Generalized Posynomial Performance Modeling

Tom Eeckelaert Walter Daems* Georges Gielen Willy Sansen
Katholieke Universiteit Leuven, Department of Electrical Engineering, ESAT–MICAS
Kasteelpark Arenberg 10, B–3001 Leuven
Tom.Eeckelaert@esat.kuleuven.ac.be

Abstract

This paper presents a new method to automatically generate posynomial symbolic expressions for the performance characteristics of analog integrated circuits. The coefficient set as well as the exponent set of the posynomial expression are determined based on SPICE simulation data with device-level accuracy. We will prove that this problem corresponds to solving a non–convex optimization problem without local minima. The presented method is capable of generating posynomial performance expressions for both linear and nonlinear circuits and circuit characteristics. This approach allows to automatically generate an accurate sizing model that composes a geometric program that fully describes the analog circuit sizing problem. The automatic generation avoids the time–consuming nature of hand–crafted analytic model generation. Experimental results illustrate the capabilities and effectiveness of the presented modeling technique.

1. Introduction

Recently, it was demonstrated that the sizing of analog integrated circuits, like amplifiers, switched–capacitor filters, LC oscillators, etc., can be formulated as a geometric program [1, 2]. The circuits are characterized with symbolic equations that have to be cast in posynomial format. The advantages of a geometric program are (1) that the problem is convex and therefore has only one global optimum, (2) that this optimum is not correlated with the optimization's starting point, and (3) that infeasible sets of constraints can be identified [3]. In addition, the geometric program's optimum can be found extremely efficiently, using interior point methods [4], even for relatively large problems. The sizing is so fast (seconds) that design space explorations and process corner analysis can be performed quite easily. Of course, the key limitation of the approach is that all circuits and device characteristics have to be written in posynomial format. In [1, 2] it is shown that this is possible for most circuit characteristics, while the others have to be approximated by a posynomial model. This used to be a manual effort.

In [5, 6] it has been demonstrated that compact and interpretable expressions can be generated very efficiently and automatically by generating posynomial symbolic expressions for linear and nonlinear circuit characteristics based on the fitting of simulated data. Because only templates with fixed exponents were considered in that approach, the fitting corresponds to the minimization of a convex quadratic form. The method presented in this paper tackles the *generalized* problem in which not only the coefficients, but also the exponents are considered for parametric regression. In this case it can be shown that the function to be minimized is non–convex but has no local minima.

Although the method of [5, 6] and this new approach are no analytical techniques, but based on parametric regression, they are able to generate symbolic expressions for the performance characteristics of analog electronic circuits. Additionally, they exhibit a number of extra assets over classical symbolic analysis techniques (see also [5]):

- Linear and nonlinear performance characteristics can by analyzed. Every characteristic that can be simulated can therefore be modeled.
- The expressions that are generated are very compact.
- The expressions are formulated in terms of design variables.
- Full–complexity transistor models can be used.
- The expressions are validated over an entire subspace of the design space.
- The posynomial nature of the expressions allows the use of very effective optimization techniques.

This paper has been organized as follows. In section 2, some basic definitions are given. The expression genera-

*Walter Daems is a Postdoctoral Research Fellow of the Fund for Scientific Research – Flanders (FWO–Vlaanderen).

tion method is described in section 3. The algorithm is described in section 4. Section 5 shows some results obtained with our new approach. Finally, conclusions can be found in section 6.

2. Preliminaries

In this section, we will concisely provide definitions for (1) signomial and posynomial expressions, and (2) geometric programming.

2.1 Signomial and posynomial expressions

Let $X = (x_1, x_2, x_3, \ldots, x_n)^T$ be a vector of real, positive variables. An expression f is called *signomial* if it has the form

$$f(X) = \sum_{i=1}^{m} \left(c_i \prod_{j=1}^{n} \left(x_j^{\alpha_{ij}} \right) \right), \tag{1}$$

with $c_i \in \mathcal{R}$ and $\alpha_{ij} \in \mathcal{R}$. If we restrict all c_i to be positive ($c_i \in \mathcal{R}_+$), then the expression f is called *posynomial*.

Whereas the signomial form has better fitting properties, the posynomial form allows formulating analog circuit sizing as a geometric program [1, 2]. Notice that the posynomial constraint on the values of c_i keeps us from treating the fitting problem as a standard parametric regression problem.

2.2 Geometric programming

A (primal) geometric program in canonical form is the constrained optimization problem:

$$\begin{aligned} &\underset{X}{\text{minimize}}\, f_0(X) \\ &\text{with the constraints: } f_i(X) \leq 1, \quad i \in [1:p] \\ &\qquad\qquad\quad g_j(X) = 1, \quad j \in [1:q] \\ &\qquad\qquad\quad x_k \geq 0, \quad k \in [1:n] \end{aligned} \tag{2}$$

with all $f_i(X)$ posynomial and all $g_j(X)$ monomial. By substituting all variables x_k by $z_k = log\,(x_k)$, and taking the logarithm of the objective function f_0 and every constraint f_i, g_j, it can be seen that the transformed problem is a convex optimization problem. Because of this, it has only one global optimum. In addition, this optimum can be found very efficiently using interior point methods [4].

In view of the canonical geometric programming formulation, all performance constraints must be modeled in the normal form of (2). Therefore, in our approach, we will scale the performance variables p_l as in [6].

3 Generalized posynomial expression generation

This section provides a description of the method used to generate the posynomial performance expressions. First, a general definition of the regression problem is given. This is followed by an explanation of how the performance data samples are generated. In a third subsection, the general definition is translated in an optimization problem with objective function and a set of constraints. In a final part, the objective function is analyzed to enable the selection of an appropriate optimization algorithm to be implemented in section 4.

3.1 Problem definition

The general posynomial parametric regression problem can be defined as:

Posynomial parametric regression problem:

Given a set of performance data samples

$$\{(X_1, p_{l,1}), (X_2, p_{l,2}), \ldots, (X_a, p_{l,a})\} \tag{3}$$

and a model template of type (1), determine the coefficients $c_i \in \mathcal{R}$, and the exponents $\alpha_{ij} \in \mathcal{R}$, subject to

$$c_i \geq 0 \tag{4}$$

such that

$$\left\| [f(X_1), f(X_2), \ldots, f(X_a)]^T - [p_{l,1}, p_{l,2}, \ldots, p_{l,a}]^T \right\|_2 \tag{5}$$

is minimal. ■

This definition differs from the one in [5, 6] where the α_{ij} were assumed to be a priori fixed.
We use an Euclidean norm in (5) for two reasons:

1. It renders (5) smooth.
2. It causes a good centered spread of the sampling data around the fitted model.

3.2 Performance Sample Generation

The set of performance samples from (3) is generated using the principles of Fig. 1. Using techniques from *design of experiments* [7], a set of experiments is composed. We based our sampling scheme on *orthogonal arrays* of strength 3, for their ability to allow the unconfounded estimation of all linear and interaction effects of a second–order model [8, 9]. This scheme places the sampling points on the edge of a fitting hypercube around a center point. Usually

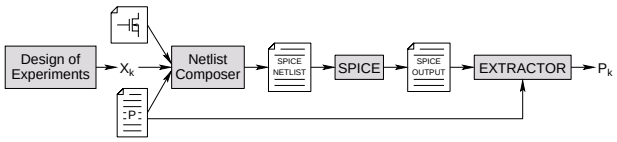

Figure 1. Performance generation outline.

the center point is provided as well. Based on the set of experimental values for the design variables and together with the specified analysis cards, a SPICE netlist is composed by a *netlist composer* (e.g., an OPD solver [10]). Next, all experiments are distributed over the available computing equipment. Afterwards all SPICE results are collected and treated by extractor scripts to determine the performance data.

3.3 Posynomial Parametric Regression

Due to the set of constraint of (4) and the fact that the exponents are part of the parameters to be optimized, this problem cannot be solved as a linear parametric regression problem. We will therefore treat (5) as an optimization problem:

$$\underset{C,A}{\text{minimize}}\ D(C,A) = \sum_{r=1}^{a}\left(\sum_{i=1}^{m} c_i \left(\prod_{j=1}^{n} x_{rj}^{\alpha_{ij}}\right) - p_r\right)^2 \tag{6}$$

$$\text{subject to } c_i \geq 0 \qquad \forall i = [1:m] \tag{7}$$

$$\text{with } C = (c_1, c_2, \ldots, c_m)^T$$

$$\text{and } A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{m1} & \ldots & \ldots & \alpha_{mn} \end{bmatrix}$$

From this, we need to develop a suitable optimization algorithm to solve the regression problem. Therefore, we will now analyze the objective function of (6) and prove that the function is not convex, but has no local minima.

3.4 Objective Function Analysis

Consider a monomial function $d = c\prod_{j=1}^{n} x_j^{\alpha_j}$. Taking into account the constraints of (7) and logarithmically scaling the design variables onto $x_j \in (0, 1]$, it is readily seen that:

$$\frac{\partial d}{\partial \alpha_l} = c\exp\left(\sum_{j=1}^{n}\alpha_j \ln x_j\right)\ln x_j \ \leq 0$$

$$\frac{\partial d}{\partial c} = \exp\left(\sum_{j=1}^{n}\alpha_j \ln x_j\right) > 0$$

Extending the monomial function to the general posynomial case by summation of different, independent monomial functions, doesn't change the sign of the partial derivatives. By subtracting the performance data p_r from the positive, monotonous descending posynomial function, it is possible the resulting function yields negative function values. Notice that the shifted function $f_{pr} = f(X_r) - p_r$ now has a domain M_+ where its values are positive and a domain M_- where its values are negative, but it still is monotonous descending.

From that observation it can be understood that the squared function is monotonous descending in M_+ and monotonous ascending in M_-. Thus, $f_{pr}^2 = (f(X_r) - p_r)^2$ has no local minima, only zero as global minimum.

The final step to get to the objective function of (6) is the summation over all performance data samples of (3). The different terms of the summation only differ from each other due to the particular values of the design variables x_{rj} and the corresponding performance data p_r. This means that the global domain, composed by the variables in C and A, is the same for all terms, but the domains M_+ and M_- are different for each term. For $D(C, A)$ to reach a minimum, its gradient should be zero:

$$\nabla D(C,A) = \sum_{r=1}^{a}\nabla f_{pr}^2 = 0. \tag{8}$$

This corresponds to a set of $(n+1)\times m$ equations, one for each partial derivative. Apart from the trivial solution where all the partial derivatives are zero, this can only hold if derivatives with opposite signs cancel out each other.

Now, as an example, observe the objective function in only 1 dimension composed with only 2 performance data samples, $D(\alpha) = f_{p1}^2 + f_{p2}^2$. In Fig. 2.a the 2 functions are drawn. In Fig. 2.b their derivatives ($\nabla f_{p1}^2, -\nabla f_{p2}^2$). It is clear that there is only 1 value of α where the derivatives are equal with opposite signs, and thus, there is only 1 value of α where $D(\alpha)$ can reach a minimum.

Because the domain of $D(\alpha)$ can also be divided in an M_+ and M_- part, the previous can be extended to the objective function from (6) observed in 1 dimension. Generalizing this to the complete set of equalities out of (8), we can conclude that $D(C, A)$ is not convex, but has no local minima. We therefore chose to develop a conjugate–gradient–based constrained optimization algorithm that takes (7) into account.

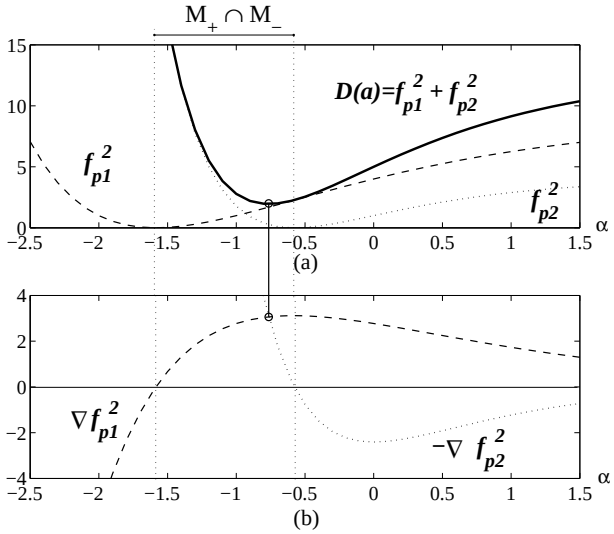

Figure 2. One dimensional objective function.

4 Algorithm Implementation

First consider the objective function of (6) without the set of constraints of (7). In a second step, we will add the constraints of (7).
The algorithm implements an iterative optimization of the model. In every step of the iteration process, the values of c_i and α_{ij} are improved to construct a better posynomial performance model than the previous one:

- first the gradient of (6) is composed out of the partial derivatives with respect to the coefficients c_i and the exponents α_{ij}:

$$\frac{\partial D}{\partial c_k} = \sum_{r=1}^{a} 2\left(f(X_r) - p_r\right) \prod_{j=1}^{n} x_{jr}^{\alpha_{kj}}$$

$$\frac{\partial D}{\partial \alpha_{kl}} = \sum_{r=1}^{a} 2\left(f(X_r) - p_r\right) c_k \ln(x_{lr}) \prod_{j=1}^{n} x_{jr}^{\alpha_{kj}}$$

- with this gradient and all the previous optimization directions, a new conjugate direction is calculated in a second step using a standard Fletcher–Reeves [11] algorithm. This new direction is conjugate to all the previous optimization directions.

- by observing the objective function in only that conjugate direction, the new coefficients and exponents can be calculated using a line–minimization along that direction.

These steps are repeated until one of the following stop criteria are met:

1. The designer can choose a certain boundary for the quality of the generated models. To assess the quality of the models a normalized *mean square error* metric is used:

$$q = \frac{\sqrt{\frac{\sum_{r=1}^{a}(f(X_r)-p_r)^2}{a}}}{\max_{r=1}^{a} p_r - \min_{r=1}^{a} p_r} \tag{9}$$

   In this, the number of performance data samples, a, is used to be able to compare the quality for different amounts of samples. The denominator relates the quality of the fit to the range of the performance parameters.
   If q gets lower than the boundary set by the designer, the iteration process ends.

2. If the improvement of succeeding steps is smaller than a chosen threshold, then the iteration is stopped.

3. To limit the time the algorithm is running, a maximum iteration boundary is also set.

Now, let's take into account the set of constraints of (7) that we left out in the previous discussion.
This can be done by checking in each iteration step if there are negative coefficients. In that case the algorithm stops, the negative coefficients are put to zero and the algorithm restarts again with only those coefficients (also considering those who were negative before) who can still contribute to a better result (i.e. their partial derivative is negative).

5 Results

As a test case, we use the circuit of Fig. 3, a high–speed CMOS operational transconductance amplifier (OTA), which we want to model in a $0.7\mu m$ CMOS technology. The supply voltage is 5V. The nominal threshold voltages of this technology are $0.76V$ for NMOS devices and $-0.75V$ for PMOS devices. The circuit has to drive a 10–pF load capacitor. Thirteen independent design variables can be chosen (see Table 1). These are all scaled according to:

$$x_i = \log\left(\frac{v_i}{lb_i}\right) \Big/ \log\left(\frac{ub_i}{lb_i}\right) \tag{10}$$

Their boundaries and nominal values are indicated in Table 1.

As an example we will derive an expression for the phase margin (PM) of this circuit in terms of the design variables. The experiments were based on an orthogonal array of strength 3 with size $dx = 0.1$ around $x_i = 0.5, \forall i \in [1:13]$

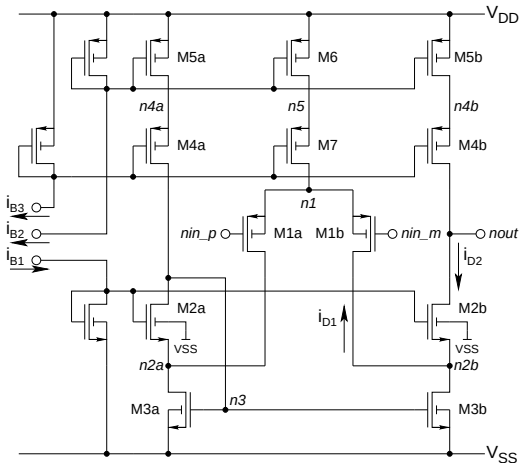

Figure 3. Schematic of a high–speed CMOS OTA.

(containing 243 experiments). These were run on 16 UNIX workstations (attached to a standard 100Mbit/s TCP–IP network and under normal load circumstances), ranging from a SUN Ultra SPARC I (with a SPECfp95 of 9) to an HP B–1000 (with a SPECfp95 of 42) using their native OS. The simulations took approximately 3 minutes.

Considering a posynomial regression template that contains a constant term (c_1), all linear (x_i) and quadratic (x_i^2) terms, all interactive terms ($x_i x_j$) and all their inverse (x_i^{-1},x_i^{-2},$x_i x_j^{-1}$) terms would result in a 807–dimensional optimization problem. In view of the highly nonlinear character of (6), using this template will take a large amount of time. We therefore take the fitted result obtained in [5, 6] as template estimator and use it as a starting template for our approach. For the PM this results in a 33–dimensional optimization problem. We then apply our regression algorithm.

i	v_i	nominal	lb_i	ub_i	i	v_i	nominal	lb_i	ub_i
1	V_{GS1}	1.73V	-0.75V	-4V	2	V_{GS2}	1.73V	0.75V	4V
3	V_{DS2}	0.63V	0.1V	4V	4	V_{GS3}	-1.73V	-0.75V	-4V
5	V_{GS4}	-1.73V	-0.75V	-4V	6	V_{GS5}	-1.73V	-0.75V	-4V
7	V_{DS5}	-0.63V	-0.1V	-4V	8	V_{DS6}	-0.63V	-0.1V	-4V
9	I_{D1}	-0.32mA	-10uA	-10mA	10	I_{D2}	0.32mA	10uA	10mA
11	I_{B1}	10uA	1uA	100uA	12	I_{B2}	10uA	1uA	100uA
13	I_{B3}	10uA	1uA	100uA					

Table 1. Design variables chosen as model inputs.

coeff		*exp*			
method of [5, 6]	*new method*	method of [5, 6]		*new method*	
$c_1 = -0.8057$	-0.9468	—	—	—	—
$c_{24} = +0.0349$	$+0.1232$	$\alpha_{24,10} = -1$	—	-0.571	—
$c_{51} = +0.0526$	$+0.0319$	$\alpha_{51,1} = -1$	$\alpha_{51,9} = +1$	-1.143	$+0.608$
$c_{64} = +0.0509$	$+0.0643$	$\alpha_{64,2} = +1$	$\alpha_{64,3} = +1$	-0.732	$+2.483$
$c_{128} = +0.0222$	$+0.0168$	$\alpha_{128,4} = +1$	$\alpha_{128,5} = -1$	$+0.724$	-1.048
$c_{141} = +0.0468$	$+0.0328$	$\alpha_{141,4} = -1$	$\alpha_{141,9} = +1$	-1.136	$+0.919$
$c_{149} = +0.0040$	$+0.0049$	$\alpha_{149,4} = +1$	$\alpha_{149,12} = -1$	$+0.983$	-1.005
$c_{170} = +0.0139$	$+0.0106$	$\alpha_{170,5} = +1$	$\alpha_{170,11} = -1$	$+0.979$	-1.179
$c_{192} = +0.0146$	$+0.0149$	$\alpha_{192,6} = -1$	$\alpha_{192,11} = +1$	-0.995	$+0.879$
$c_{203} = +0.0330$	$+0.0299$	$\alpha_{203,7} = +1$	$\alpha_{203,9} = -1$	$+0.944$	-1.213
$c_{232} = +0.1094$	$+0.2957$	$\alpha_{232,9} = +1$	$\alpha_{232,10} = +1$	$+1.297$	$+1.018$
$c_{243} = +0.0033$	$+0.0029$	$\alpha_{243,9} = -1$	$\alpha_{243,13} = +1$	-1.020	$+0.990$
Model quality for [5, 6] = 0.074013					
Model quality for our new method = 0.054978					

Table 2. Results obtained for the phase margin model for our new method compared with the results obtained in [5, 6].

The resulting expression is:

$$\begin{aligned} PM =& c_1 + c_{24}x_1^{\alpha_{24,1}} + c_{51}x_1^{\alpha_{51,1}}x_9^{\alpha_{51,9}} + c_{64}x_2^{\alpha_{64,2}}x_3^{\alpha_{64,3}} \\ &+ c_{128}x_4^{\alpha_{128,4}}x_5^{\alpha_{128,5}} + c_{141}x_4^{\alpha_{141,4}}x_9^{\alpha_{141,9}} \\ &+ c_{149}x_4^{\alpha_{149,4}}x_{12}^{\alpha_{149,12}} + c_{170}x_5^{\alpha_{170,5}}x_{11}^{\alpha_{170,11}} \\ &+ c_{192}x_6^{\alpha_{192,6}}x_{11}^{\alpha_{192,11}} + c_{203}x_7^{\alpha_{203,7}}x_9^{\alpha_{203,9}} \\ &+ c_{232}x_9^{\alpha_{232,9}}x_{10}^{\alpha_{232,10}} + c_{243}x_9^{\alpha_{243,9}}x_{13}^{\alpha_{243,13}} \end{aligned} \tag{11}$$

The coefficient and exponent values can be found in Table 2. The results obtained with the technique of [5, 6] are also indicated in the table.

To compare the quality of our new model with the quality of the model out of [5, 6] the normalized mean square error metric, as defined in section 4, was calculated. For the model quality in Table 2 the set of sampling points, used during the fitting process, was reused to calculate the model quality. It shows an improvement of 26 %. This improvement is due to some exponents that change considerably (e.g., $\alpha_{64,2}$ and $\alpha_{64,3}$).

In order to have a more realistic view on the fit quality, samples located in the interior of the fitting hypercube (as defined in section 3.2) can be selected to calculate the model quality. Table 3 shows the values of two quality parameters for our new method in comparison with the method of [5, 6] for the phase margin (PM), the unity-gain frequency (f_u), the low–frequency gain ($A_{v,LF}$) and the positive and negative slew rate (SR_p, SR_n). The model quality calculated with the set of sampling points is labeled q_{sp}, the model quality calculated with the set of points located in the interior of the fitting hypercube is labeled q_{ip}.

The differences in model quality is due to some exponents that change considerably, as was seen for the PM in Table 2. For the quality calculated with the sampling points

performance characteristic	q_{sp} [5, 6]	q_{sp} new	q_{ip} [5, 6]	q_{ip} new
PM	0.074013	0.054978	0.052019	0.063279
f_u	0.034391	0.029491	0.060076	0.045909
$A_{v,LF}$	0.057677	0.044213	0.065098	0.045412
SR_n	0.16598	0.16461	0.057331	0.055831
SR_p	0.10901	0.10513	0.26298	0.26770

Table 3. Model qualities obtained with our new method compared with the model qualities in [5, 6]. q_{sp} **quality in sampling points,** q_{ip} **quality in interior points.**

(q_{sp}), a clear improvement can be noticed for PM, f_u and $A_{v,LF}$, but it is not that clear for the two slew rates. For the quality calculated with the new points out of the interior of the hypercube (q_{ip}), for some performance characteristics there is the same clear improvement, but not for all characteristics, some even get worse.

So, there is no unambiguous proof that the method with variable exponents generates better or worse performance models than the method with constant exponents. In addition, because of the steep increase in number of variables there is a 10–20 times increase in CPU time. Which makes this variable–exponents method less attractive than the constant–exponents method of [5, 6].

6 Conclusions

In this paper, we presented a new technique to generate symbolic expressions for the performance characteristics of analog electronic circuits. The technique determines the coefficients and the exponents of a posynomial template based on performance data extracted from numerical simulations. This allows to determine the real exponents that determine the performance instead of relying on a general (but restricted) second–order model or relying on expert designer's knowledge.

References

[1] M. Hershenson, S. P. Boyd, and T. H. Lee. Optimal design of a CMOS op–amp via geometric programming. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, 20(1):1–21, Jan. 2001.

[2] P. Mandal and V. Visvanathan. CMOS op–amp sizing using a geometric programming formulation. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, 20(1):22–38, Jan. 2001.

[3] R. J. Duffin, C. Zener, and E. L. Peterson. *Geometric programming: theory and application*. John Wiley & Sons, New York, 1967.

[4] K. O. Kortanek, X. Xu, and Y. Ye. An infeasible interior–point algorithm for solving primal and dual geometric programs. *Math. Programming*, 76:155–181, 1996.

[5] W. Daems, G. Gielen, and W. Sansen. A fitting approach to generate symbolic expressions for linear and nonlinear analog circuit performance characteristics. In *Proceedings Design Automation and Test in Europe Conference*, pages 268–273, Mar. 2002.

[6] W. Daems, G. Gielen, and W. Sansen. An efficient optimization–based technique to generate posynomial performance models for analog integrated circuits. In *Proceedings Design Automation Conference*, June 2002.

[7] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for experimenters: an introduction to design, analysis and model building*. John Wiley & Sons, New York, 1978.

[8] A. S. Hedayat, S. N. J. A., and J. Stufken. *Orthogonal arrays: theory and applications*. Springer–Verlag, New York, 1999.

[9] J. C. Zhang and M. A. Styblinski. *Yield and Variability Optimization of Integrated Circuits*. Kluwer Academic Publishers, 1995.

[10] F. Leyn, G. Gielen, and W. Sansen. An efficient dc root solving algoritm with guaranteed convergence for analog integrated cmos circuits. In *Proceedings IEEE/ACM International Conference on Computer Aided Design*, pages 304–307, Nov. 1998.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. Numerical Recipes in C: the art of scientific computing. Cambridge Univ. Pres, 1995.