A Statistical Optimization-Based Approach for Automated Sizing of Analog Cells

F. Medeiro, F. V. Fernández, R. Domínguez-Castro and A. Rodríguez-Vázquez

Dept. of Analog Circuit Design, Centro Nacional de Microelectrónica, Sevilla, SPAIN

Abstract

This paper presents a CAD tool for automated sizing of analog cells using statistical optimization in a simulation based approach. A nonlinear penalty-like approach is proposed to define a cost function from the performance specifications. Also, a group of heuristics is proposed to increase the probability of reaching the global minimum as well as to reduce CPU time during the optimization process. The proposed tool sizes complex analog cells starting from scratch, within reasonable CPU times (approximately 1hour for a fully differential opamp with 51 transistors), requiring no designer interaction, and using accurate transistor models to support the design choices. Tool operation and feasibility is demonstrated via experimental measurements from a working CMOS prototype of a folded-cascode amplifier.

1: Introduction

Most previously reported approaches for automated analog cell design are *closed* systems where the knowledge of the available topologies is provided as *analytical* design equations. The design equations associated to new topologies must be generated -- a task for only real analog design experts. Also, sizing is carried out using simplified analytical descriptions of the blocks and thus, manual fine-tuning using an electrical simulator and detailed MOS transistor models may be necessary once rough automated sizing is complete. These drawbacks are overcome in the so-called *simulation-based* systems [1], which reduce sizing to a constrained optimization problem and aim to solve it by following an iterative procedure built around an electrical simulator, with no design equations required. A representative example is DELIGHT.SPICE [2] where DELIGHT (a general algorithmic optimizer) and SPICE are combined. Also, advanced electrical simulators, like HSPICE [3], incorporate optimization routines. However, the optimization routines in both tools search for *local* solutions, and are consequently inappropriate to size analog cells from scratch.

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 requiring no designer interaction -- a very appealing feature for ASIC applications.

2: Cost function formulation

Three different specification classes are considered:

- **Strong restrictions**. No relaxation of the specified value is allowed. Hence, if any 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.
- **Design objectives**. Stated as the minimization/maximization of some performance features,

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

where **x** is the vector of design parameters.

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

$$\begin{aligned} &minimize \quad y_{\Psi i}(\mathbf{x}) \quad , 1 \le i \le P \\ &subjected \quad to \\ &\begin{cases} y_{sj}(\mathbf{x}) \ge Y_{sj} \quad or \quad y_{sj}(\mathbf{x}) \le Y_{sj} & , 1 \le j \le Q \\ y_{wk}(\mathbf{x}) \ge Y_{wk} \quad or \quad y_{wk}(\mathbf{x}) \le Y_{wk} & , 1 \le k \le R \end{cases} \end{aligned} \tag{2}$$

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, $A_o \ge 80$dB, settling time ≤ 0.1μs).

A cost function, which transforms the constrained optimization problem into an unconstrained one, is defined in the *minimax* sense as,

$$minimize \Phi(\mathbf{x}) = max\{F_\Psi(y_{\Psi i}), F_{sj}(y_{sj}), F_{wk}(y_{wk})\} \quad (3)$$

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

$$F_\Psi(y_{\Psi i}) = -\sum_i w_i \log(|y_{\Psi i}|) \qquad F_{sj}(y_{sj}) = K_{sj}(y_{sj}, Y_s)$$
$$F_{wk}(y_{wk}) = -K_{wk}(y_{wk}, Y_{wk}) \log\left(\frac{y_{wk}}{Y_{wk}}\right) \quad (4)$$

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,

$$K_{sj}(y_{sj}, Y_{sj}) = \begin{cases} -\infty & \text{if strong restriction holds} \\ \infty & \text{otherwise} \end{cases}$$
$$K_{wk}(y_{wk}, Y_{wk}) = \begin{cases} \infty \operatorname{sgn}(k_k) & \text{if weak rest holds} \\ k_k & \text{otherwise} \end{cases} \quad (5)$$

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 (5) weak specifications are assumed positive. Sign criteria is inverted for negative specifications.

3: Process management

Fig.1 shows a block diagram illustrating the operation flow in the proposed methodology. An updating vector, $\Delta\mathbf{x}_n$, is *randomly* generated at each iteration. Strong restrictions are then checked. If any of them is not met, the corresponding movement is rejected. Otherwise, weak restrictions are examined. Only if all of them are fulfilled, the design objectives are included in the cost function. The value of the cost function is calculated at the new point and compared to the previous one. The new point is accepted if the cost function has a lower value. Otherwise, it may also be accepted according to a *probability* function,

$$P = P_o e^{-\frac{\Delta \Phi}{T}} \quad (6)$$

depending on a *control* parameter, *T*.

3.1: Cooling schedule

Unlike classical simulated annealing algorithms [4], where *T* in (6) decreases monotonically during the process, our tool uses a composed temperature parameter,

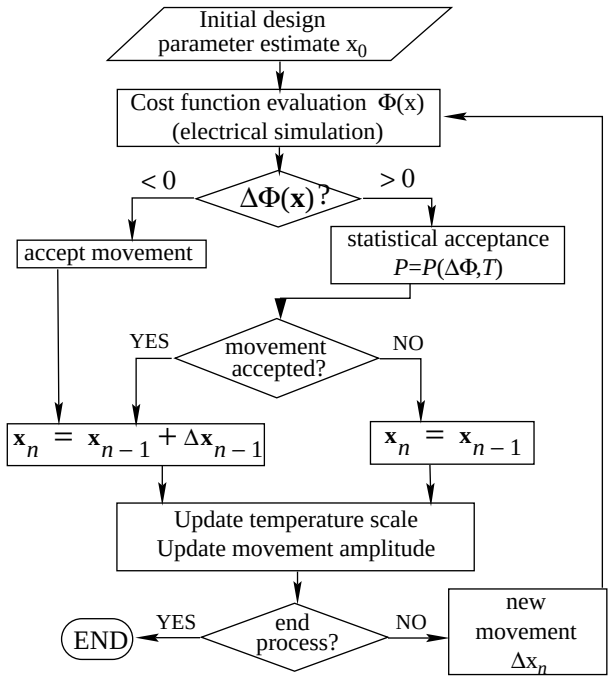

Figure 1: Block diagram of the tool

$$T = \alpha(\mathbf{x}) T_o(n) \quad (7)$$

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**. Instead of a conventional slow monotonically decreasing temperature [5], a sequence of fast coolings and reheatings is used. This enables obtaining feasible designs for low iteration counts for circuits with not very demanding specifications. In the more general case, this strategy reduces iteration count by an average factor of 6. Two different evolutionary laws for the normalized temperature are incorporated in the tool: *exponential* decreasing and linear decreasing. 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 change as a function of the percentage of accepted movements:

$$T_o(n) = T_o(n-1) + \beta\left(1 - \frac{\rho}{\rho_s(n)}\right) \quad (8)$$

where ρ is calculated as the ratio of accepted movements to the total number of movements during the last *M* iterations, where *M* is a heuristic variable whose typical value is around 25; β in (8) controls the rate of temperature change 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.

- **Nonlinear scale.** This is 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 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,

$$T = T_o|k_{max}| \Rightarrow \alpha(\mathbf{x}) = k_{max} \tag{9}$$

where k_{max} is the weight associated to the maximum among the $F_w(\bullet)$'s in (4), and T_o is the normalized temperature at the current iteration. Finally, if both strong and weak restrictions hold, temperature is given as,

$$T = T_o\sum|w_i| \Rightarrow \alpha(\mathbf{x}) = \sum|w_i| \tag{10}$$

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

3.2: Parameter updating

Three kinds of heuristics have been adopted:

- **Temperature-dependent amplitude.** 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).
- **Logarithmic scales for independent variables.** This avoids underexploring the design space of design parameters which vary over several decades, for example, bias currents.
- **Discretization of the design parameter space.** With this partitioning, 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 revisited the corresponding simulation need not be performed. Hence, an important number of simulations is avoided. When this optimization process ends, a local optimization starts within 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.

Large efficiency enhancements are also achieved by proper control of the DC electrical simulator routines. A dynamic, adaptive, DC initialization schedule is implemented which uses operating point information of previous iterations to increase convergence speed of the simulator. This significantly reduces CPU time, especially at low temperatures.

3.3: Heuristics comparison

The proposed heuristics have been tested using the function

$$f(x) = K \cdot min\{A\prod_{k=1.N}\cos(x_k - d), A\prod_{k=1.N}\cos(x_k + d) + \gamma\} \tag{11}$$
$$A = -e^{-\xi\sum_{k=1.N}(x_k - d)^2}$$

where K, ξ, d, and γ are constants. It has one absolute minimum (of value $-K$) and many local minima, whose count increases linearly with the number of variables. Thus, the complexity of the optimization process is determined exclusively by the number of variables, and not by structural changes in the cost function. Fig.2 shows this function for two independent variables.

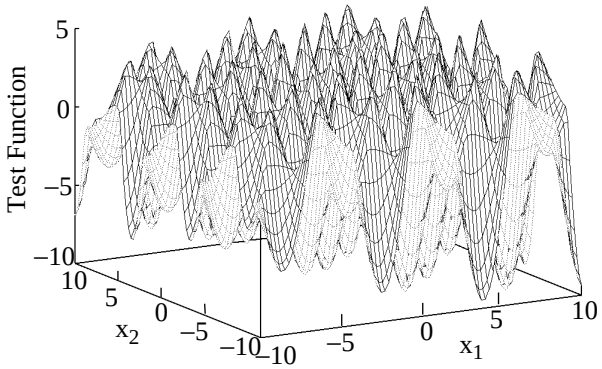

Figure 2: Test function for heuristics comparison.

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. The best achieved minimum for each of these executions was stored. Experimental results from these tests are shown in the three-dimensional plots in Fig.3. In order to get better insight into the test results, the plot of the test function is allowed to assume only integer values. Hence, the minimum achieved at each test execution is represented by its closest integer value. The *X*-axis in Fig.3 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.3a corresponds to a conventional cooling schedule. It had a single cooling with fixed scale in variable movements

and variable Markov chain length [4]. 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.3b corresponds to our improved cooling schedule with the same number of iterations. The cooling schedule used had four successive coolings and reheatings, 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.

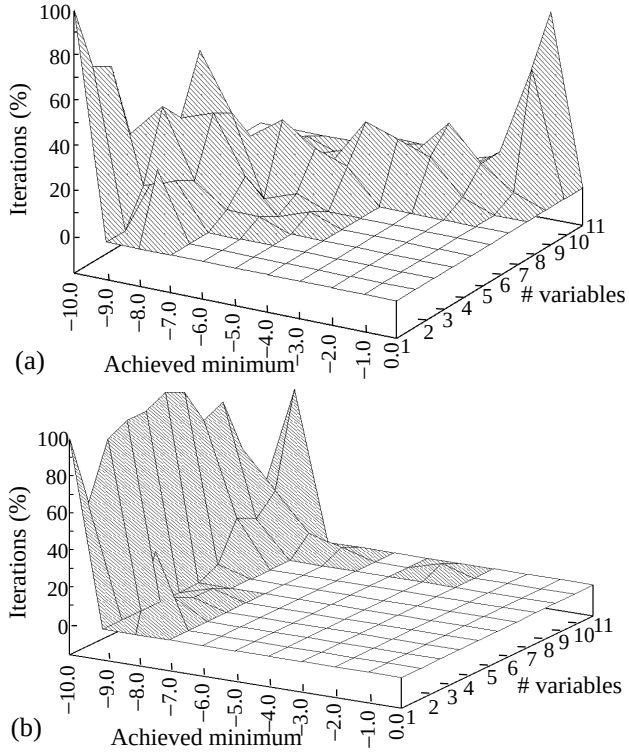

Figure 3: Cooling schedule heuristics comparison.

4: Practical results

Let us consider the folded-cascode fully-differential opamp of Fig.4, which displays the sizes provided by the tool. These sizes were obtained for the specifications needed in a 17bit@40KHz fourth order ∑Δ modulator. The specifications are given in the first column of Table 1. The power consumption was the only design objective. The optimization process started from scratch on a 10-dimension design space and required about 45min. CPU time on a 100MIPS Sparcstation. Program results for the sized circuit, corresponding to the electrical simulator output, are shown in the second column of Table 1. The opamp has been integrated in a CMOS 1.2μm double poly technology. Experimental results are given in third column of Table 1. The final ΣΔ modulator prototype displayed 16.8bit@40Khz.

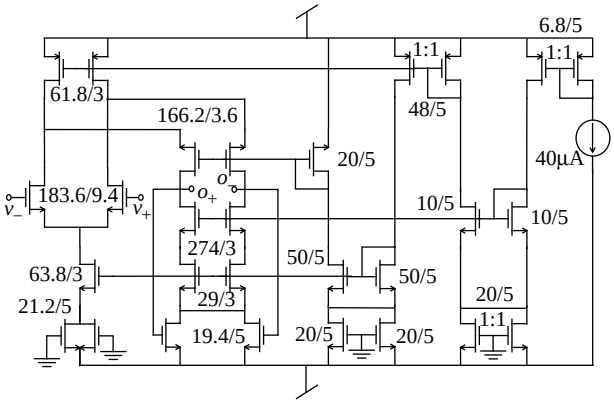

Figure 4: Fully-differential folded-cascode opamp.

These results compare advantageously to equation-based design systems. These typically spend a few seconds or minutes for the design of similar analog cells. But the effort to generate the knowledge required for new topologies varies between several weeks and 12 months. On the contrary, input file preparation in our tool requires no more than one hour of a SPICE user.

Table 1. Simulated and measured results for Fig.4.

	Specs	**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

5: References

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

[2] 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.

[3] "HSPICE User Manual". Meta Software Inc. 1988.

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

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