Robust Analog Circuit Sizing Using Ellipsoid Method and Affine Arithmetic

Xuexin Liu, Wai-Shing Luk*, Yu Song, Pushan Tang and Xuan Zeng

ASIC & System State Key Lab., Microelectronics Dept.

Fudan University, Shanghai 200433, P.R.China

luk@fudan.edu.cn

Abstract — Analog circuit sizing under process/parameter variations is formulated as a mini-max geometric programming problem. To tackle such problem, we present a new method that combines the ellipsoid method and affine arithmetic. Affine arithmetic is not only used for keeping tracks of variations and correlations, but also helps to determine the sub-gradient at each iteration of the ellipsoid method. An example of designing a CMOS operational amplifier is given to demonstrate the effectiveness of the proposed method. Finally numerical results are verified by SPICE simulation.

I. Introduction

In this paper, we will introduce a new method for analog circuit sizing under process/parameter variations via convex programming formulation. In addition, we take a design of a CMOS op-amp as an example to illustrate our method.

Several performance parameters of the CMOS op-amp are determined by the geometries and bias currents settled down by designers. Before the design process is started, a suite of detailed specifications is given, which in usual includes common-mode input voltage, output voltage swing range, open-loop voltage gain, unity-gain bandwidth, phase margin, common-mode rejection ratio, slew rate, power dissipation, total die area, and so on. Provided that all the specifications are achieved in the design, the designer can optimize some aspects of performances in his/her best. Thus this scenario can be formulated in a mathematical terminology "constrained optimization". Correspondingly those specifications become the constraints we must obey.

The CMOS op-amp design has been studied extensively and a lot of excellent design tutorials are available today [1]. With their help a list of equations can be set up so that every performance can be determined by design parameters. Obviously, the main focus of optimization is on a specific configuration that some aspects of performance achieve optima, provided all restrictions are not broken. Circuit optimization of this style does give an optimal solution at a narrow operating condition range. However, a slight variation of process or working environment may arise wrong function or even complete failure in the circuit (e.g., field failure, when the chip is used under extreme operating conditions like high temperatures). This case will drastically damage the robustness or reduce the yield. Hence industrial practice not only demands fully optimized design solutions, but also expects high robustness and yield under the varying operating conditions, statistical process tolerances and mismatches [2].

Traditional worst-case analysis suffers expensive computation that even the optimization of some small and simple circuit will take a terribly long time. This disadvantage roots in the reason that worst case is calculated on the base of real arithmetic, which also leads to underestimation of true worst case. Consequently, the results may be too optimistic and cause the final design to violate the specifications if process variation or operating condition changes a little [3].

To take variations into account, a robust mathematical programming problem is formulated in a mini-max form:

$$\begin{array}{ll} \text{minimize} & \sup_{q \in Q} f_0(y, q) \\ \text{subject to} & f_j(y, q) \leq 0 \\ & \forall q \in Q \text{ and } j = 1, 2, \cdots, m. \end{array} \tag{1}$$

In this formulation, Q stands for all the variations of parameter space and y still represents our design space. In this way the constraints $f_j(y, q) \leq 0$ are the specifications that must be met with, while the object function is the supremum of all the possible values of $f_0(y, q)$. In case of the CMOS op-amp design, almost every critical equation can be expressed in form of *geometric programming* (GP) [4]. The most helpful characteristic of geometric programming is that it can be easily reformulated as a convex optimization problem, thus we can further assume that the function $f_0(y, q)$ and the constraints $f_j(y, q)$ are convex for all $q \in Q$. Hence the feasible region is convex because the intersection of convex sets remains a convex set. As a result, the overall mini-max objective function is still convex but *non-differentiable* in general. In addition, there are infinite number of constraints in mini-max form (1) because of the variations of q. The standard interior point method may not be able to handle such problem effectively. To deal with this special form of convex programming, we propose using the *ellipsoid method*, which can be classified as the geometric approach and treat the programming progress by approximating the feasible region by a sequence of ellipsoids with decreasing

This work is supported in part by National Science Foundation of China (NSFC) Research Project 90307017 and 60676018, by National Basic Research Program of China under Grant 2005CB321701, by Cross-Century Outstanding Scholar's Fund of Ministry of Education of China, by Doctoral Program Foundation of Ministry of Education of China 20050246082, by Shanghai Science and Technology Committee (STCSM) Key Project 05JC14007 and 04JC14015, and by Shanghai AM R&D Fund 0418.

volumes. It can be proved that the ellipsoid method preserves the property of containing a bounded convex feasible region and converges to an ellipsoid the center of which is the optimal result [5].

With the aim to implement a variation-tolerant analog circuit optimization, we propose using *affine arithmetic* to participate at each iteration of the ellipsoid method. Affine arithmetic has been developed for a wide range of applications in which process variations are considered. One of the advantages of affine arithmetic over *interval arithmetic* is that it can keep tracks of correlations in computations, by sharing a set of global variables named *noise symbols*. Nevertheless, it is usually employed in analysis rather than in optimization. In this paper, we further utilize affine arithematic for finding the *sub-gradient* at each iteration of the ellipsoid method.

This paper is organized as follows. Section II reviews previous works on geometric programming. Some issues related to this area are summarized. Section III describes the ellipsoid method that will be applied in our proposed optimization program. In Section IV, we propose our affine arithmetic based method to optimize circuits as well as analyze and predict the impacts of process variations. Experimental results are described in Section V. Finally, Section VI will give the conclusion.

II. Geometric Programming in Convex Form

The geometric programming is an optimization problem which has the following form:

$$\begin{array}{lll} \text{minimize} & p_0(x) & \\ \text{subject to} & p_i(x) \leq 1, & i = 1, \dots, l \\ & g_j(x) = 1, & j = 1, \dots, m \\ & x_k > 0, & k = 1, \dots, n, \end{array} \tag{2}$$

where $p_0, p_1, \dots, p_l$ are posynomial functions and $g_1, \dots, g_m$ are monomial functions. A posynomial function is real-valued function of n real, positive variable x_k, which is defined as

$$p(x_1, \dots, x_n) = \sum_{s=1}^{t} c_s x_1^{\alpha_{1s}} x_2^{\alpha_{2s}} \cdots x_n^{\alpha_{ns}}, \quad x_k > 0, \tag{3}$$

where nonnegative coefficient $c_s \geq 0$ and $\alpha_{ks} \in$ R. When the posynomial has only one nonzero term in the sum, i.e., with the following form, it is called monomial.

$$g(x_1, \dots, x_n) = c x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}, \quad x_k > 0. \tag{4}$$

By taking the logarithm of x_k and define $y_k = \log x_k$ as well as $x_k = e^{y_k}$, we obtain $f_i(y) = \log\left(p_i(e^{y_1}, \dots, e^{y_n})\right) = \log\left(\sum_{k=1}^{t} e^{\alpha_k^T y + b_k}\right)$. Thus the problem is transformed into convex form:

$$\begin{array}{lll} \text{minimize} & f_0(y) & \\ \text{subject to} & f_i(y) \leq 0, & i = 1, \dots, l \\ & g_j(y) = 0, & j = 1, \dots, m. \end{array} \tag{5}$$

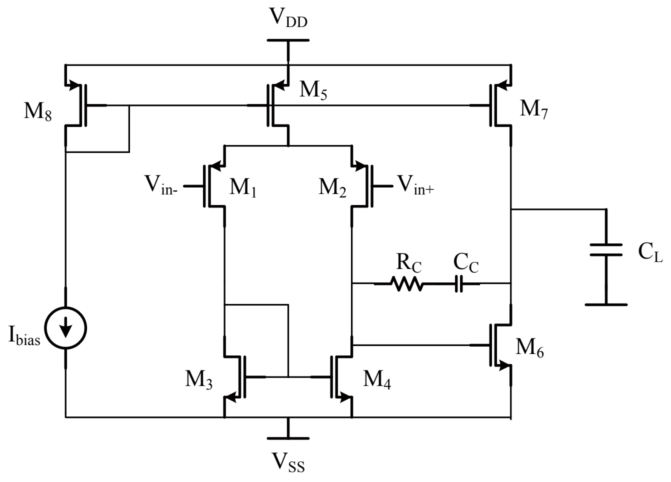


Fig. 1. A typical two-stage CMOS op-amp.

The most essential advantage of convex programming is the efficient solution of global optimum, and is especially apt to handle a large number of concurrent constraints. It can handle the problems with hundreds of variables and thousands of constraints. Furthermore, it gives an unambiguous recognition of infeasible specifications. It is also a rather versatile method that it is recommended to solve floor planning, gate scaling, Elmore delay modeling, and even analog filter or RF circuit optimization [6].

Previous works have formulated the design optimization of two-stage CMOS op-amp, with the structure in Fig. 1, as a convex optimization problem after expressing the object and constraint functions in geometric programming form [4]. It also shows that if the equations are constructed by some data fitting approaches, the results of the optimization will be more accurate and reliable when compared with simulation tools such as SPICE.

To solve the convex programming problem, it is reported in [4] that an interior-point method is implemented, which consists a primal barrier method for solving the convex form problem, and applies a modified Newton's method to minimize a smoothed convex function. However, in case that some quasi-convex or non-differentiable functions are presented as in robust design, the interior-point method may not perform well. In addition, even though an optimization is finished successfully using this method, the designers can obtain nothing more than an optimal result, i.e., some helpful hints about the covariance of the design space considering uncertainty are still unavailable. In the next section we will describe how to apply the ellipsoid method to tackle this problem.

III. The Ellipsoid Method

The ellipsoid method first generates an ellipsoid large enough to contain the feasible region. A hyperplane $g^T x = b$, can be constructed by linearizing the region boundary at a

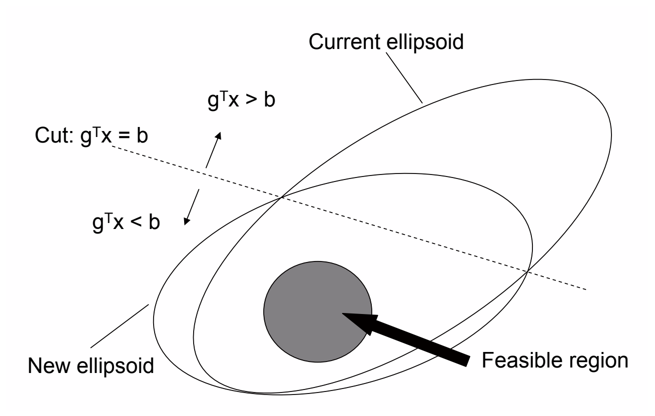


Fig. 2. In each iteration, new ellipsoid is constructed according to Algorithm 1.

boundary point. Assume g is the gradient of f, the constraint active at the boundary point x_B (if f is non-differentiable or quasi-convex, sub-gradient or quasi-gradient of f is used [7]), then the constructed hyperplane $b = g^T x_B$ divides the current ellipsoid into two parts. For a convex feasible region, one part, $g^T x > b$, is completely infeasible; the other, $g^T x < b$, contains the feasible region. Hence we can construct a new ellipsoid with a smaller volume to replace the old one. Fig. 2 illustrates this updating process in one iteration of the algorithm.

Assume that an ellipsoid is

$$E(x, A) = \left\{z \in \mathsf{R}^n | (z - x)^T A^{-1}(z - x) \le 1\right\} \quad (6)$$

with the center at x and a positive definite matrix A indicating its size and feature. The center of the starting ellipsoid may be feasible or infeasible. In case that the center is infeasible, the volume of the starting ellipsoid is expected to be larger.The basic ellipsoid method for unconstrained problem is presented in Algorithm 1.

Algorithm 1 The basic ellipsoid method [8]

```
Input: A convex function $f : \mathsf{R}^n \to \mathsf{R}$. An initial ellipsoid
    $E(x, A)$ large enough to contain the optimum $x^*$.
Output: Return optimum $x^*$ after converging to tolerance $\epsilon$.
1: repeat
2:   compute $\nabla f(x)$
3:   if $\sqrt{\nabla f(x)^T A \nabla f(x)} \le \epsilon$ then
4:     return $x$
5:   end if
     /* update ellipsoid */
6:   $\tilde{g} = \nabla f(x)/\sqrt{\nabla f(x)^T A \nabla f(x)}$
7:   $x = x - \frac{1}{n+1} A\tilde{g}$
8:   $A = \frac{n^2}{n^2-1}(A - \frac{2}{n+1} A\tilde{g}\tilde{g}^T A)$
9: until converge
```

The ellipsoid method has the following properties. First, it reduces to a bisection method when the space dimension reduces to one-dimension, i.e., R^n is R if $n = 1$. Second, the calculations about generating a new ellipsoid are less complex and arduous than other methods. If $E(x^{(k)}, A^{(k)})$ and $\nabla f(x^{(k+1)})$ are known, $E(x^{(k+1)}, A^{(k+1)})$ is found out with the complexities at most matrix-vector multiplication. Third, the new ellipsoid $E(x^{(k+1)}, A^{(k+1)})$ may be larger than $E(x^{(k)}, A^{(k)})$ in diameter (max semi-axis length), but is always smaller in volume, and we have

$$\mathrm{vol}\left(E(x^{(k+1)}, A^{(k+1)})\right) < e^{-\frac{1}{2n}} \mathrm{vol}\left(E(x^{(k)}, A^{(k)})\right),$$

where n is the dimension of variable space. This inequality ensures the convergence after a finite number of iterations. Lastly, this method can handle non-differentiable (or non-smooth) or quasi-convex constraints.

IV. Optimization Using Affine Arithmetic

The analog circuits face to the challenges posed by process variations and if designers do not give enough consideration to these impacts before manufacturing, the performance and yield of the chips will suffer much. That is why design for manufacturability (DFM) technology becomes a prerequisite in the early stage of design to enhance yield of devices in the final stage. Monte Carlo analysis is a popular method most useful in the production phase as a fine-tuning method of design. It is necessary to find the respond surface model that describes the circuit sensitivities and correlations to and among the process parameters and design variables. However, this method suffers from large computation effort and lacks reliable information of the statistical circuit response [5]. Another widely used technique is the corner-enumeration worst-case design optimization in which the circuit performance is optimized for all the corners. However, this method suffers from inefficiency and often produces an over-conservative design [9].

In this paper we propose a new methodology using affine arithmetic and give a practical algorithm implementing this optimization under process variations. In our approach, traditional standard real arithmetic are replaced by affine arithmetic. Therefore it will estimate the worst case more accurately without sacrificing the computational efficiency. Recent researches in affine arithmetic have brought innovative ideas to the field of IC design in which process/parameter variations are considered. One of such applications is the use of affine arithmetic in model order reduction (MOR) and statistical modeling of interconnect and effective capacitance [10], in which they developed new correlated interval technique for representing statistics inside algorithms, for example, an interval involved LU decomposition are solved when statistical uncertainties are encountered.

Affine arithmetic was developed from interval arithmetic, they are both self-validated numerical algorithms keeping track of the accuracy of all quantities that it computes, as part of the process of computing them. In interval arithmetic, a real quantity x is represented by an interval of floating-point numbers:

$$\bar{x} = [x_{min}, x_{max}] := \{x \in \mathsf{R} | x_{min} \le x \le x_{max}\}. \quad (7)$$

Algorithm 2 Ellipsoid method for Robust Convex Programming

```
1:  while not converge do
2:    if for some $j$, $\sup_{q\in Q} f_j(y,q) > 0$ then /* $y$ infeasible */
3:      Let $q_{\max} = \arg\sup_{q\in Q} f_j(y,q)$
4:      Find $g = \nabla f_j(y, q_{\max})$
5:      if $f_j(y,q_{\max}) - \sqrt{g^T A g} > 0$ then
6:        return infeasible
7:      end if
8:    else /* $y$ feasible */
9:      Let $q_{\max} = \arg\sup_{q\in Q} f_0(y,q)$
10:     Find $g = \nabla f_0(y, q_{\max})$
11:     if $\sqrt{g^T A g} < \epsilon$ then
12:       return optimal solution
13:     end if
14:   end if
15:   Update $Ellipsoid(y, A)$
16: end while
```

It also guaranteed that functions defined on real arithmetic can be safely extended to interval arithmetic and some necessary properties are preserved, such as inclusion isotonicity [3].

Unfortunately, interval arithmetic often yields a result wider in range than the exact result. For instance, assume $x = [-1,1]$, it will give a subtraction $x-x = [(-1)-1, 1-(-1)] = [-2,2]$ instead of [0,0], which is the exact range of the result. The reason for this overestimation is the missing of information on the correlation of intervals in (7) where an interval is defined.

Affine arithmetic was proposed to overcome the error overestimation in such a method that besides recording a range for each quantity, it also keeps track of correlations between those quantities, in order to add information to the process of computing [11]. In affine arithmetic a quantity x is formulated in an affine form $\hat{x}$:

$$\begin{aligned}\hat{x} &= x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_k\varepsilon_k \\ &= x_0 + \sum_{i=1}^{k} x_i\varepsilon_i, \quad -1 \le \varepsilon_i \le 1, \quad x_i \in R \qquad (8)\end{aligned}$$

All ε_i are called noise symbols whose values are unknown but assumed to lie in the interval $[-1,1]$ and they are all global variables that can be shared by all affine forms. Thus if we take out a computation between two variables sharing the same noise symbol, the error will not be exaggerated as in the preceding case that an interval arithmetic came across.

Now we propose our affine arithmetic based ellipsoid method which is illustrated in Algorithm 2. As defined in minimax form (1), assume all $q \in Q$ represent the parameters under process variations and are initialized as affine forms, our algorithm starts a modified ellipsoid method for robust convex programming.

In Section III, we have known that a hyperplane is constructed at a boundary point and then the gradient at that point is computed. The key issue is to determine this point, and here affine arithmetic is applied with special consideration. In our algorithm, this point have an intimate relation with q_{max} which is defined in supremum over a set of convex functions.

For instance, let q_{max} denotes the parameter q where $f_j(y,q)$ is maximized for a fixed y, i.e.,

$$q_{\max} = \arg\sup_{q\in Q} f_j(y,q). \qquad (9)$$

After affine arithmetic operations, f_j is approximated as an affine form:

$$\begin{aligned}\hat{f}_j &= f_{j,0} + f_{j,1}\varepsilon_1 + f_{j,2}\varepsilon_2 + \cdots + f_{j,k}\varepsilon_k \\ &= f_{j,0} + \sum_{i=1}^{k} f_{j,i}\varepsilon_i, \quad \varepsilon_i \in [-1,1]. \qquad (10)\end{aligned}$$

We have that $\hat{f}_j$ is maximized when

$$\varepsilon_i = \begin{cases} +1 & \text{if } f_{j,i} > 0, \\ -1 & \text{if } f_{j,i} < 0. \end{cases} \qquad (11)$$

This is to say that we can determine ε_j be either $+1$ or -1 and hence $q_{\max}$ is determined.

As can be proved in [7], if all $f_j(y,q)$ functions are convex, the supremum $\sup_{q\in Q} f_j(y,q)$ is also convex, though it is non-differentiable in general. Nevertheless, this will not make much difficulty for the ellipsoid method to solve it.

V. Example

In this section, we will describe a simple example and show the performance of our approach. We implemented an optimization tool prototype on two-stage CMOS op-amp based on our proposed method in C++ with the Libaffa package [12]. In fact, it is convenient that the researchers of different interest can build customized affine arithmetic libraries for their own use. We have fulfilled many useful functions and added them to the library, which will greatly facilitate our computations.

The schematic of our circuit is shown in Fig. 1. The specifications of the op-amp, which are the constraints of the optimization tool, are listed in Table I. The objective is the term to be maximized or minimized subject to all the specifications. Here our object is the unity gain frequency. The design verification of our results are based on HSPICE level-1 model.

If process variations are not considered, the design progress is purely a geometric programming to obtain an optimization of some terms of performance. The results of our tool are the parameters shown in column "Standard" in Table II. And the corresponding performance are listed in column "Standard" in Table III. It is very reliable that the results of HSPICE meet the specifications in Table I.

If we set some process parameters to be affine arithmetic variables, which means these parameters are under the impacts of process variations, our optimization tools will exert the improved ellipsoid method that we described in last section and finally give an optimal answer under process variations.

TABLE I
SPECIFICATIONS FOR THE EXAMPLE OF OP-AMP.

Constraint	Units	Spec.
Device Width	μm	≥ 2.0
Device Length	μm	≥ 0.8
Estimated Area	μm^2	≤ 20000
Supply Voltage	V	$V_{DD} = 5$ $V_{SS} = 0$
Common-Mode Input	V	$V_{DD}/2$
Output Range	V	$[0.1, 0.9]V_{DD}$
Gain	dB	≥ 80
Unity Gain Freq.	MHz	Maximize
Phase Margin	degree	≥ 60
Slew Rate	V/μs	≥ 10
Load Capacitance	pF	3
CMRR	dB	≥ 60
Neg. PSRR	dB	≥ 80
Power	mW	≤ 5
Noise, Flicker	nV/$\sqrt{\text{Hz}}$	≤ 300

TABLE II
OPTIMAL DESIGN RESULTS USING THE PROPOSED METHOD.

Variable	Standard	Robust
$W_1 = W_2$	401.3μm	422.9μm
$W_3 = W_4$	132.6μm	139.7μm
W_5	91.5μm	67.4μm
W_6	183.8μm	134.9μm
W_7	62.9μm	32.2μm
W_8	2.0μm	2.0μm
$L_1 = L_2$	0.8μm	0.8μm
$L_3 = L_4$	0.8μm	0.8μm
$L_5 = L_7 = L_8$	0.8μm	0.8μm
L_6	0.8μm	0.8μm
R_c	110.0Ω	200.0Ω
C_c	7.4pF	9.8pF
I_{bias}	12.7μA	19.3μA

For example, here we set the MOSFET threshold voltage, lateral diffusion length, gate oxide thickness and power supply voltage under process/parameter variations:

$$\begin{aligned}
V_{TH,NMOS} &= [0.65, 0.75]\ \text{V} \\
V_{TH,PMOS} &= [-0.95, -0.85]\ \text{V} \\
L_D &= [0.19, 0.21]\ \mu\text{m} \\
T_{OX} &= [16, 24]\ \text{nm} \\
V_{DD} &= [4.7, 5.3]\ \text{V}.
\end{aligned}$$

Then the optimal design will like the values in column "Robust" in Table II. And the performances of the circuit using these parameters are shown in column "Robust" in Table III. If we compare the transistors' geometric parameters, e.g. W_1 to W_8 in "Standard" and "Robust" columns in Table II, it can be inferred that after the process variations are considered

TABLE III
PERFORMANCE OF OP-AMP WITH OPTIMAL DESIGN'S RESULTS.

Performance	Standard	Robust
Estimated Area	16400μm^2	18200μm^2
Output Range	$[0.08, 0.92]V_{DD}$	$[0.08, 0.92]V_{DD}$
Gain	92dB	91dB
Unity Gain Freq.	70MHz	74MHz
Phase Margin	61.3degree	64.5degree
Slew Rate	21V/μs	22V/μs
CMRR	95dB	95dB
Neg. PSRR	101dB	101dB
Power	5.0mW	4.9mW
Noise, Flicker	298nV/$\sqrt{\text{Hz}}$	290nV/$\sqrt{\text{Hz}}$

in the design, the standard sizing results must be relaxed so as to allocate enough margins for the performances. Therefore the circuits can still meet the specifications even though process/parameter variations occur.

Each of the two cases was solved in less than one second on Intel Pentium IV running at 2.4 GHz.

Fig. 3 illustrates the variation of lateral diffusion L_D, and here we plot three lines which are the cases $L_D = 0.19\mu$m, $L_D = 0.20\mu$m, and $L_D = 0.21\mu$m. We can find out that the performances in the range of L_D satisfy our specifications. The open loop gains are larger than the one we set in Table I and phase margins are all greater than 60 degree. Fig. 4(a) shows the case when power supply voltage V_{DD} and L_D are changed simultaneously in the range $V_{DD} = [4.7, 5.3]$V and $L_D = [0.19, 0.21]\mu$m, while Fig. 4(b) plots how the value of phase margin changes when L_D and oxide thickness T_{OX} are both under variations in the range $L_D = [0.19, 0.21]\mu$m and $T_{OX} = [16, 24]$nm. As a result, the phase margin never become less than 60 degree. It confirms the validity of our optimization tool again since the constraints we set previously are strictly met.

VI. CONCLUSIONS

In this paper we have presented a novel approach to variation-tolerant analog circuit sizing via robust convex programming formulation. We proposed using the ellipsoid method to approximate the feasible region at each iteration, thus guarantee the bounding of the whole design space with good efficiency. Furthermore, we adopt affine arithmetic rather than standard real arithmetic to solve the optimization problem. It can precisely take the variations and worst-case performances into account without loss of efficiency. The example of a two-stage CMOS op-amp given in this paper demonstrated the practical usability of our solution.

REFERENCES

[1] Behzad Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill, 2001.

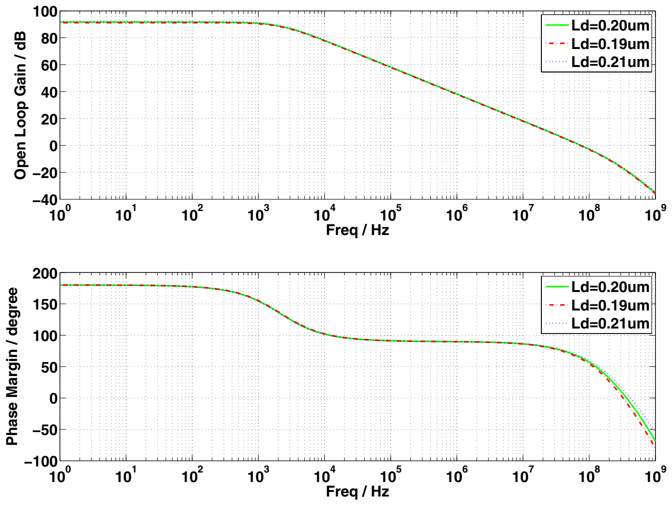
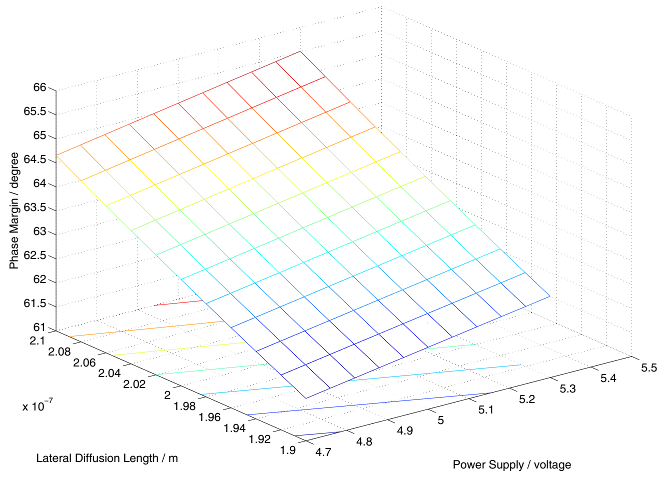
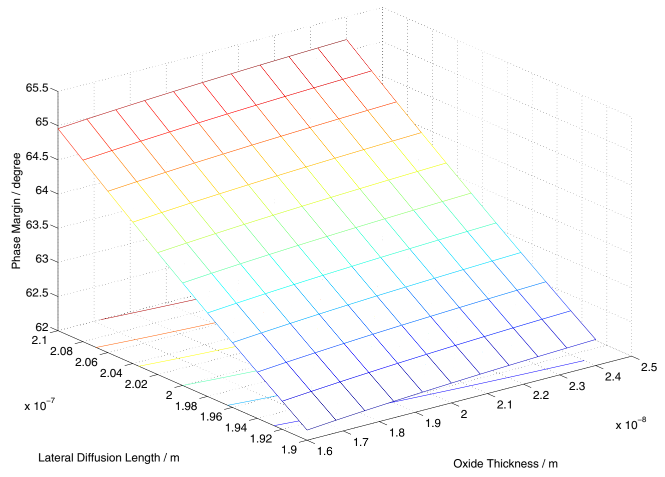


Fig. 3. Open loop gain and phase margin versus frequency. (The solid line represent the case when $L_D = 0.20\mu$m, the dash-dot line $L_D = 0.19\mu$m, and the dotted line $L_D = 0.21\mu$m.)



(a) Phase margin vs. the power supply voltage V_{DD} and MOSFET Lateral diffusion Length L_D.



(b) Phase margin vs. MOSFET Lateral diffusion Length L_D and gate oxide thickness T_{OX}.

Fig. 4. Frequency response under the variation of power supply, Lateral diffusion Length and gate oxide thickness.

[2] S. Director, W. Maly, and A. Strojwas. *VLSI Design for Manufacturing: Yield enhancement*. Kluwer, Norwell, MA, 1990.

[3] Andreas Lemke, Lars Hedrich, and Erich Barke. Analog circuit sizing based on formal methods using affine arithmetic. In *IEEE/ACM Proc. Int. Conf. Computer Aided Design*, pages 486–489, November 2002.

[4] Maria del Mar Hershenson, Stephen P. Boyd, and Thomas H. Lee. Optimal design of a CMOS op-amp via geometric programming. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 20(1):1–21, January 2001.

[5] Hany L. Abdel-Malek and Abdel-Karim S. O. Hassan. The ellipsoidal technique for design centering and region approximation. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 10(8):1006–1014, August 1991.

[6] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming. Technical report, Electrical Engineering Department, Stanford University, September 2004. Available at `http://www.stanford.edu/~boyd/gp_tutorial.html`.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[8] S. Boyd. Lecture notes of convex porgramming. Available at `http://www.stanford.edu/class/ee364`.

[9] Yang Xu, Kan-Lin Hsiung, Xin Li, Ivan Nausieda, Stephen Boyd, and Lawrence Pileggi. OPERA: Optimization with ellipsoidal uncertainty for robust analog IC design. *Proc. Design Automation Conf.*, pages 632–637, June 2005.

[10] James. D. Ma and Rob. A. Rutenbar. Fast interval-valued statistical modeling of interconnect and effective capacitance. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 25(4):710–724, April 2006.

[11] J. Stolfi and L. H. de Figueiredo. An introduction to affine arithmetic. *Tend. Mat. Apl. Comput.*, 4(3):297–312, 2003.

[12] Olivier Gay, David Coeurjolly, and Nathan Hurst. Libaffa: C++ affine arithmetic library for GNU/Linux. Available at `http://savannah.nongnu.org/projects/libaa/`, May 2005.