Global Design of Analog Cells using Statistical Optimization Techniques

F. Medeiro, R. Rodríguez-Macías, F.V. Fernández, R. Domínguez-Castro, J.L. Huertas and A. Rodríguez-Vázquez

Instituto de Microelectrónica de Sevilla - Centro Nacional de Microelectrónica
Avda. Reina Mercedes s/n, (Edif. CICA)
E-41012, Sevilla, Spain

Analog Integrated Circuits and Signal Processing, Vol. 6, pp. 179-195, Kluwer Academics, November 1994.

© 1994 Kluwer Academic Publicher. Personal use of this material is permitted. However, permission to reprint/ republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the Kluwer Academic Publicher.

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reported without the explicit permission of the copyright holder.

Global Design of Analog Cells using Statistical Optimization Techniques

F. Medeiro, R. Rodríguez-Macías, F.V. Fernández, R. Domínguez-Castro, J.L. Huertas and A. Rodríguez-Vázquez

Dept. of Analog Circuit Design,
Centro Nacional de Microelectrónica,
Edificio CNM, Avda. Reina Mercedes sn.
41012-Sevilla, SPAIN
FAX #34 5 4624506, Phone #34 5 4239923
email angel@cnm.us.es

Abstract

We present a methodology for automated sizing of analog cells using statistical optimization in a simulation based approach. This methodology enables to design complex analog cells from scratch within reasonable CPU time. Three different specification types are covered: strong constraints on the electrical performance of the cells, weak constraints on this performance, and design objectives. A mathematical cost function is proposed and a bunch of heuristics is given to increase accuracy and reduce CPU time to minimize the cost function. A technique is also presented to yield designs with reduced variability in the performance parameters, under random variations of the transistor technological parameters. Several CMOS analog cells with complexity levels up to 48 transistors are designed for illustration. Measurements from fabricated prototypes demonstrate the suitability of the proposed methodology.

Global Design of Analog Cells using Statistical Optimization Techniques

F. Medeiro, R. Rodríguez-Macías, F.V. Fernández, R. Domínguez-Castro, J.L. Huertas and A. Rodríguez-Vázquez

Dept. of Analog Circuit Design,
Centro Nacional de Microelectrónica,
Edificio CNM, Avda. Reina Mercedes sn.
41012-Sevilla, SPAIN
FAX #34 5 4624506, Phone #34 5 4239923
email angel@cnm.us.es

1. Introduction

The design of analog VLSI building blocks, and in general the design of any integrated circuit, comprises three major steps. First, a suitable schematic must be selected. Then this schematic must be sized to comply required *performance* specifications on gain, bandwidth, slew-rate, etc., as well as to meet design *objectives* regarding area, power consumption, etc. Finally, a layout must be generated for the sized schematics. Of these three major steps, this paper focuses on the problem of analog sizing.

Analog sizing is a very complicated, time-consuming task whose automation has drawn strong attention in recent years, where several tools and methodologies have evolved [1]-[8]. Two basic reasons lie behind these developments: a) market pressure to reduce the design cost of the analog components of modern analog-digital ASICs and b) the need for custom analog design to be available to ASIC system designers.

Most previously reported approaches for automated analog cell design are *closed* systems covering only a limited number (though not necessarily small, see for instance [1]) of schematics. Some tools work on a *flat* schematic library where topologies are defined at the device-level [1], [6], [8], [9]. In others [3], [5], [7], [10] architectures are defined at the conceptual level as a connection of sub-blocks (differential pairs, current mirrors, etc.), each of which can be expanded *hierarchically* down to the device-level. Tools also differ among themselves depending on the sizing strategy used. In some approaches, the sizing process is reduced to a *constrained optimization* problem [6],[8]; in others, sizing is performed by following specific *design plans* for each topology, previously developed by expert designers and stored in the tool database [1], [3], [5], [7], [10].

Closed sizing systems are all *equation-based*; that is, the knowledge about the available topologies is provided as *analytical* design equations. The associated design equations for new topologies must be generated -- a task for only real analog design experts to tackle. Another

drawback relating to closed systems is that they do not allow the exploration of topology enhancements as conceived by designers with some expertise.

Some of the drawbacks of closed systems are overcome by the approaches in [9],[11], which are also equation based. The distinctive feature is that some of the design equations for new topologies are automatically generated via auxiliary *symbolic* analysis tools [9],[12]. Expert concourse is not further required to that end. Unfortunately, symbolic analysis tools provide equations for neither DC nor large signal transient characteristics, whose associated design equations must still be manually provided. Hence, the methodology is only partially open. Furthermore, the level of complexity for AC automatic modeling is limited by the capabilities of symbolic analysis tools (currently, about 15 MOS transistors using high-frequency MOST models and workstation standard configurations). Consequently, this approach is not the most suitable for the automated sizing of complex analog building blocks (for instance, fully-differential opamps), or for applications where large signal specifications play a major role, for instance, oversampled modulators for high resolution A/D converters [13].

Whether closed or open, equation-based systems have a common drawback in that sizing is carried out using simplified analytical descriptions of the blocks. Hence, manual fine-tuning using an electrical simulator and detailed MOS transistor models may be necessary once rough automated sizing is completed. This drawback is overcome in the so-called *simulation-based* systems [14], which also reduce sizing to a constrained optimization problem, and aim to solve it by following an iterative procedure built around an electrical simulator. No design equations are required in these approaches; the design parameters are updated at each iteration based on the results provided by simulations with detailed transistor models. Thus, they are intrinsically *open*. A representative example of this methodology is DELIGHT.SPICE [14] where DELIGHT (a general algorithmic optimization tool) and SPICE are combined. Also, advanced electrical simulators, like HSPICE [15], incorporate optimization routines. However, the optimization routines in both tools search for a *local* solution, and consequently are typically used to redesign cells whose performance specifications are close to the design goals (for instance, technology updating of a cell library), but are inappropriate to size analog cells from scratch. This is a real challenge in analog design automation and requires the development of other techniques.

This paper presents a simulation based approach for *global* sizing of *arbitrary* topology analog cells using *statistical* optimization. We demonstrate that by combining proper cost function formulation and innovative optimization heuristics complex cells are designed starting from arbitrary initial points, within reasonable CPU times and with no designer interaction required -- a very appealing feature for ASIC applications. We present results obtained for two fully-differential CMOS opamps, a comparator and an analog output buffer, which were sized using the proposed methodology, fabricated in different CMOS technologies, and whose performance was corroborated from actual silicon prototypes. The proposed technique is also

extended to design for low variability incorporating mismatching information in the design procedure. This is illustrated in the design of a CMOS folded-cascode operational amplifier.

2. Some Generalities on Optimization-Based Sizing

Analog sizing is a constructive procedure to map cell *specifications* into design *parameter* values. Design specifications are given a broad meaning here which includes *constraints* on the electrical performance parameters of the cell as well as design *objectives*. Let us consider for illustration purposes the output buffer of Fig.1, one of the examples covered in this paper. A possible specification set for this circuit could include constraints on its DC gain (A_o > target), input capacitance (C_{in} < target), 3-dB frequency (f_{3dB} > target), and output voltage range (target < OS < target), in addition to the design objective of minimum possible power consumption. With regards to the design parameters, these include transistor dimensions and passive component values.

In a generic circuit, the design parameters can be viewed as components of a vector $\mathbf{x}^T = \{x_1, x_2, \ldots x_N\}$ defining a multidimensional design space. Thus, performance parameters and the features involved in design objectives are given as functions of $\mathbf{x}$; referring again to the example of Fig.1: $A_o(\mathbf{x})$, $C_{in}(\mathbf{x})$, $f_{3dB}(\mathbf{x})$, $OS(\mathbf{x})$, and $Power(\mathbf{x})$. Then the problem of sizing is formulated as a constrained optimization problem; in particular, for the case of the buffer of Fig.1,

$$\begin{aligned} & minimize \;\; Power(\mathbf{x}) \\ & A_o(\mathbf{x}) > target \\ subjected \;\; to \quad & C_{in}(\mathbf{x}) < target \\ & f_{3dB}(\mathbf{x}) > target \\ & target < OS(\mathbf{x}) < target \end{aligned} \tag{1}$$

Unfortunately, even for elementary analog cells like that shown in Fig.1, the analytical solution

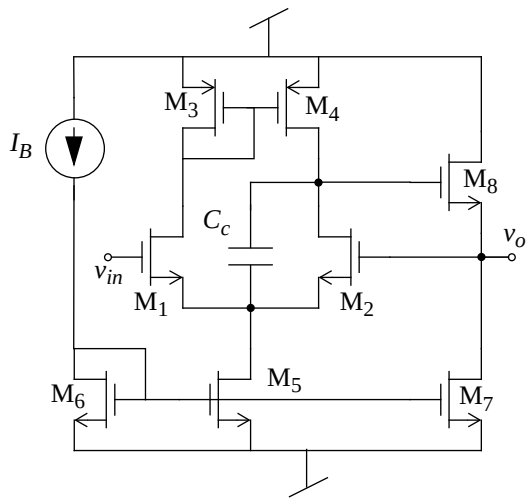


Figure 1: A CMOS output buffer.

to the sizing problem is not possible due, among other factors, to the following:

- Design equations, i.e., functional relationships among performance parameters and design objectives on one hand, and design parameters on the other, are very difficult to obtain accurately.
- These relationships are typically highly nonlinear and, consequently, unsolvable analytically. A further complication arises due to the large dimensions of the design and the specification spaces.
- The need to minimize some functions forces the calculation of first and second derivatives and hence, introduces additional complications to the analytical solution process.

Due to these difficulties, analog circuits are most conveniently sized by using an *iterative*, dynamic process. This concept is illustrated in Fig.2: starting from an initial design parameter estimate, $\mathbf{x}_0$, a discrete sequence of movements (represented generically as $\Delta\mathbf{x}_n$) is performed

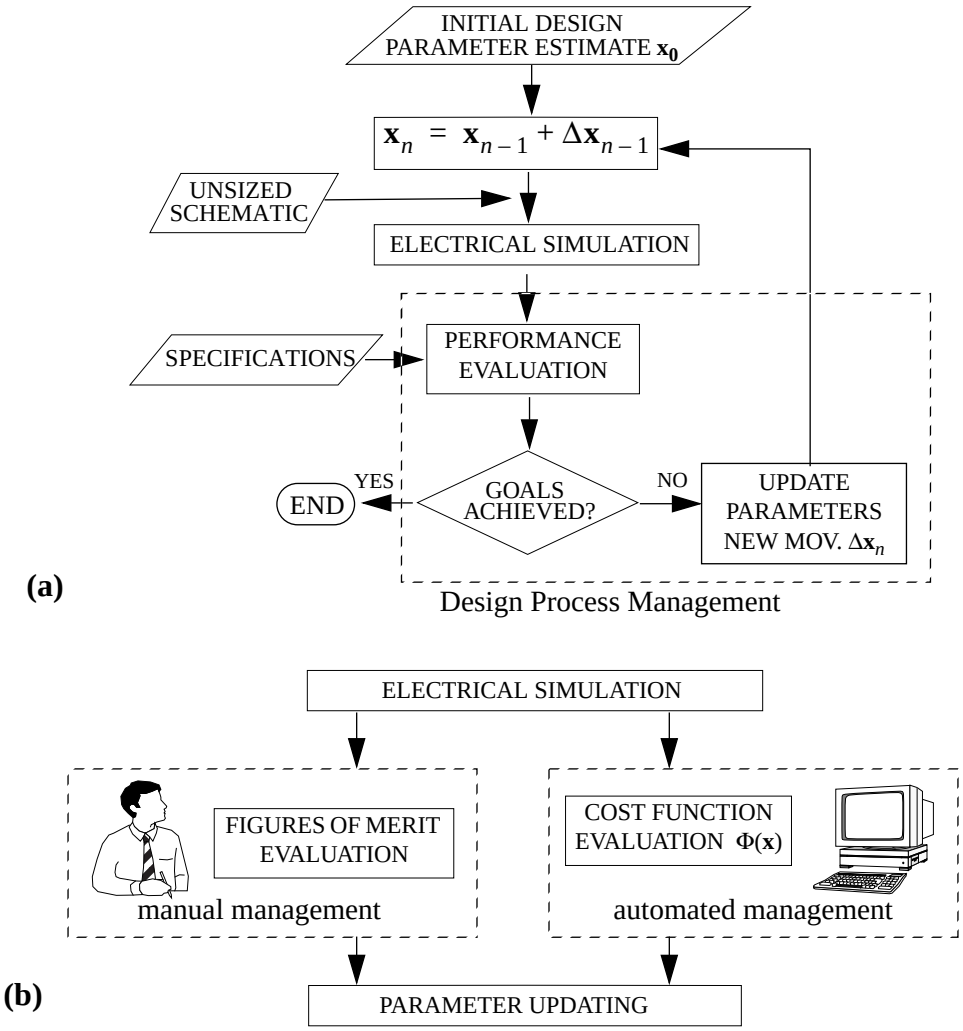


Figure 2: Iterative analog cell sizing: (a) General concept. (b) Manual and automated design updating management.

through the design parameter space until an equilibrium solution point $\mathbf{x}^*$ is found.

A key component of this iterative loop is process *management*: the calculation of the direction and magnitude of the movement $\Delta\mathbf{x}_n$ to be made at each iteration. In *manual* design, $\Delta\mathbf{x}_n$ is chosen by the designer based on his/her knowledge of the circuit structure being sized -- a difficult and time-consuming task even for experienced analog designers. In *automated* design, the selection of $\Delta\mathbf{x}_n$ must be performed by the computer based on the evaluation of some critical circuit performance indicators. A convenient approach to do this is to recast the problem formulation as a *cost* function $\Phi(\mathbf{x})$ which quantifies the degree of achievement of the design goals and their relation to the design parameters. Thus, the parameter updating to be done for the subsequent iteration $\Delta\mathbf{x}_n$ is selected at each iteration using functional analysis data of $\Phi(\mathbf{x})$. This approach also provides simple and accurate criteria to finish the sizing process at points where the cost function is either maximized or minimized.

In the simplest case, $\Delta\mathbf{x}_n$ is calculated by using pieces of information calculated only at $\mathbf{x}_n$. However, as demonstrated in this paper, the use of additional information from previous points, at time instances n–1, n–2, etc., may produce more robust solutions of the sizing problem, in the sense of yielding cells whose specifications have lower *variability* when statistical variations of the technological parameters are taken into account. In this more general case, the updating process is described as a high-order nonlinear discrete-time system,

$$\Delta\mathbf{x}_n = \mathbf{S}[\Phi(\mathbf{x}), \mathbf{x}_n, \mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \ldots, \mathbf{x}_{n-M}] \tag{2}$$

As stated in the introduction, we will assume that performance evaluations in Fig.2 (equivalently, the calculation of performance specification values and the values of the features involved in the design equations as functions of $\mathbf{x}$) are made using electrical simulation and detailed transistor models to guarantee accuracy of the sizing process. Many different alternative implementations of Fig.2 are possible depending on: **a)** formulation of the cost function itself, **b)** the updating procedure. Two major alternatives can be roughly identified, depending of the functional structure of $\mathbf{S}[\bullet]$ in (2):

- Deterministic, *incremental* techniques where $\Delta\mathbf{x}_n$ calculation uses information about the derivatives of the cost function. This is an important drawback since analytical expressions for the cost function and its derivatives as functions of the design parameters are not commonly available, so that the derivatives must be calculated by numerical interpolation. Another major drawback is that only $\Delta\mathbf{x}_n$ values which lower the cost function are considered. Hence, the optimization process is easily trapped in local minima, rendering it very suitable only for fine adjustment of the design.
- *Statistical* techniques, where $\Delta\mathbf{x}_n$ is calculated at random and hence, requires no information about the cost function derivatives.

Parameter updating in deterministic techniques is done only in the direction which lowers

the cost function. This makes them very sensitive to the starting point and hence, inadequate for global circuit sizing. This is overcome using statistical optimization techniques where movements in the design space are done heuristically, following statistical optimization principles [16]. The price to pay for an independent initial point is a larger number of iterations and hence, longer CPU times. However, as shown here, proper formulation of the cost function, the movement generator, and the cooling schedule, adapted to the nature of analog synthesis, palliates the high computational cost and thus provides a convenient methodology for global design of analog cells.

3. Cost Function Formulation

A first step towards devising a tool for automated sizing of analog cells using statistical optimization is to formalize the setting of performance specifications. In a more general case, three different specification classes must be considered:

- **Strong restrictions**: These are specifications whose fulfillment is considered essential by the designer; for instance, the phase margin of an opamp must be larger than 0 *(PM* > 0) for stability [17]. No relaxation of the specified value is allowed. Hence, if any setting of the design parameters (equivalently, any point of the design parameter space) does not satisfy one strong restriction, it must be rejected immediately.
- **Weak restrictions**: These are the typical performance specifications required of analog building blocks, i.e. A_o > 80dB. Unlike strong restrictions, weak restrictions allow some relaxation of the target parameters, making such circuit sizings which do not meet such specifications acceptable.
- **Design objectives**: Stated as the minimization (maximization reduces to this case by either changing the sign or using the inverse of the function to maximize) of some performance features,

$$minimize \qquad y_{\Psi i}(\mathbf{x}) \qquad 1 \le i \le P \tag{3}$$

for instance, minimize $-GB$ of an opamp (equivalently, maximize GB), where GB denotes the gain-bandwidth product; or minimize the occupied area of the circuit.

Mathematically, the fulfillment of these specifications can be formulated as a multi-objective constrained optimization problem,

$$\begin{aligned} &minimize \qquad y_{\Psi i}(\mathbf{x}) \qquad , 1 \le i \le P \\ &subjected \quad to \quad \begin{cases} y_{sj}(\mathbf{x}) \ge Y_{sj} \quad or \quad y_{sj}(\mathbf{x}) \le Y_{sj} & , 1 \le j \le Q \\ y_{wk}(\mathbf{x}) \ge Y_{wk} \quad or \quad y_{wk}(\mathbf{x}) \le Y_{wk} & , 1 \le k \le R \end{cases} \end{aligned} \tag{4}$$

where $y_{\Psi i}$ denotes the value of the *i-th* design objective; y_{sj} and y_{wk} denote values of the circuit

specifications (subscripts s and w denote strong and weak specifications respectively); and Y_{sj} and Y_{wk} are the corresponding targets (for instance, $A_o \geq 80$dB, settling time $\leq 0.1\mu$s).

The cost function is defined in the *minimax* sense as follows,

$$minimize \qquad \Phi(\mathbf{x}) = max\{F_\Psi(y_{\Psi i}), F_{sj}(y_{sj}), F_{wk}(y_{wk})\} \tag{5}$$

where the *partial* cost functions $F_\Psi(\bullet)$, $F_{sj}(\bullet)$, and $F_{wk}(\bullet)$ are defined as,

$$\begin{aligned} F_\Psi(y_{\Psi i}) &= -\sum_i w_i \log(|y_{\Psi i}|) \qquad , F_{sj}(y_{sj}) = K_{sj}(y_{sj}, Y_{sj}) \\ F_{wk}(y_{wk}) &= -K_{wk}(y_{wk}, Y_{wk}) \log\left(\frac{y_{wk}}{Y_{wk}}\right) \end{aligned} \tag{6}$$

where w_i (called weight parameters for the design objectives) is a positive (alternatively negative) real number if $y_{\Psi i}$ is positive (alternatively negative), and for $K_{sj}(\bullet)$ and $K_{wk}(\bullet)$ we have,

$$\begin{aligned} K_{sj}(y_{sj}, Y_{sj}) &= \begin{cases} -\infty & , if \ \ strong \ \ restriction \ \ holds \\ \infty & , otherwise \end{cases} \\ K_{wk}(y_{wk}, Y_{wk}) &= \begin{cases} \infty \operatorname{sgn}(k_k) & , if \ \ weak \ \ restriction \ \ holds \\ k_k & , otherwise \end{cases} \end{aligned} \tag{7}$$

where k_k (weight parameters assigned to weak restrictions) is a positive (alternatively negative) real number if the weak specification is of $\geq$ (alternatively $\leq$) type. Weight parameters are used to give priority to the associated design objectives and weak specifications. As shown in the cost function formulation, only relative magnitude of the weight parameters of the same type makes sense. In (7) weak specifications are assumed positive. Sign criteria is reversed for negative specifications.

Strong restrictions are checked first at each iteration. If any of them are not met, the corresponding movement must be rejected. Otherwise, weak restrictions are examined. Weak restrictions have priority over design objectives. If some weak restriction is not fulfilled, the cost function is built only with their contribution. Hence, if no circuit sizing is able to cover all weak specifications, the optimization process will provide results as close as possible. Once all of them are met, the design objectives are evaluated and their influence in the cost function guides their maximization or minimization.

4. Parameter Updating and Process Management

Fig.3 shows a block diagram illustrating the operation flow in the proposed methodology. The updating vector, $\Delta\mathbf{x}_n$, is *randomly* generated at each iteration. The value of the cost function is calculated at the new parameter space point and compared to the previous one. The new point is accepted if the cost function has a lower value. Unlike deterministic techniques, it may also be accepted if the cost function increases, according to a *probability* function,

$$P = P_o e^{-\frac{\Delta\Phi}{T}} \tag{8}$$

depending on a *control* parameter, *T*. The random character of movements and the statistical acceptance of those which increase the cost function enable escaping from local minima and hence, wide exploration of the design space. This probability of acceptance changes during the optimization process, being high at the beginning (for large *T*) and decreasing as the system *cools (*decreasing *T)*. This is the general concept lying behind *simulated annealing* optimization techniques -- a process whose name is justified by its analogies to the physical annealing in solids [16]. The tool proposed herein incorporates new heuristics relating to both parameter updating and the cooling schedule itself, as explained below.

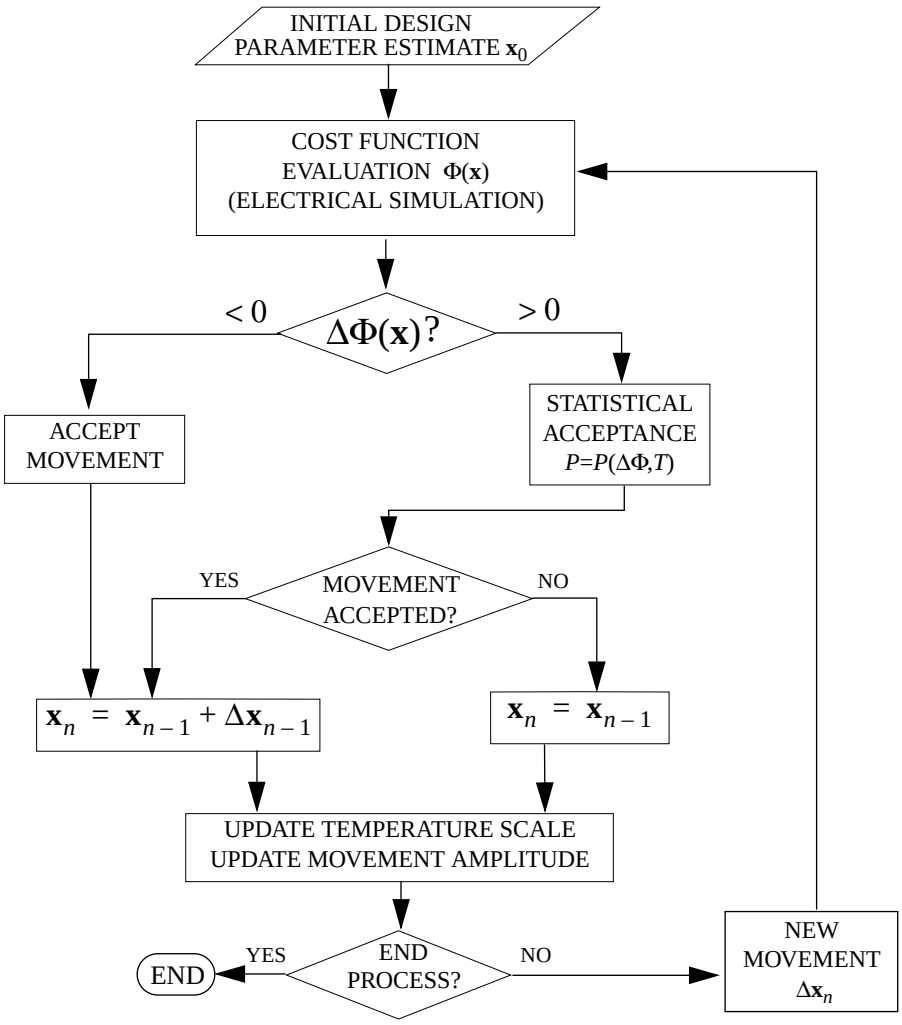


Figure 3: Operation flow in the proposed methodology.

4.1. Cooling Schedule

Cooling schedule refers to the strategy used to modify the temperature while the process evolves. Unlike classical simulated annealing algorithms [16], where *T* in (8) decreases mono-

tonically during the process, our tool uses a composed temperature parameter,

$$T = \alpha(\mathbf{x})T_o(n) \tag{9}$$

where n denotes the iteration count, $T_o(n)$ (the *normalized* temperature) is a function of n, and $\alpha(\mathbf{x})$ (the temperature *scale*) is a function of the position in the design parameter space. Our tool incorporates heuristics to choose T_o and α for increased convergence speed, namely:

- Non-monotonic and adaptive normalized temperature.
- Use of a nonlinear scale, with different expressions for different regions of the design parameter space.

4.1.1. Normalized Temperature. Instead of a conventional slow monotonically decreasing temperature [18], a sequence of fast coolings and re-heatings is used. In circuits with not very demanding specifications, this enables to obtain feasible designs for low iteration counts. Also, for those cases where demanding specifications are asked for, we have found that this strategy reduces iteration count by, on the average, a factor of 6. Two different evolutionary laws for the normalized temperature are incorporated in the tool: *exponential* decreasing, and linear decreasing. For illustration purposes Fig.4a shows an exponential schedule with 8 re-heatings. Initial and final temperatures, number of coolings, decreasing law and rate, etc. are completely controlled by the user. An alternative cooling schedule makes T_o to change as a function of the percentage of accepted movements,

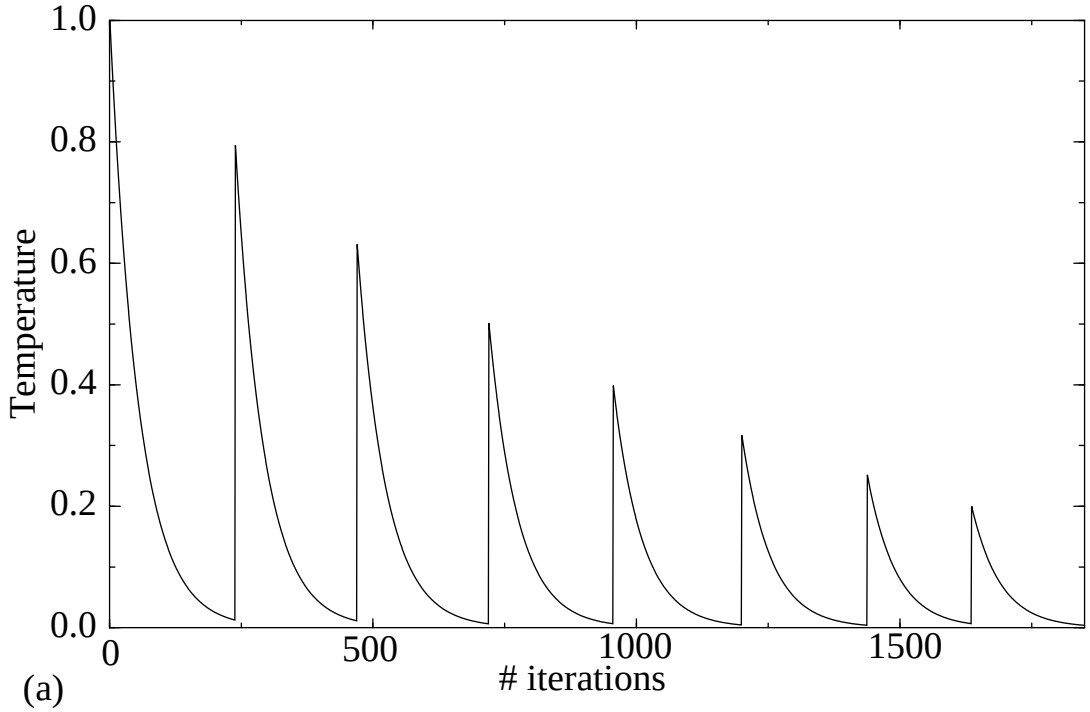
$$T_o(n) = T_o(n-1) + \beta\left(1 - \frac{\rho}{\rho_s(n)}\right) \tag{10}$$

where ρ is calculated as,

$$\rho = \frac{number\ \ of\ \ accepted\ \ movements}{number\ \ of\ \ movements} \tag{11}$$

during the last M iterations, where M is an heuristic variable whose typical value is around 25; β in (10) controls the rate of temperature chage and has a typical value around 0.1; and $\rho_s(n)$ is a prescribed acceptance ratio, which can be fixed or vary with some given law. This schedule provides very good results for practical circuits, rendering the outcome of the optimization process somewhat independent of the specified values of the initial and final temperature. Fig.4b illustrates this type of cooling schedule.

4.1.2. Temperature Scale. As (9) shows, the temperature scale parameter is a function of the position in the design parameter space. More specifically, the scale depends on which region of the parameter space is reached after each movement. This is so done to compensate the large differences that may eventually appear in the increments of the cost function in the different regions. Thus, no temperature definition is used for those regions where strong restrictions do



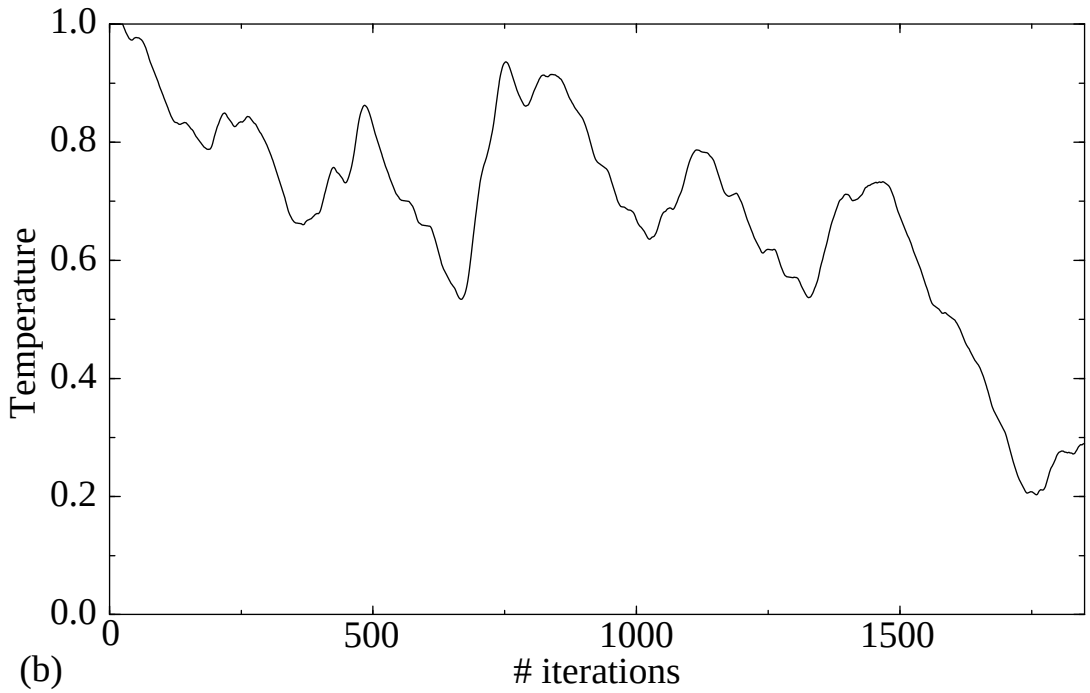


Figure 4: Cooling schedules: (a) Exponential decreasing with re-heatings. (b) Adaptive temperature with given acceptance ratio.

not hold, due to the fact that any design entering this region is automatically rejected. On the

other hand, in regions where some weak specifications are violated, temperature is given as,

$$T = T_o|k_{max}| \Rightarrow \alpha(\mathbf{x}) = k_{max} \tag{12}$$

where k_{max} is the weight associated to the maximum among the $F_w(\bullet)$'s in (6), and T_o is the normalized temperature at the current iteration. Finally, if both strong and weak restrictions hold, temperature is given as,

$$T = T_o\sum|w_i| \Rightarrow \alpha(\mathbf{x}) = \sum|w_i| \tag{13}$$

where w_i is the weight associated to the *i-th* design objective.

4.2. Parameter Updating

Concerning the updating of design parameters three kinds of heuristics have been adopted:

- Changes in the amplitude of the movement $\Delta\mathbf{x}_n$ as a function of the temperature. In particular, at high *T*, large amplitude movements are allowed as they are likely to be accepted and favor wide exploration of the design parameter space. On the contrary, at low *T*, acceptance probability decreases and, hence, only small movements are performed (equivalent to fine-tuning the design).
- The possibility of defining logarithmic scales for independent variables. This has been done because many design parameters, i.e., transistor sizes, bias currents, etc., may vary over several decades. For instance, a change of 2μA in a bias current does not have the same significance if the previous bias current values is 5μA as if it is 100μA; hence, linear movement of this variable would underexplore the low bias current range -- a drawback which is overcome by using logarithmic scales.
- Discretization of the design parameter space. Many design parameters are already discrete in nature, i.e. in many microelectronic technologies transistor dimensions can only vary over integer multiples of the technology grid. Our discretization consists in making discrete those variables which are continuous, and define a larger size grid for those variables which are already discrete in nature. Then, the parameter space can be viewed as a collection of *hypercubes*. Only movements over vertices of this multidimensional grid are allowed, being marked when they are visited. Thus, if during the optimization process one vertex is re-visited the corresponding simulation need not be performed. Hence, an important number of simulations is saved. When this optimization process ends, a local optimization is started inside a multidimensional cube around the optimum vertex for fine tuning of the design. In this local optimization, design variables recover their continuous nature or their original grid size.

Together with these heuristics, large efficiency enhancements are also achieved by proper

control of the DC electrical simulator routines. For this purpose a dynamic, adaptive, DC initialization schedule is implemented which uses operating point information of previous iterations to increase convergence speed of the simulator. This yields significant CPU time, especially at low temperatures.

4.3. Heuristics Comparison

A multiminima analytical function is used in what follows to demonstrate the advantages of the proposed heuristics. Its mathematical structure for a *N*-dimensional case is,

$$f(x) = K \cdot min\left\{ -e^{-\xi \sum_{k=1}^{N} (x_k - d)^2} \prod_{k=1}^{N} \cos(x_k - d),\ \ -e^{-\xi \sum_{k=1}^{N} (x_k - d)^2} \prod_{k=1}^{N} \cos(x_k + d) + \gamma \right\} \quad (14)$$

where K, ξ, d and γ are constants. This function has one absolute minimum (of value $-K$) and many local minima, and exhibits the interesting feature that the number of minima increases linearly with the number of variables. This means that the complexity of the optimization process is determined exclusively by the number of variables, and not by structural changes in the cost function. Fig.5a shows this function for two independent variables. A cross-section is shown in Fig.5b.

The heuristics described in Section 4.1 and 4.2 have been tested using the test function in (14) with different number of independent variables. The test procedure consisted in the repeated execution of the different heuristics on the test function, starting from random points of the parameter space and with a fixed iteration count. For each of these executions the best achieved minimum was stored. Experimental results arising from these tests are shown in the three-dimensional plots in Fig.6. In order to get better insight into the test results, the plot of the test function is allowed to take only integer values. Hence, the minimum achieved at each test execution is represented by its closest integer value. The *X*-axis in Fig.6 represents the magnitude of the achieved minimum (its closest integer value). The *Y*-axis corresponds to the number of independent variables in the test function $f(\bullet)$, and the *Z*-axis represents the percentage of iterations that achieved that minimum. Fig.6a corresponds to a conventional cooling schedule. It had a single cooling with fixed scale in variable movements and variable Markov chain length [16]. For a function with a small number of variables most iterations provided the global minimum of the function but this percentage decreased rapidly when the number of variables was increased. Fig.6b corresponds to our improved cooling schedule with the same number of iterations. The cooling schedule used had four successive coolings and re-heatings, variable scale, and a Markov chain length equal to 1. Most iterations provided the global minimum of the function, even when the number of independent variables was increased.

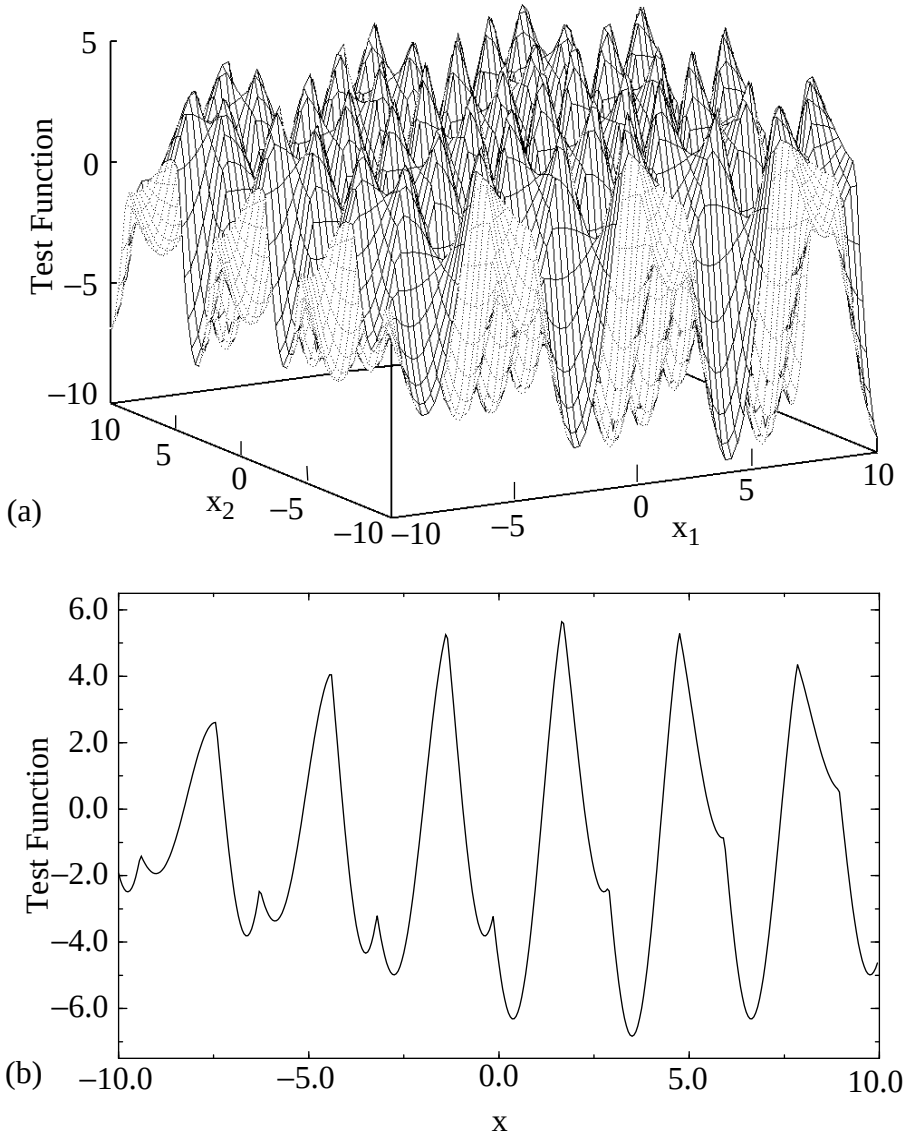
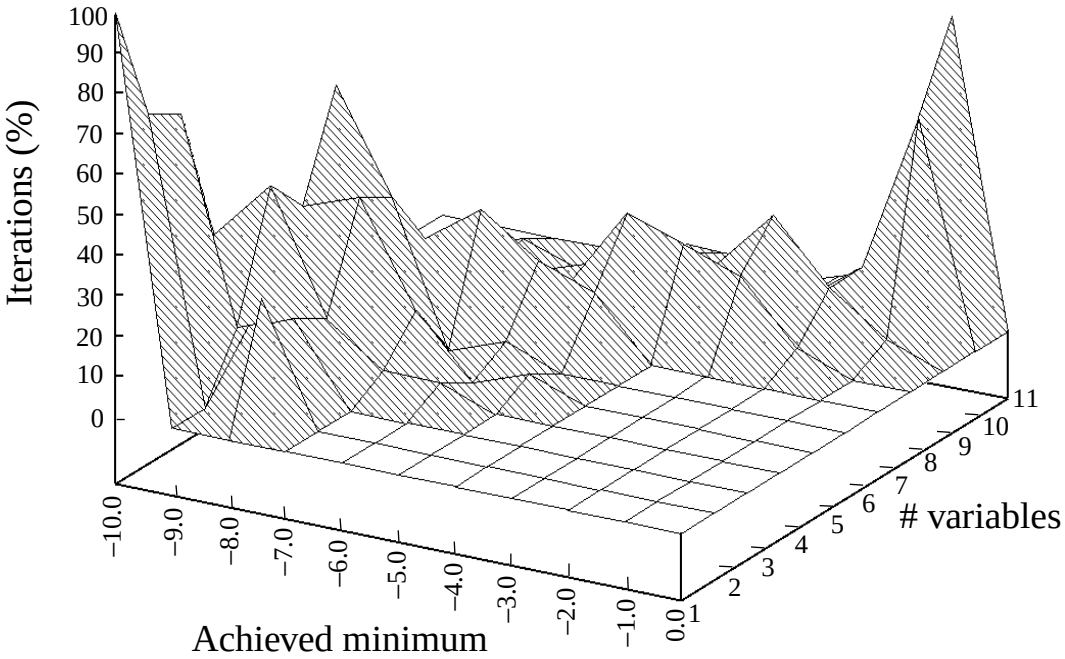


Figure 5: (a) Test function with two variables for optimization heuristics comparison. (b) Cross section.

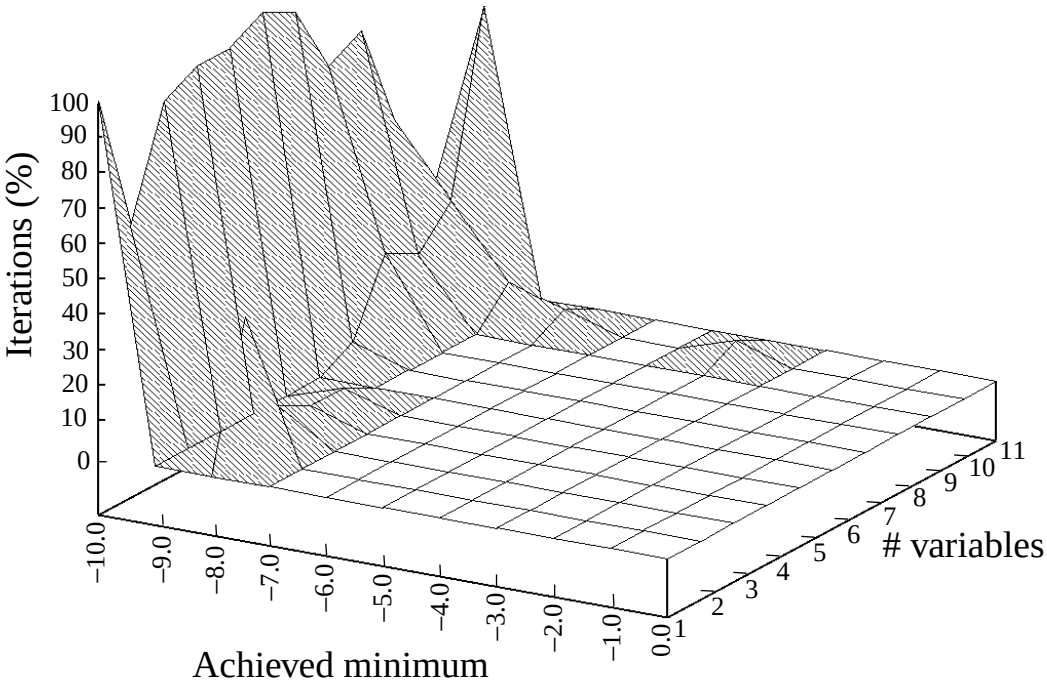
5. Extension to Low-Variability Sizing

All heuristics mentioned above assume that devices of the same type (i.e, NMOS transistor, PMOS transistors, etc.) have identical technological parameters (i.e., threshold voltages, intrinsic transconductance, etc.) and that these parameters remain constant for a given technology. However, this does not hold in practice; technological parameters are subjected to large random variations which may degrade significantly the performance of analog cells, specially when small devices are used [19],[20],[21].

Although statistical process variations can not be annulled, they can be measured, captured into models [19],[20],[21] and incorporated to the circuit design process. However, the



(a)



(b)

Figure 6: Cooling schedule heuristics comparison.

conventional approach to measure performance variations by Monte Carlo simulations is very

costly in CPU time and, consequently, not well suited to be used into an iterative optimization loop. Since dispersion of the transistor parameter values is inversely proportional to the device's area, and to the distance among nominally identical devices [21], a strategy to reduce variability of the cells is to put additional constraints on the design variables. However, this strategy drastically reduces the search space, limiting the achievement of demanding performances. The heuristics described below provides a more convenient approach that take advantage of the large amount of data generated during the statistical optimization process. It encompasses a modification of the cost function structure and a new comparison methodology, in combination to the nonmonotonic cooling schedule.

First of all, design specifications, and, hence, the cost function is made to depend not only on the vector of design parameters **x**, but also on a vector of transistor model parameters **e**. These model parameters change during the optimization process as a consequence of the dependance of their statistical variability with device area and distances between devices [21]. At each iteration, design parameters are updated according to the heuristics in Section 4.2 and device model parameters are changed according to the statistical distribution of the technological independent parameters [20]. In addition to enlarging the number of parameters, a new addend is incorporated to the cost function to evaluate the sensitivity of performance specifications to device model parameter variations. Such addend is:

$$\frac{1}{M}\sum_{n=1}^{M}\left\{\frac{\sum_{i=1}^{P}\left|\frac{y_{spec_i}(\mathbf{x}_n,\mathbf{e}_n)-y_{spec_i}(\mathbf{x}_{n-1},\mathbf{e}_{n-1})}{y_{spec_i}(\mathbf{x}_n,\mathbf{e}_n)}\right|}{\sum_{j=1}^{L}\left|e_{j,n}-e_{j,n-1}\right|}\cdot w_1\cdot w_2\right\} \tag{15}$$

which incorporates information from the last M iterations, where M corresponds typically to the number of iterations performed in one cooling. Each addend in (15) contains the ratio of the relative increase in the specifications (either weak specifications or design objectives) to the increase in the L device model parameters with respect to previous iteration. The numerator evaluates relative variations of the P performance specifications with respect to previous iteration due to variations in the design parameters **x**, and the device model parameters **e**.

The amplitude of design parameter variations decreases along each cooling. Therefore, the variability of performance specifications is evaluated with higher precision as the optimization process evolves. This fact is reflected in (15) by the weight parameter w_1, which is given by:

$$w_1 = r^{n-1} \tag{16}$$

where r in an heuristic parameter larger than 1. Hence, w_1 increases along a cooling, giving more importance in (15) to the addends corresponding to the last iterations within each cooling.

A similar weighting between different coolings is done with parameter w_2, which is given

by

$$w_2 = r'^{l-1} \tag{17}$$

where *r'* is a heuristically chosen constant parameter which must be smaller than 1 and *l* is the ordinal of the current re-heating within the cooling schedule.

The cost function at some given iteration must be compared with some previous iteration in order to accept or reject the current design parameter movement. A new comparison methodology is introduced adapted to the new cost function formulation. Each iteration in a given cooling is compared with the iteration of equal ordinal from the best of previous coolings, for acceptance or rejection, following the statistical optimization principles of Section 2. If rejected, the design point of the best cooling is adopted as new point in the current iteration and the optimization process continues. Since (15) is added to the cost function, the optimization process tends to minimize it. That implies small specification increases in the last *M* iterations and, hence, reduced performance statistical deviations.

A mathematical test function has also been used here, to prove the method's capability and as a benchmark for heuristics refinements. Its analytical struture for a *N*-dimensional case is,

$$f(\mathbf{x}) = K + K' \sum_{k=1}^{n} \left| \sin\left(\frac{2\pi x_k}{3}\right) \right| + h\left(dist^2(\mathbf{x}, \mathbf{x}_{min}) + \sum_{k=1}^{N} \left| \sin\left(\frac{2\pi x_k}{6}\right) \right| \right) \tag{18}$$

The first addend in (18), sets the mean value of *f(•)* and the second one creates variations around that mean value. Statistical deviations are simulated at the third addend by means of a random variable, *h*. The global minimum of dispersion is located in x_{min}. Sinusoidal variations set local dispersion minima. A method to reduce variance will be acceptable, if the final solution is close to x_{min}.

For a test example with eight independent variables comprised in the interval $x_n \in [-0.7, 0.7]$, a mean value K=100 and sinusoidal variations with an amplitude K'=10, the new heuristics provides a solution to a distance of 1.75 from the global minimum, where standard deviation is σ= 0.25. A conventional statistical optimization technique ends in distances around 12 from x_{min}, where standard deviation is σ= 12.

6. Practical Results

Proposed techniques have been applied to a wide variety of analog building blocks. Results are shown for the design of two fully-differential opamps, a comparator and an output buffer. Simulated results and measurement of silicon prototypes of the circuits demonstrate the feasibility of the approach.

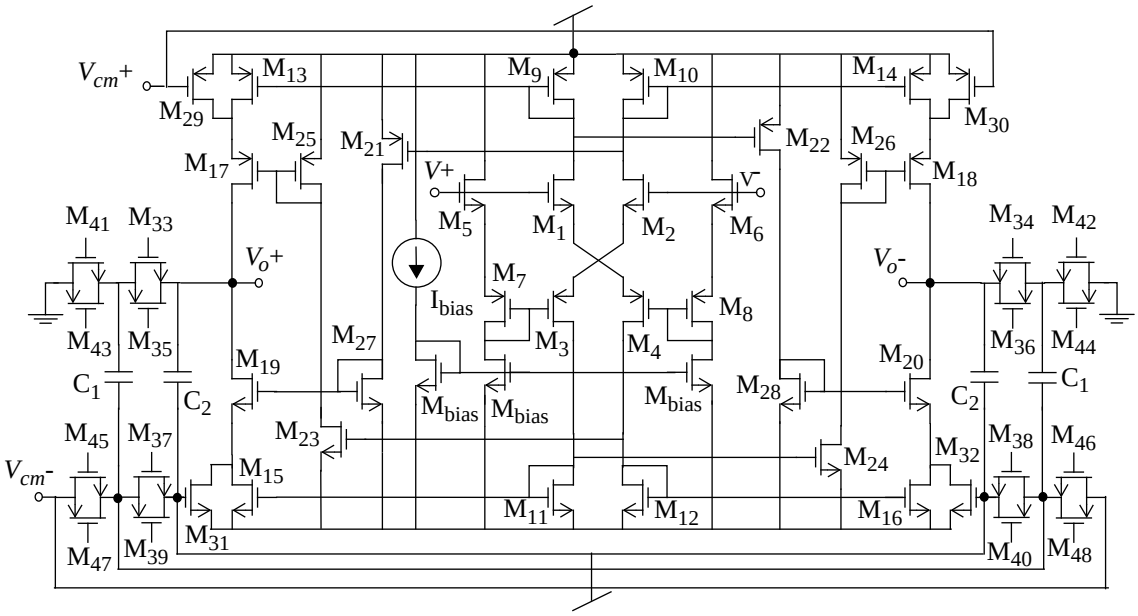


Figure 7: Fully-differential opamp.

6.1. Fully Differential Class-AB Opamp with Dynamic Biasing

Let us first consider the fully differential opamp of Fig.7 [22], intended for a 16bit@16KHz second order ∑Δ modulator. This class-AB opamp includes dynamic biasing of the output branches to obtain large output swing and high slew-rate, and uses a dynamic common-mode feedback network. These advanced circuit strategies, and the complexity of the circuit itself (it contains 48 transistors) renders its sizing a difficult task, hard to handle for system level designers. However, the herein proposed methodology was able to automatically size the circuit for the intended application after 1hour CPU time on a 100mips sparcstation, starting from scratch and with no designer interaction required. Table 2 shows the sizing obtained.

The first column in Table 1 contains the design goals, which includes a design objective on the power consumption and weak restrictions on the gain-bandwidth product (GBW), phase margin (PM), input white noise and output swing (OS). Fig.7 shows the evolution of the cost function during the optimization process. Note that the vertical axis contains two regions, separated by a dashed line. The weak region corresponds to the case where any of the weak restric-

Table 1. Simulated and measured results for the class-AB opamp.

	Specifications	**Simulated**	**Measured**	**Units**
A_0	≥ 70	74.9	74.6	dB
GBW	≥ 20	19.7	19.4	MHz
PM	≥ 60	63.3	65	o
Input white noise	≤ 50	44.7	-	nV/√Hz
OS	≥ 7	8.0	8.2	V
Offset	-	–	3.35	mV
Power	minimize	4.3	4.3	mW

tions is violated, while the objective region corresponds to the case where all weak restrictions are fulfilled; inside this region the optimization process focuses on the design objectives. Note that a *good* design (meaning one that fulfills all the weak restrictions) is obtained after 750 iterations. Simulated results corresponding to the obtained sizing are shown in the second column of Table 1.

Fig.9 is a microphotograph of a CMOS 1.2μm double poly n-well prototype of the fully differential opamp. Measured results from the silicon prototype are also shown in Table 1. The

Table 2. Sizing for the opamp of Fig.7

$M_{1,2}$	149.2 / 2.2	μm	$M_{21,22}$	48.2 / 2.2	μm
$M_{3,4}$	22.0 / 2.2	“	$M_{23,24}$	42.8 / 2.2	“
$M_{5,6}$	80.4 / 2.2	“	$M_{25,26}$	6.2 / 2.2	“
$M_{7,8}$	11.8 / 2.2	“	$M_{27,28}$	5.4 / 2.2	“
$M_{9,10}$	149.8 / 2.2	“	$M_{29,30}$	78.8 / 2.2	“
$M_{11,12}$	65 / 2.2	“	$M_{31,32}$	34.2 / 2.2	“
$M_{13,14}$	78.8 / 2.2	“	$M_{33\text{-}48}$	5.0 / 1.2	“
$M_{15,16}$	34.2 / 2.2	“	M_{bias}	378.0/ 5	“
$M_{17,18}$	121.8 / 2.2	“	$C_{1\text{-}4}$	0.4	pF
$M_{19,20}$	142.8 / 2.2	“	I_{bias}	74	μA

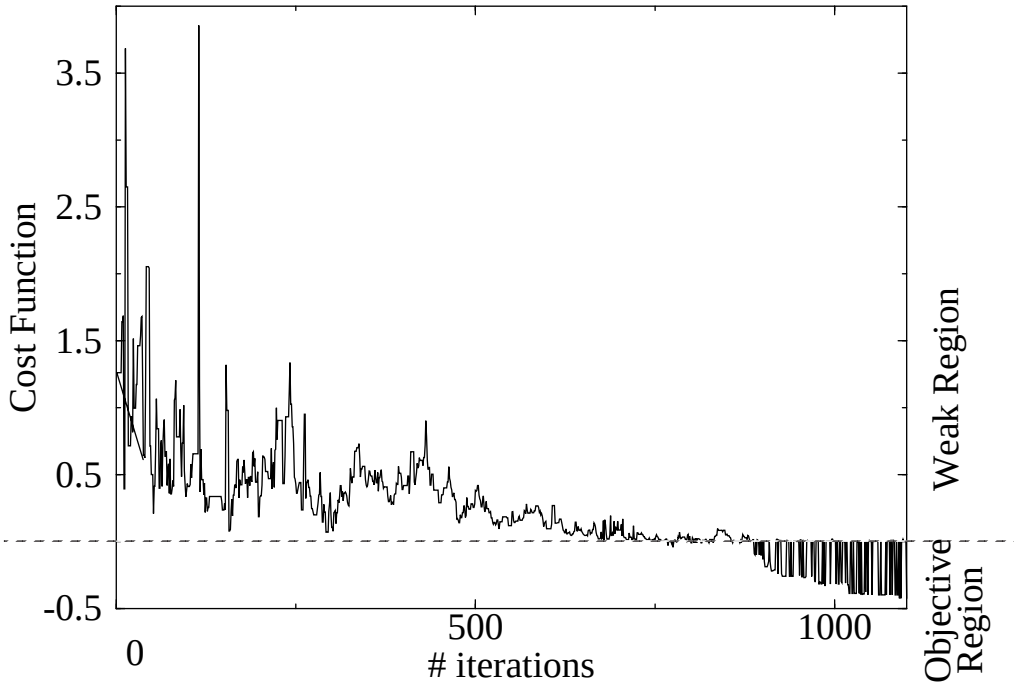


Figure 8: Cost function evolution for the optimization of Fig.7.

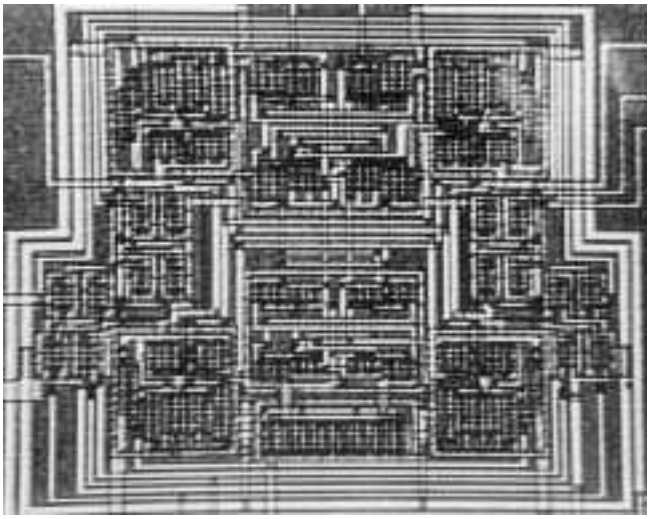


Figure 9: Microphotograph of the fully-differential opamp of Fig.7.

ΣΔ modulator CMOS prototype, which was built using this opamp, displayed a measured resolution of 15.7bit@16Khz.

6.2. Fully-Differential Folded-Cascode Opamp

As a second example, let us consider the folded-cascode fully-differential opamp of Fig.10, which displays the sizes provided by the tool. These sizes were obtained fot the specifications needed in a 17bit@40KHz fourth order ΣΔ modulator. The specifications are given in the first column of Table 3. Once again only the power consumption was a design objective. The optimization process started from scratch on a 10-dimension design space and required about 45mins. of CPU time on a 100mips sparcstation. Simulation results for the sized circuit are shown in the second column of Table 3. The opamp has been integrated in a CMOS 1.2μm double poly n-well technology. Experimental results are given in third column of Table 3. The final ΣΔ modulator prototype displayed 16.8bit@40Khz [23].

Table 3. Simulated and measured results for the folded-cascode opamp.

	Specifications	**Simulated**	**Measured**	**Units**
A_0	≥ 70	78.52	76.01	dB
GBW (1pF)	≥ 30	34.88	-	MHz
GBW(12pF,1MΩ)		4.17	4.21	MHz
PM(1pF)	≥ 60	66.28	-	^{o}
PM(12pF, 1MΩ)		87.2	86.8	^{o}
Input white noise	≤ 12	13.53	-	nV/√Hz
SR	≥ 70	74.81	70.5	V/μs
OS	≥ ±3	± 3.2	± 3.0	V
Offset	-	–	3.35	mV
Power	minimize	1.95	1.93	mW

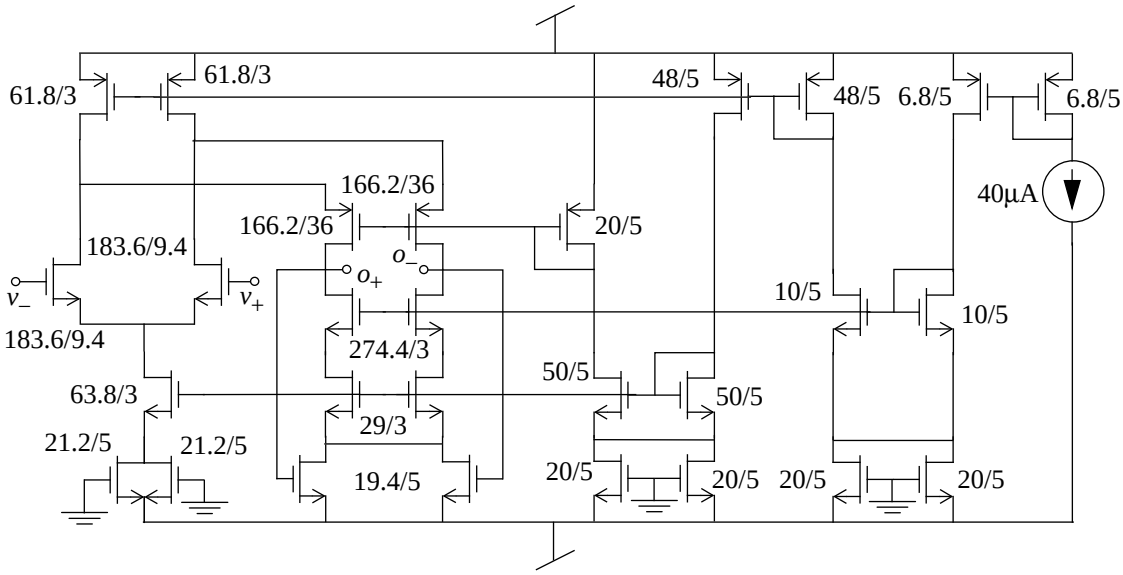


Figure 10: Fully-differential folded-cascode opamp.

6.3. Regenerative Comparator

The comparator used in the same 17bit ΣΔ modulator was designed using the high-frequency regenerative latch of Fig.11 to meet the specifications of Table 4. The simulated and measured results of the sized schematics provided by the design tool are also shown in Table 4. As for the previous opamp, measurements correspond to a prototype built in a CMOS 1.2μm double poly n-well technology. We have analyzed the origin of the slight deviations observed in the measured resolution time and have found that they can be fully explained by taking into account the dynamics of the measurement set-up.

Table 4. Simulated and measured results for the comparator.

	Specifications	Simulated	Measured	Units
TP_{HL}	< 20	8.0	12.0	ns
TP_{LH}	< 20	10.0	14.0	ns
Resolution	< 60	40	36.4	mV
Offset	-	77	22.5	mV

6.4. High-Frequency Analog Buffer

The analog buffer of Fig.1 was designed for very low input capacitance. The sizing obtained after 30mins CPU time is shown in Table 5. Table 6 shows the specifications, where the DC gain (A_0), output range (OS), and power consumption are design objectives; the 3-dB frequency is a weak restrictions, and the input capacitance is included as a constrained design objective. As shown in the third column of Table 6 the tool was able to obtain a solution with input capacitance as low as 0.07pF and f_{3dB} of 34.35Mhz.

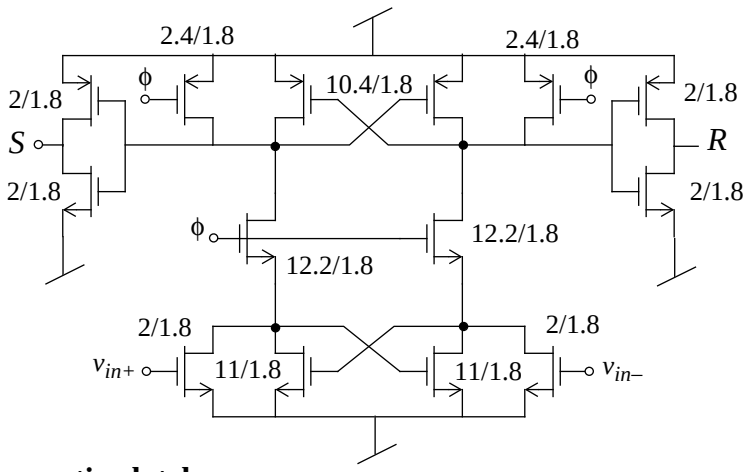


Figure 11: Regenerative latch.

Table 5. Transistor sizes for the analog buffer of Fig.1.

$M_{1,2}$	48/2.2	$M_{5,6}$	20.8/2.2	M_8	403.2/2.2	μm
$M_{3,4}$	167.2/2.2	M_7	148.8/3	C_c	4.7	pF

Table 6. Simulated results for the analog buffer (Output load 10pF@1MΩ).

Specifications	**Simulated**	**Units**
f_{-3dB} > 30Mhz	34.35	MHz
minimize C_{in}, with C_{in} < 0.1pF	0.07	pF
maximize A_0	–0.169	dB
maximize *OS*	0.6 <-> –2.2	V
minimize Power	3.726	mW

6.5. An Example of Low Variability Sizing

The reduced variability technique has been applied to practical topologies with good results. Table 7 gives the results of the application to the folded-cascode opamp of Fig.10. Electrical parameters were correlated according to [16], and their variations are proportional to transistor area and the distances between them [21]. Comparative Monte Carlo analyses are shown for the design obtained with the statistical optimization technique described in Section 3 and 4, and that described in this Section. In particular, we have focused on those specifications which are more sensitive to technological variations: offset, DC gain, common-mode rejection ratio, and power supply rejection ratio. Probability distributions resulting from Monte Carlo analysis for the latter two are very asymmetric. Hence, it is more interesting to show their possible minimum value in said probability distribution.

Experimental results with the memory-less technique of Section 3 and 4 are seen to differ with the results shown in Table 3 for the same example. This is a due to the change in techno-

Table 7. Simulation results for the folded-cascode opamp.

	Memory-less technique				Reduced variance technique			
Specs	nominal	mean	variance	min. value	mean	variance	min. value	Units
SR > 70	80.8				109.8			V/µs
offset	-	2.0	1.7		1.5	1.0		mV
power	1.73				1.1			mW
DC gain > 70	80.9	71.4	13.5		72	7		dB
GB > 30	35				31			MHz
PM > 60	65				62.9			o
noise < 12	13.4				11			n
OS > 3	3.89				4.1			V
CMRR		75.12	-	45.3	82.5	-	49	dB
PSRR		150.0	-	94.4	165.0	-	105.5	dB

logical parameters. A technology with available data about electrical parameter correlations was necessary to apply the reduced variance technique. Hence, it was reasonable to compare the results with the memory-less technique using the same technological parameters.

6.6. Discussion of Results

Summarizing, previous results demonstrate the possibility to size complex analog cells in fully automatic way, starting from scratch and without designer iteraction required -- features that render the proposed methodology very appealing for system designers. As a matter of fact, resorting to the concourse of this methodology, and using it also at the functional and system levels has enabled to design full-custom ΣΔ modulators with reduced manpower in short time cycles [23].

7. References

[1] M.G.R. Degrauwe et al. “IDAC: An Interactive Design Tool for Analog CMOS Circuits”. *IEEE Journal of Solid-State Circuits*, Vol. 22, pp. 1106-1114, December 1987.

[2] C. Meixenberger, R. Henderson, L. Astier and M. Degrauwe: “Tools for Analog Design”, *Proc. Workshop on Advances in Analog Circuit Design*, pp. 357-368, Scheveningen, The Netherlands, 1992.

[3] F. El-Turky and E.E. Perry: “BLADES: An Artificial Intelligence Approach to Analog Circuits Design”. *IEEE Transactions on Computer-Aided Design*, Vol. 8, pp. 680-691, June 1989.

[4] G. Gielen and W. Sansen: “*Symbolic Analysis for Automated Design of Analog Integrated Circuits”*. Kluwer, 1991.

[5] R. Harjani, R. Rutenbar and L.R. Carley: “OASYS: A Framework for Analog Circuits Synthesis”. *IEEE Transactions on Computer-Aided Design*, Vol. 8, pp. 1247-1265, December 1989.

[6] H. Onodera et al.: “Operational-Amplifier Compilation with Performance Optimization”. *IEEE Journal of Solid-State Circuits*, Vol. 25, pp. 466-473, April 1990.

[7] B. J. Sheu, J.C. Lee and A.H. Fung: “Flexible Architecture Approach to Knowledge-Based Analogue IC Design”. *IEE Proceedings*, Vol. 137, Pt. G, pp. 266-274, August 1990.

[8] H. Young Koh, C.H. Sequin and P.R. Gray: “OPASYN: A Compiler for CMOS Operational Amplifier”. *IEEE Trans. on Computer Aided D*esign, Vol. 9, pp. 113-125, Feb. 1990.

[9] G.E. Gielen, H. Walsharts and W. Sansen: “Analog Circuits Design Optimization Based on Symbolic Simulation and Simulated Annealing”. *IEEE Journal of Solid-State Circuits*, Vol. 25, pp. 707-713, June 1990.

[10] C. Makris and C. Toumazou: “ISAID: Qualitative Reasoning and Trade-off Analysis in Analog IC Design Automation”, *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 2364-2367, 1992.

[11] J. Jongsma et al.: “An Open Design Tool for Analog Circuits”, *Proc. Int. Symp. on Circuits and Systems*, pp. 2000-2003, 1991.

[12] F. V. Fernández, A. Rodríguez-Vázquez and J. L. Huertas: “Interactive AC Modeling and Characterization of Analog Circuits via Symbolic Analysis”, *Analog Integrated Circuit and Signal Processing*, Vol.1, pp. 183-208, Kluwer, Nov. 1991.

[13] B. E. Boser and B. A. Wooley: “The Design of Sigma-Delta Modulation Analog-to-Digital Converters”. *IEEE Journal of Solid-State Circuits*, Vol. 23, pp. 1298-1308, December 1988.

[14] W. Nye et al.: “DELIGHT.SPICE: An Optimization-Based System for the Design of Integrated Circuits”. *IEEE Transactions on Computer-Aided Design*, Vol. 7, pp. 501-519, April 1988.

[15] “HSPICE User Manual”. Meta Software Inc. 1988.

[16] P.J.M. van Laarhoven and E.H.L. Aarts: *“Simulated Annealing: Theory and Applications”*, Kluwer Academic Pub., 1987.

[17] J.K. Roberge: *“Operational Amplifiers: Theory and Practice”*. John Wiley & Sons Inc., 1975.

[18] R. A. Rutenbar: “Simulated Annealing Algorithms: An Overview”. *IEEE Circuits and Devices Magazine*, Vol. 5, pp. 19-26, January 1989.

[19] S. Director, W. Maly and A. Strojwas: *“VLSI Design for Manufacturing: Yield Enhancement”*, Kluwer Academic Publishers, 1990.

[20] C. Michael and M. Ismail: “Statistical Modeling of Device Mismatch for Analog MOS Integrated Circuits”, *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 2, pp. 154-166, Feb. 1992.

[21] M. Pelgrom, A. Duinmaijer and A. Welbers: “Matching Properties of MOS Transistors”, *IEEE Journal of Solid-State Circuits*, Vol. 24, No. 5, pp. 1433-1440, Oct. 1989.

[22] C. Wang, R. Castello and P.R Gray: “A Scalable High-Performance Switched-Capacitor Filter”. *IEEE Journal of Solid-State Circuits*, Vol. 21, pp. 57-64, February 1986.

[23] F. Medeiro, B. Pérez Verdú, A. Rodríguez Vázquez y J. L. Huertas: "A Tool for Automated Design of Sigma-Delta Modulators using Statistical Optimization". *Proc. of ISCAS’93*,, pp. 1373-1376. Chicago, May, 1993.