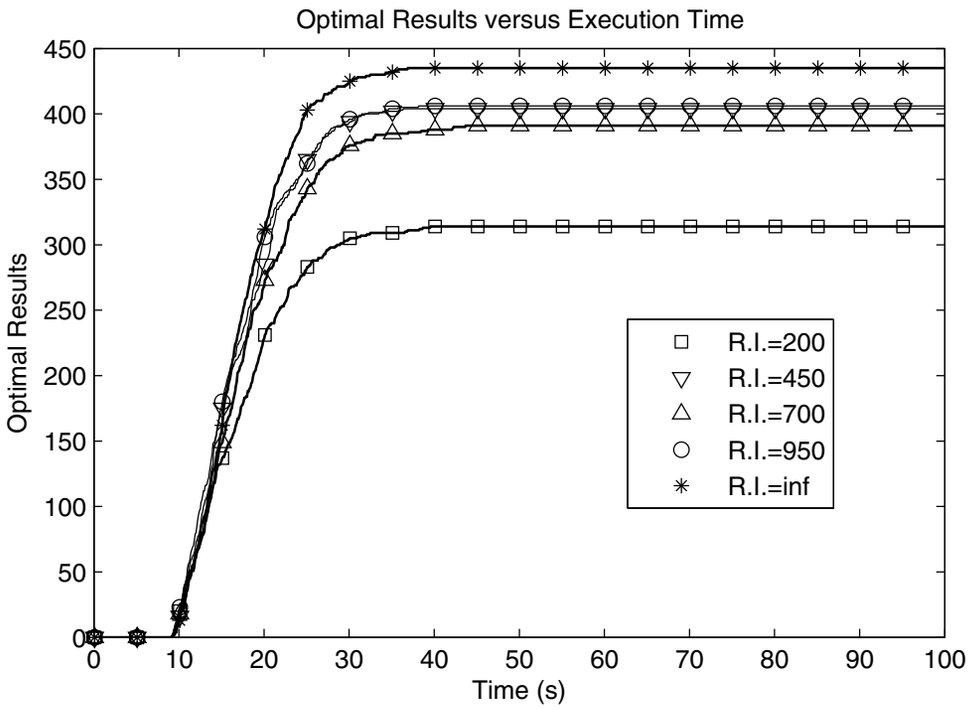


Figure 9. Optimal results versus execution time for the global optimization of a differential amplifier with Boltzman temperature schedule function and different reannealing intervals.

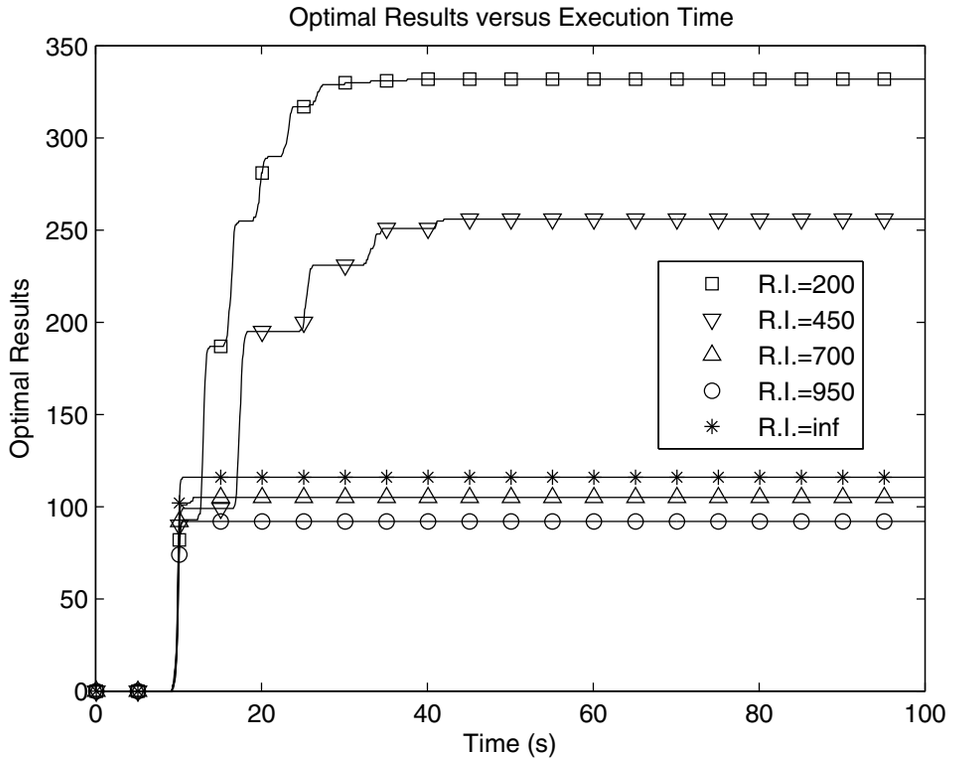


Figure 10. Optimal results versus execution time for the global optimization of a differential amplifier with Exponential temperature schedule function and different reannealing intervals.

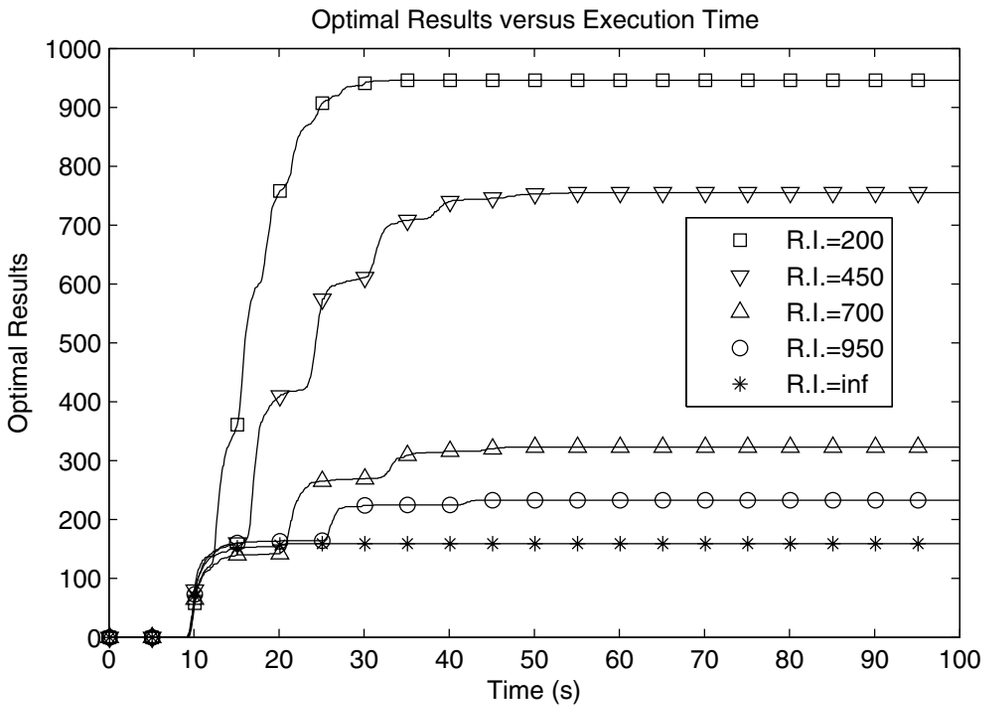


Figure 11. Optimal results versus execution time for the global optimization of a differential amplifier with Fast temperature schedule function and different reannealing intervals.

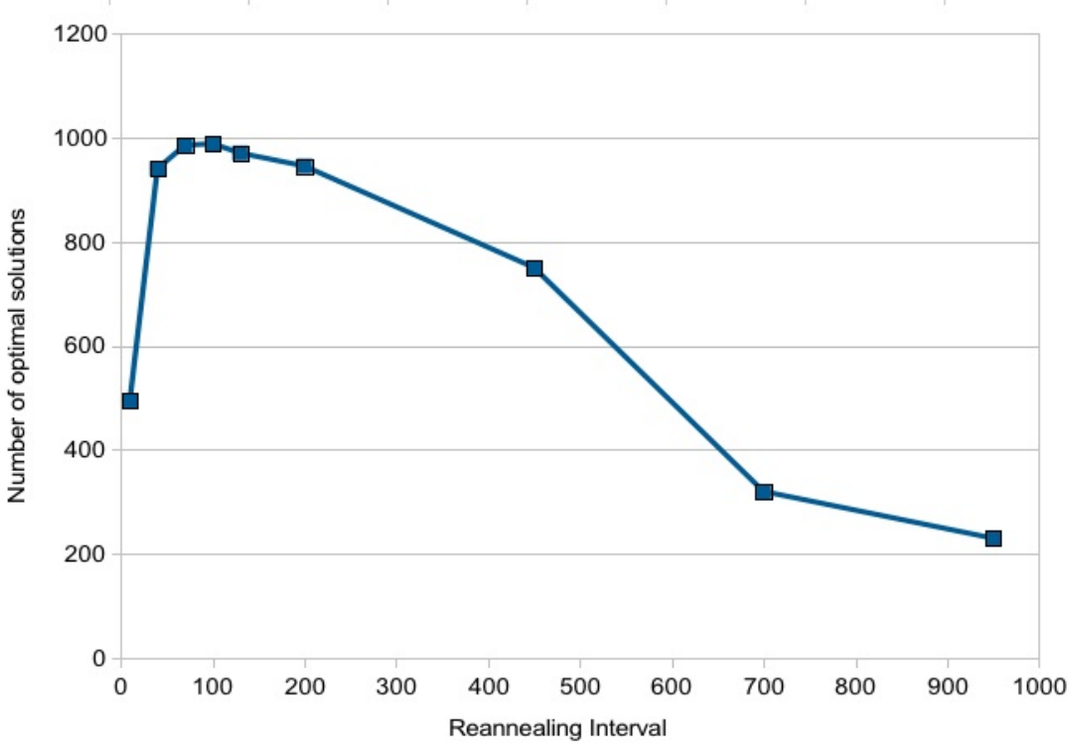


Figure 12. Maximum number of optimal results returned by the optimization process versus reannealing interval for Fast temperature schedule. The optimum value for the reannealing interval is near 100.

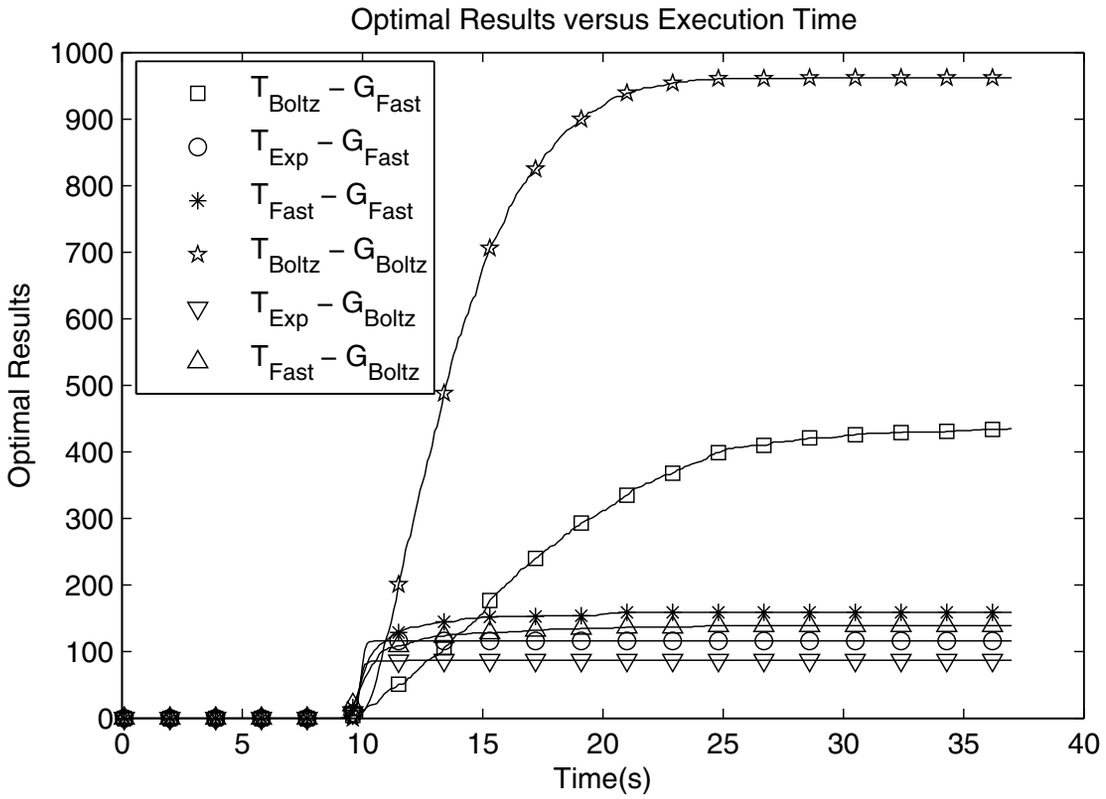


Figure 13. Number of optimal results returned by the optimization process for the differential amplifier for different annealing functions and temperature function schedules.

4.3.5. Analysis of best SA options for the differential amplifier

Results presented before allow us to suppose that the temperature schedule function affects directly the quality of the solutions generated by the global optimization algorithm. The Boltzman schedule, followed by a post-processing with a local search algorithm, demonstrate best convergence to the optimal point, at the expenses of a larger execution time. This additional time, however, is not a problem if we consider that the chances of finding the optimal (or near the optimal) solution are increased. For our 2-variables problem, this additional time is irrelevant (about 10s for 1000 executions). For more complex circuits with dozens of variables, the execution time can be a factor of concern. It is increased exponentially with the number of free variables, since the design space grows fast with the number of free variables. We can estimate the design space size $D_s(X)$ as:

$$D_s(X) = \prod_i \frac{x_{i(ub)} - x_{i(lb)}}{x_{i(step)}} \tag{28}$$

where $x_{i(ub)}$ and $x_{i(lb)}$ are upper an lower bounds of variable x_i, respectively, and $x_{i(step)}$ is the minimum step allowed for variable x_i. It is clear that the exploration of the entire design space is hard for a problem with several free variables. An alternative, in this case, is to use the Fast temperature schedule with reannealing, which is also efficient in the design space exploration. Both Boltzman followed by local search and Fast with reannealing achieved the optimal solution in about 90% of the cases. These configurations are candidates to be tested in a larger circuit.

5. Operational amplifier design

In order to apply simulated annealing in a more realistic and practical operational amplifier, we syntesized a folded cascode in CMOS IBM $0.18\mu m$, regular Vt, $1.8V$ technology node. The schematics of this amplifier is shown in Fig. 14.

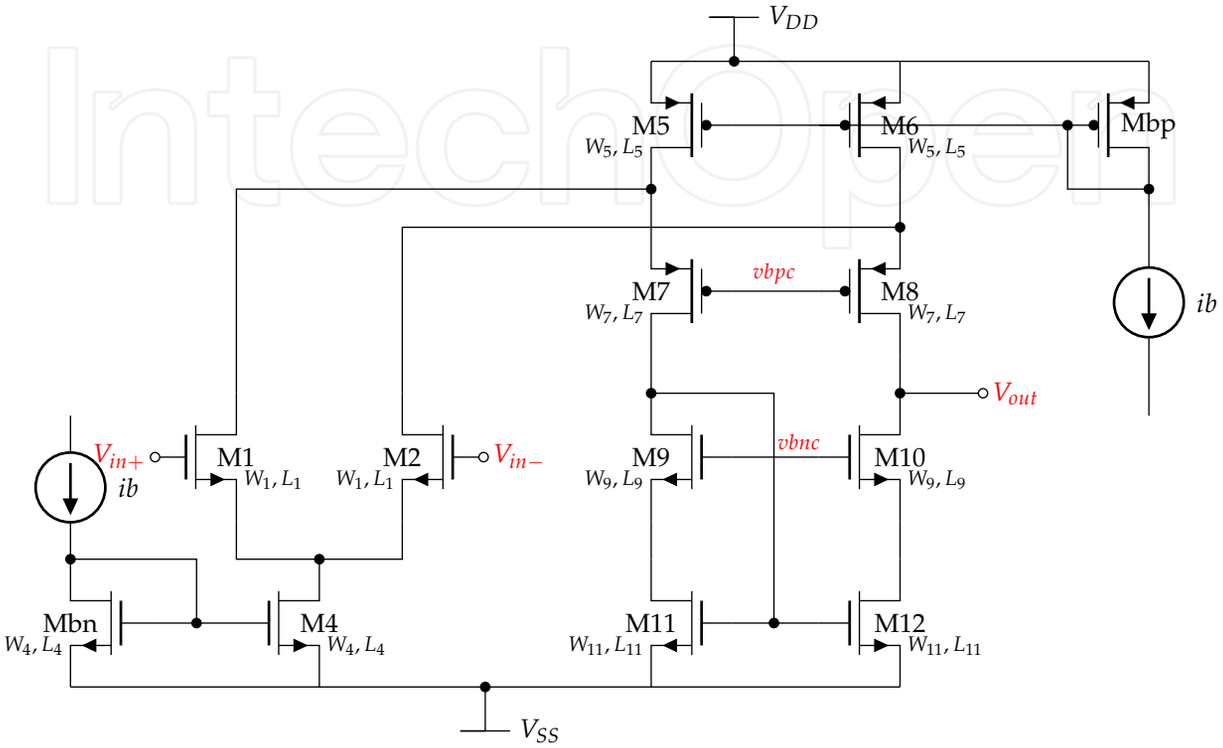


Figure 14. Schematics of a CMOS folded cascode amplifier.

The modeling of this circuit for the proposed optimization process is simple and similar to the previous described modeling of the differential amplifier. The SPICE netlist and the testbench are the information necessary to describe the circuit and bias. The specifications are simulated by an external electrical simulator (HSpice), which returns, for a given set of variables, the electrical characteristics of the circuit. In our design there are 15 free variables, summarized in Tab. 6. It leads to a very large 15-dimensional design space, which is difficult to explore and find the minimum cost value. It is possible to limit the design space inserting constraints in the cost function related to the operation region of each transistor, forcing the devices to operate at saturation ($V_{DS} > V_{GS} - V_T$) and strong inversion ($V_{GS} > V_T$) regions. The specifications and design goals for this circuit are shown in Tab. 7. In the output is connected a capacitive load of $3pF$. We expect to size the circuit optimizing gate area and power dissipation while maintaining the constraints of GBW, low-voltage gain, phase margin and slew rate inside a given range.

Using Boltzman for both temperature schedule function and state generation function, followed by local search with interior point algorithm, we find the final results shown in the third and fourth columns of Tab. 7 for global and global followed by local searches, respectively. It is possible to note that all design objectives were reached, while keeping all devices in the specified operation region. There is an improvement in the multi-objective design goal with the post-processing local search. The final gate area is $145.13\mu m^2$ and

Variable	Final values (our work)	Final values - GENOM ([3])
W_1	11.58 μm	14.91 μm
W_4	22.39 μm	6.99 μm
W_5	14.13 μm	36.78 μm
W_7	30.72 μm	63.04 μm
W_9	7.16 μm	31.45 μm
W_{11}	6.58 μm	7.32 μm
L_1	0.73 μm	1.38 μm
L_4	0.71 μm	1.94 μm
L_5	0.29 μm	0.37 μm
L_7	0.52 μm	0.91 μm
L_9	0.87 μm	0.89 μm
L_{11}	4.54 μm	2.19 μm
$vbnc$	0.0579 V	0.001 V
$vbpc$	-0.0408 V	-0,0449 V
ib	36.78 μA	48.51 μA

Table 6. Free variables and final results found for the folded-cascode amplifier optimization.

dissipated power is 133.2μW. The advantages of this approach is that the resulting circuit is already validated by electrical simulations and does not need to be verified in another design stage.

We can make a direct comparison of the results obtained by this work using SA with other approaches, such as the tools that use genetic algorithms as main optimization heuristic. Although it is difficult to perform a fair comparison with other works in the literature, mainly because the experimental setup in general can not be reproduced with the provided information and there is no standard benchmarks in analog design automation, it is still interesting to compare the general performance of our methodology with other results over similar circuits and design objectives.

In this sense, the results presented by [3] with the GENOM tool are passible to comparison, because the same experimental setup can be reproduced - although some implementation details are not available, such as the parameters of the electrical model. This tool is based on a variation of genetic algorithm as the main optimization heuristic. The folded cascode was implemented in UMC 0.18μm technology. The final results obtained by GENOM for the same circuit synthesized by our approach are summarized in the fiftieth column of Tab. 7.

We can see that both methodologies present similar results for the design constraints. By the other side, both power dissipation and gate area depicted by our approach using Simulated Annealing are about half the final values provided by GENOM. Power dissipation was decreased in 45.5% and gate area in 49%, a great improvement in circuit performance. These results prove that SA is a powerful heuristic for the design of micro-power operational

Specification	Objective	Global (SA Boltz)	Global+Local	GENOM ([3])
GBW	>12MHz	14.86 MHz	14.98 MHz	15.35 MHz
Av0	> 70dB	73.04 dB	70dB	70.61dB
PM	$> 55°$	76.87°	78.76°	79.6°
SR	$> 10V/\mu s$	$10.98V/\mu s$	$11.37V/\mu s$	$15.36V/\mu s$
Area	minimize	$188.25\mu m^2$	$145.13\mu m^2$	$284.7\mu m^2$
Power	minimize	$129.9\mu W$	$133.2\mu W$	$244.6\mu W$

Table 7. Design performance and final results found by the optimization process for the folded-cascode amplifier.

amplifiers. Again, it is important to note that the comparison between the results can not be exact because some parameters in the device electrical model and other configurations are not equal. The final values for the free variables are shown in Tab. 6. We can see that the gate widths of the transistors trend to be larger than the gate lengths and that the magnitudes are similar in both approaches.

6. Conclusion

The design of analog integrated blocks and the search for an optimum design point in a highly non-linear design space evolve different approaches and choices. Simulated Annealing and its variations are a good option for the exploration of this kind of problem. This chapter presented some implications of the algorithm tuning over the final results. We could demonstrate that the correct configuration of SA options can lead to good solutions near the optimality in reasonable execution time. Although it is not clear that some configuration is suitable for sizing all types of analog blocks, it is possible to notice that the approach is correct and, with minimum adjusts for different circuits, SA can be used as a general optimization algorithm, providing good solutions. A direct comparison with a tool based on genetic algorithms for the synthesis of a folded cascode operational amplifier showed that better results can be obtained with the correct design space exploration with SA. As future work, the analysis of parameter variation in the optimization methodology for design centering must be implemented.

Acknowledgements

We would like to thank CNPq and Fapergs Brazilian agencies for supporting this work.

Author details

Lucas Compassi Severo, Alessandro Girardi,Alessandro Bof de Oliveira, Fabio N. Kepler and Marcia C. Cera
Federal University of Pampa – UNIPAMPA, Alegrete Campus
Av. Tiaraju, 810, CEP 97546-550, Alegrete-RS, Brazil

7. References

[1] Aguiar, H., Junior, O., Ingber, L., Petraglia, A., Petraglia, M. R. & Machado, M. A. S. [2012]. *Adaptive Simulated Annealing*, Vol. 35 of *Intelligent Systems Reference Library*, Springer, pp. 33–62.

[2] Allen, P. E. & Holberg, D. R. [2011]. *CMOS Analog Circuit Design*, The Oxford Series in Electrical and Computer Engineering, 3rd edn, Oxford.

[3] Barros, M., Guilherme, J. & Horta, N. [2010]. *Analog Circuits and Systems Optimization based on Evolutionary Computation Techniques*, Vol. 294 of *Studies in Computational Intelligence*, 1st edn, Springer.

[4] Degrauwe, M., Nys, ., Dukstra, E., Rijmenants, J., Bitz, S., Gofart, B. L. A. G., Vitoz, E. A., Cserveny, S., Meixenberger, C., Stappen, G. V. & Oguey, H. J. [1987]. Idac: An interactive design tool for analog cmos circuits, *IEEE Journal of Solid-State Circuits* SC-22(6): 1106–1116.

[5] der Plas, G. V., Gielen, G. & Sansen, W. M. C. [2002]. *A Computer-Aided Design and Synthesis Environment for Analog Integrated Circuits*, Vol. 672 of *The Springer International Series in Engineering and Computer Science*, Springer.

[6] El-Turky, F. M. & Perry, E. E. [1989]. BLADES: an artificial intelligence approach to analog circuit design, *IEEE Trans. on CAD of Integrated Circuits and Systems* 8(6): 680–692.
URL: *http://doi.ieeecomputersociety.org/10.1109/43.31523*

[7] Geman, S. & Geman, D. [1984]. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6(6): 721–741.

[8] Girardi, A. & Bampi, S. [2003]. LIT - an automatic layout generation tool for trapezoidal association of transistors for basic analog building blocks, *DATE*, IEEE Computer Society, pp. 11106–11107.
URL: *http://doi.ieeecomputersociety.org/10.1109/DATE.2003.10028*

[9] Girardi, A. & Bampi, S. [2007]. Power constrained design optimization of analog circuits based on physical gm/id characteristics, *Journal of Integrated Circuits and Systems* 2: 22=28.

[10] Glelen, G., Walscharts, H. & Sansen, W. [1989]. Isaac: A symbolic simulator for analog circuits, *IEEE Journal of Solid-State Circuits* 24(6): 1587–1597.

[11] Harjani, R., Rutenbar, R. A. & Carley, L. R. [1989]. OASYS: a framework for analog circuit synthesis, *IEEE Trans. on CAD of Integrated Circuits and Systems* 8(12): 1247–1266.
URL: *http://doi.ieeecomputersociety.org/10.1109/43.44506*

[12] Hershenson, M., Boyd, S. P. & Lee, T. H. [2001]. Optimal design of a CMOS op-amp via geometric programming, *IEEE Transactions on Computer-Aided Design* 20(1): 1–21.
URL: *http://www.stanford.edu/ boyd/opamp.html*

[13] Ingber, L. [1996]. Adaptive simulated annealing (asa): Lessons learned, *Control and Cybernetics* 25(1): 33 – 54.

[14] Ingber, L. [1989]. Very fast simulated re-annealing, *Mathematical Computer Modelling* 12(8): 967 – 973.

[15] Liu, B., Fernandez, F. V., Gielen, G., Castro-Lopez, R. & Roca, E. [2009]. A memetic approach to the automatic design of high-performance analog integrated circuits, *ACM Transactions on Design Automation of Electronic Systems* 14.

[16] Mandal, P. & Visvanathan, V. [2001]. CMOS op-amp sizing using a geometric programming formulation, *IEEE Trans. on CAD of Integrated Circuits and Systems* 20(1): 22–38.
URL: *http://doi.ieeecomputersociety.org/10.1109/43.905672*

[17] Nye, W., Riley, D. C., Sangiovanni-Vincentelli, A. L. & Tits, A. L. [1988]. DELIGHT.SPICE: an optimization-based system for the design of integrated circuits, *IEEE Trans. on CAD of Integrated Circuits and Systems* 7(4): 501–519.
URL: *http://doi.ieeecomputersociety.org/10.1109/43.3185*
[18] Press, W., Teukolsky, S., Vetterling, W. & Flannery, B. [2007]. *Numerical Recipes: The Art of Scientific Computing*, 3rd edn, Cambridge University Press, New York, chapter Section 10.11. Linear Programming: Interior-Point Methods.
[19] Razavi, B. [2000]. *Design of Analog CMOS Integrated Circuits*, 1st edn, McGraw-Hil.
[20] Szu, H. & Hartley, R. [1987]. Fast simulated annealing, *Physical Letters A* 122: 157–162.
[21] Vytyaz, I., Lee, D. C., Hanumolu, P. K., Moon, U.-K. & Mayaram, K. [2009]. Automated design and optimization of low-noise oscillators, *Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28(5): 609–622.

IntechOpen