Analog System Verification in the Presence of Parasitics using Behavioral Simulation*

Edward W. Y. Liu Henry C. Chang Alberto L. Sangiovanni-Vincentelli

Department of EECS, University of California, Berkeley, CA 94720

Abstract

In analog system design, final verification in the presence of parasitic loading effects is crucial to guarantee functionality of the entire circuit. In this paper, we present a methodology for analog system verification in the presence of parasitics using behavioral simulation. When applied to a synthesized 10 bit D/A, our approach is accurate to 0.005 LSB compared with SPICE, while being several orders of magnitude faster.

1 Introduction

We have proposed a constraint-driven, top-down design methodology for mixed-mode systems[1] that is supported by analog design tools. An integral part of such methodology is the complete verification of the synthesized circuit in the presence of parasitics due to routing, supply variations, and coupling. Because parasitics degrade analog system performance, it is crucial to verify by simulation as completely as possible the circuit functionality in the presence of these second order effects.

Existing design methodologies, manual or automatic [2, 3, 4], use SPICE simulations to verify the final designs at the transistor level. Unfortunately, SPICE simulations often fail for larger systems due to non-convergence or unreasonable computational cost. For example, to simulate process variations of a circuit, designers traditionally use worst case analysis. In this approach, designers simulate all, 2^M, process corners where M is the number of process variables. In a large system such as a 10 bit D/A with 600 matched transistors, the number of process corners reaches an astronomical 2^{600}, rendering exact worst case analysis impossible. Consequently, designers simulate a subset of the process corners, but no general algorithm has been proposed for choosing the subset.

Due to the difficulties, sevaral behavioral simulation approaches have been proposed earlier[5, 6]. Unfortunately, they do not take into account loading parasitics and must also use worst case analysis to compute process variation effects.

In some design methodologies such as standard cell analog designs, designers verify each cell extensively by SPICE simulation. Yet, the final system of interconnected cells may fail due to interconnect parasitics or cell interactions.

In contrast, our top-down design methodology[1] decomposes a system into its components, then the components into subcomponents using the same methodology until constraint-driven physical layout synthesis. In such a design environment, the key verification tasks are the verification of component functionality and the verification of the system *after* the components are routed in a system. We use the former type of verification right after component synthesis to check whether or not an individual component works *without loading effects*. If not, we redesign the component immediately, thus avoiding expensive design iterations. This type of verification is easier because the isolated component can often be verified with SPICE. After we verify each component against its specifications, we check the system performance in the presence of layout parasitics. This last step has been carried out in the past at the SPICE simulation level. In this paper, we propose a new methodology that uses behavioral models of the components and behavioral simulation. The benefits of the approach are overwhelming when compared to SPICE level simulation.

Section 2 introduces the new verification strategy, followed by the application of the strategy to verification of Nyquist D/A converters in Section 3. Experimental results are presented in Section 4, followed by a conclusion in Section 5.

2 Verification in the presence of parasitics between components

In behavioral simulation, we follow these steps:

- Development of behavioral models of components.
- Component layout extraction.
- Component identification using SPICE simulation of the components at ideal bias conditions.
- Parameter fitting for behavioral models.
- System simulation at the *behavioral* level using behavioral models only.

The problem with this approach is that the components are characterized at ideal bias conditions. Because of parasitics, e.g. parasitic resistance, the components bias will shift causing system performance degradation. To remedy the problem, we propose the following approach.

In general, suppose the system performance, S, is a function of the behavioral model parameters, W, which in turn are functions of the bias conditions, V. Due to parasitics, p, V becomes

$$V = V_0 + \frac{\partial V}{\partial p}p + \frac{\partial^2 V}{\partial p^2}p^2 + \ldots \approx V_0 + \frac{\partial V}{\partial p}p \qquad (1)$$

where V_0 is the ideal bias. The first order Taylor approximation is justified by the fact that most components have linear impedance characteristics for the typical range of parasitics. Therefore,

$$W(V) \approx W(V_0) + \frac{\partial W}{\partial V}\frac{\partial V}{\partial p}p \qquad (2)$$

$$S(W(V)) \approx S(W(V_0)) + \frac{\partial S}{\partial W}\frac{\partial W}{\partial V}\frac{\partial V}{\partial p}p \qquad (3)$$

using first order Taylor approximation. For (2) and (3) to be valid, the partial derivatives $\frac{\partial S}{\partial W}$ and $\frac{\partial W}{\partial V}$ must exist.

To use (3) to compute effects due to parasitic loading, we change the verification methodology to the following:

*This project is supported by the Semiconductor Research Corporation under grant 93-DC-008. Its support is gratefully acknowledged.

- Layout extraction,
- Component identification using SPICE simulation of the components at ideal bias conditions V_0,
- Parameter fitting for behavioral models.
- Component linearization at ideal bias conditions,
- Linearized component insertion into interconnection network,
- Network solving for new bias conditions V,
- Optional step: Sensitivity computation of V with respect to all elements in the interconnect using adjoint techniques[7].
- Bias change (from ideal bias conditions) computation, $V - V_o$, on linearized components.
- Changes (sensitivity) of V propagation to changes (sensitivity) of behavioral model parameters W, and then to changes (sensitivity) of system performance S.

The advantage of the new approach is that we can compute system level performance sensitivity due to routing parasitics. In our performance driven layout synthesis approach, the layout synthesis tools use the sensitivity information intelligently to assemble a global layout from a collection of component layouts. Then, a behavioral simulator computes the system performance degradation due to the extracted routing parasitics.

3 Verification of D/A converters

In this section we apply the new verification approach to a synthesized current-switched, interpolative 10 bit D/A (Figure 1) that consists of a 5 bit linear current source array, a 5 bit binary current source array, and a mirror. The key non-idealities in the design are mismatches in

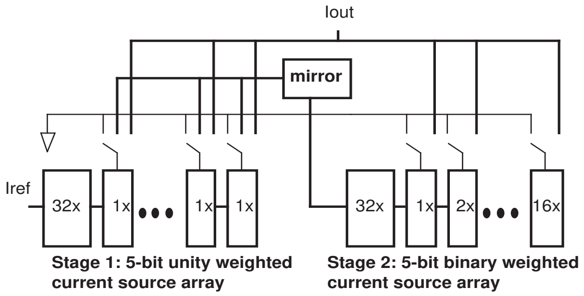

Figure 1: 10 bit interpolative D/A architecture

the transistors, parasitic resistances, parasitic capacitances, and output resistances in the transistors. Capacitances cause output glitches during fast switching. Because the converter is assumed to operate at a moderate speed, parasitic capacitances are less significant and ignored in this design. Parasitic resistances, transistor mismatches, and output resistances degrade system performance such as integral nonlinearity. We compute the system performance in three steps. First, we extract SPICE netlists for each component from the layout, then behavioral models parameters from the netlists, and finally system performance from behavioral models. Because we only consider effects due to parasitic resistance and resistive loading in our D/A, we only implemented software to compute system degradation due to these effects. As a result, software implementation to compute effects due to parasitic capacitances and inductances will be developed in the future.

3.1 From layout to SPICE netlist

An extractor capable of extracting resistances is used to convert the layout to schematic. Unique to our system is the process variation random vector, v, which is assumed to be Gaussian distributed,

$$v \sim normal(0, \Sigma_v) \tag{4}$$

where Σ_v is obtained by combining information about the process and the *geometry of the layout*. Suppose transistor i has parameter, $v(i)$, that depends on its location on the wafer,

$$v(i) = f(x_i, y_i, r_1, \ldots, r_n)$$

$$\approx \frac{\partial f}{\partial r_1}(x_i, y_i, r_1, \ldots, r_n) r_1 + \ldots + \frac{\partial f}{\partial r_n}(x_i, y_i, r_1, \ldots, r_n) r_n \tag{5}$$

where x_i is the x-coordinate, y_i is the y-coordinate of the transistor, $r_1 \ldots r_n$ are zero mean random coefficients, and the approximation is justified by a Taylor approximation. The random coefficients correspond to random process variations such as random slope and orientation of process gradients. The joint distribution of the parameter variation of k different transistors has covariance matrix

$$\Sigma_v = \frac{\partial v}{\partial r} \Sigma_r \frac{\partial v}{\partial r}^T \tag{6}$$

$$\frac{\partial v}{\partial r}(i, j) = \frac{\partial v(i)}{\partial r_j} \tag{7}$$

where Σ_r is the covariance of the coefficients.

We applied the above theory to the analysis of process gradient effects. Random coefficients, r_1 and r_2, corresponds to random slopes on the x-axis and the y-axis, respectively, such that

$$v(i) = f(x_i, y_i, r_1, r_2) = r_1 x_i + r_2 y_i \tag{8}$$

where (x_i, y_i) is the location of transistor i. Suppose the covariance of $[r_1\ r_2]^T$ is

$$\Sigma_r = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} \tag{9}$$

we compute the covariance of the parameters of two transistor located at (x_1, y_1) and (x_2, y_2) from

$$\Sigma_v = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix} \Sigma_r \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix}^T \tag{10}$$

Next, we express the mismatch of the two parameters as $v(1) - v(2)$, a linear transformation of v,

$$v(1) - v(2) = \begin{bmatrix} 1 \\ -1 \end{bmatrix} v$$

Consequently, the variance of mismatch is given by $\sigma^2 = \sigma_r^2((x_1 - x_2)^2 + (y_1 - y_2)^2))$, implying that the mismatch σ is proportional to the Euclidean distance between two components. Such prediction is consistent with data in [8][9]. Although theories for mismatches in MOS transistors were developed in [9] based on **spatial frequency**, we propose here a new theory that can compute covariance of transistor parameters in large transistor arrays efficiently by using sensitivity analysis and matrix algebra.

3.2 From SPICE netlist to behavioral models

We extract behavioral model parameters from netlist for each component using SPICE. We compute parameters by applying *ideal* voltage and current sources, V_s, to the pins of the components. For instance, the mirror model shown in Figure 2 has four parameters; namely, the current mismatch, $K_M = \frac{I_{out}}{I_{in}}$, the output resistance of the mirror, R_M, the standard deviation of K_M, S_M, and the input voltage of the mirror, V_{in}. We compute current mismatch K_M and input voltage V_{in}

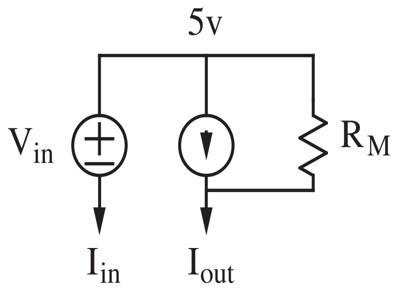

Figure 2: Mirror behavioral model

by a DC analysis, the output resistance R_M by an AC analysis, and the standard deviation S_M by

$$S_M = \left[\frac{\partial K_M}{\partial v}\right] \Sigma_v \left[\frac{\partial K_M}{\partial v}\right]^T \tag{11}$$

where v is a random vector of process variations (4) such as the width and length variations of all the devices. For simplicity, we compute $\frac{\partial K_M}{\partial v}$ using finite differences, and Σ_v from layout and process variation information using (10).

To characterize the effects due to interconnect and loading parasitics, we first compute the sensitivity of the bias due to changes in ideal sources V_s using AC analysis in SPICE. We obtain a *linearized* mirror model for later insertion into the linear interconnect network.

We characterize other components in the D/A the same way. But, due to the mixed-mode nature of some components in the D/A, the parameters for some linearized components change as a function of the input bits. For example, the output impedance of the 5 bit linear array decreases with the input code because more current sources turning on in parallel decreases output resistance. To characterize completely the linear array, we take 32 sets of linearized parameters corresponding to every combination of the 5 bit input code. The same characterization is done for the binary array.

3.3 From behavioral models to INL

In this subsection we introduce the converter behavioral model along with techniques to compute system performance from the model.

Definition 1 *The ideal output vector, L, of a D/A converter is an n-dimensional vector, where the i^{th} component of L, $L(i)$, represents the desired output for input code i, $n = 2^N$, and N is the number of bits.*

Definition 2 *The output vector, t, of a D/A converter is an n-dimensional vector, where the i^{th} component of t, $t(i)$, is the output for input code i.*

In our previous work[10], we approximate the distribution of t as a multivariate Gaussian distribution,

$$t \sim normal(\mu_t, \Sigma_t) \tag{12}$$

We compute the nominal values, μ_t, by calculating the output current for every input code using a behavioral simulator which takes as input the deterministic mismatch, the output resistance, and the input voltage of all the components. Next, we compute the covariance Σ_t using

$$\Sigma_t = \left[\frac{\partial t}{\partial K_M}\right] S_M^2 \left[\frac{\partial t}{\partial K_M}\right]^T + \left[\frac{\partial t}{\partial K_1}\right] \Sigma_{K_1} \left[\frac{\partial t}{\partial K_1}\right]^T + \left[\frac{\partial t}{\partial K_2}\right] \Sigma_{K_2} \left[\frac{\partial t}{\partial K_2}\right]^T \tag{13}$$

where vector parameters K_1 and K_2 correspond to the mismatches of the array of outputs in the linear and binary stages, respectively. Matrix parameters Σ_{K1} and Σ_{K2} correspond to the covariance of the mismatches of the array of outputs in the linear and binary stages, respectively.

The output vector, t, is sufficient to describe the converter behavior due to process variations. Designers traditionally used a different set of specifications which can be derived easily from t. Relevant specifications are

Definition 3 *Offset error: $t(1) - L(1)$.*

Definition 4 *Full scale gain error: $t(n) - L(n) + L(1) - t(1)$.*

The integral nonlinearity is the deviation of the output from the ideal value *after* compensation for the gain and offset errors.

Definition 5 *The integral nonlinearity vector, s, of a D/A converter is an n-dimensional vector*

$$s(i) = at(i) + b - L(i) \tag{14}$$

where a, b are constants such that $s(1) = s(n) = 0$.

Intuitively, Definition 5 means that we transform t using constants a and b such that the two end-points are ideal. In this case, we have adopted the more popular end-point method for linear error compensation[11]. The motivation behind the transformation is to compensate for the offset and gain errors before calculation of nonlinearity. In many applications, offset and gain errors are irrelevant as they do not contribute to distortion.

Lemma 1 *Integral nonlinearity is given by*

$$s(i) = \left[\frac{L(n) - L(1)}{t(n) - t(1)}\right] (t(i) - t(1)) + L(1) - L(i) \tag{15}$$

The sensitivity of the integral nonlinearity to an output vector is given by

$$\frac{\partial s}{\partial t}(i,j) = \begin{cases} \frac{(L(n)-L(1))(t(i)-t(n))}{(t(n)-t(1))^2} & i \neq 1,n \quad j = 1 \\ 0 & i \neq 1,n \quad j \neq 1,i,n \\ \frac{L(n)-L(1)}{t(n)-t(1)} & i \neq 1,n \quad j = i \\ 0 & i = 1,n \quad j = 1 \ldots n \\ \frac{(L(n)-L(1))(t(1)-t(i))}{(t(n)-t(1))^2} & i \neq 1,n \quad j = n \end{cases} \tag{16}$$

Proof. Solving for a, b using constraints in Definition 5 yield (15). Equation (16) follows from differentiation of (15). ■

Using a first order Taylor approximation of s as a function of t,

$$s \approx \frac{\partial s}{\partial t}(t - \mu_t) + \mu_s \tag{17}$$

we find the integral nonlinearity to have distribution,

$$s \sim normal(\mu_s, \Sigma_s) \tag{18}$$

where $\mu_s = s(t)$, at $t = \mu_t$ using (15) and $\Sigma_s = \frac{\partial s}{\partial t}\Sigma_t \frac{\partial s}{\partial t}^T$ at $s = \mu_s$. In general, the rank r_s of Σ_s is much less than full rank since not all the INL errors are independent. In other words, any realization of s must fall into a space spanned by r_s independent basis vectors. A set of such independent basis vectors can be found by singular value decomposition[12] of Σ_s. Mathematically,

$$\Sigma_s = U_s \Sigma_{cs} U_s^T \tag{19}$$

where U_s is a 2^N x r_s transformation matrix, Σ_{cs} is an r_s x r_s diagonal matrix, and r_s is the rank of Σ_{cs}. The columns of U_s are called the **INL signatures** and form a set of orthonormal vectors spanning the **INL space**. For storage and computational efficiency s is expressed as the following equation,

$$s = U_s c_s + \mu_s \tag{20}$$

where $c_s \sim normal(0, \Sigma_{cs})$ is a small, zero mean multivariate normal distribution with r_s statistically *independent* variates.

The motivation behind finding the distribution s is that INL is an important user constraint. By using the distribution of s, bounds on s can be determined quickly and made to fit user constraints during design.

Definition 6 *The $\pm k\sigma$ INL bounds are*

$$s_{\pm k\sigma}(i) = \mu_s(i) \pm k\sqrt{\Sigma_s(i,i)}. \tag{21}$$

These bounds are useful for design since any realization of s will have a high probability of falling within the $k\sigma$ bounds if $k \geq 2$. Furthermore, once the design is done, circuit yield can be computed by Monte Carlo integration over the distribution s.

4 Experimental results

We verified an "industrial strength" 10 bit D/A[1] bottom-up from layout to system level performance on a DEC 5000/125. By using the adjoint technique for sensitivity analysis on the linearized chip-level interconnect network, we computed efficiently the INL performance sensitivity with respect to all chip level routing resistances. According to our goal of constraint-driven layout synthesis, the layout synthesis tools use the sensitivity information intelligently to assemble a global layout from a collection of component layouts. Figure 3 shows the sensitivity of INL with respect to the resistance from the ground pad to the linear array.

To confirm the accuracy of the new approach, we compare our results with SPICE simulation of the final extracted circuit. Due to the long simulation times for SPICE, we consider only one process variable, the main transistor width of the mirror, when computing the $\pm 3\sigma$ worst case INL (21). The behavioral simulation results are verified against SPICE results for 18 random input codes in Figure 4. Each data point on the graph is a comparison, and a perfect match will fall on the 45 degree line. Behavioral simulation achieved an accuracy of 0.005LSB. Furthermore, results from behavioral simulation for all

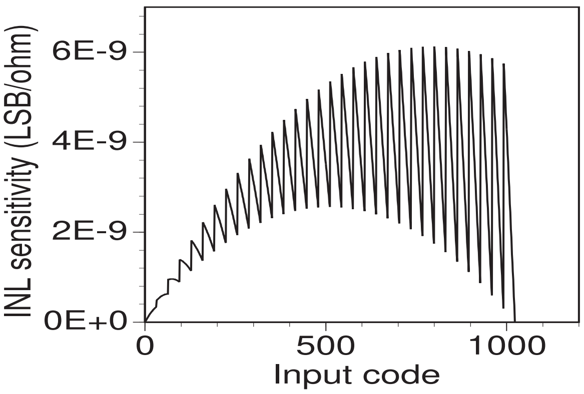

Figure 3: INL sensitivity to R from Vss pad to linear array

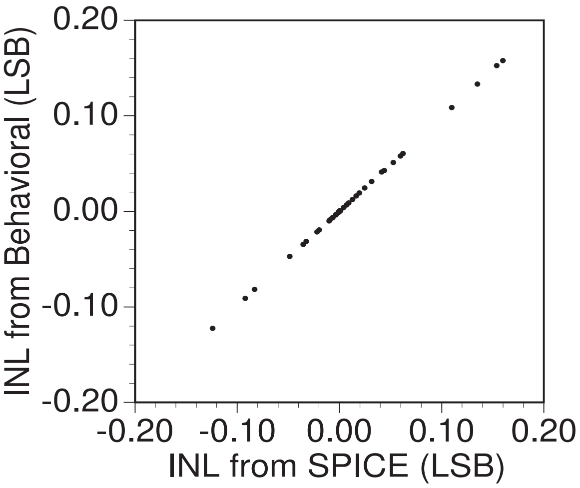

Figure 4: Accuracy comparison between SPICE and behavioral simulation

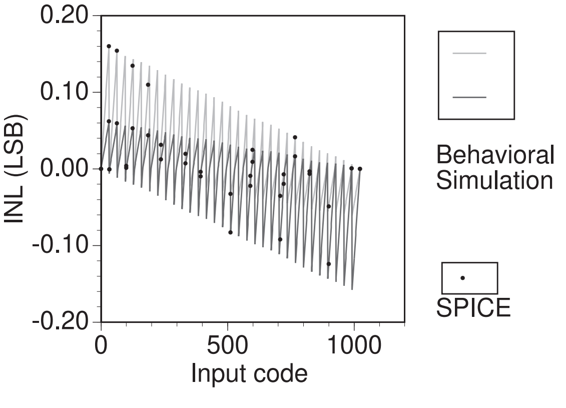

Figure 5: $\pm 3\sigma$ INL bounds compared with SPICE

1024 input codes plotted along with results from SPICE on Figure 5 show good agreements.

In our example, SPICE takes 118 CPU seconds per input code. For behavioral simulation, the characterization of the components takes 1000 CPU seconds, the linear network solver takes 100 CPU seconds, and the behavioral simulation takes 100 CPU seconds. On average for 1024 input codes, behavioral simulation takes 1.2 CPU seconds per input code. Therefore, the speed advantage is 100 *for simulation of one process variable*. For M process variable, SPICE would take 59×2^M CPU seconds per input code, while behavioral simulation would take about the same time per input code because the CPU time is dominated by the characterization of the components. For our circuit with 600 transistors, M is at least 600, so the speed advantage of behavioral simulation is proportional to 2^{600}.

As a result of the speed advantage, we verified our D/A performance under many types of process variations. Assuming typical process data such as transistor width and length standard deviation of $0.05\mu m$ and threshold variation of $\frac{0.0623 V m}{\sqrt{WL}}$, we computed the $\pm 3\sigma$ INL bounds (Figure 6) for the converter. From our theory, the INL of any fabricated chip will likely fall within the bounds. The maximum of the INL bounds is 1.5LSB which satisfy the constraints of 2.0LSB.

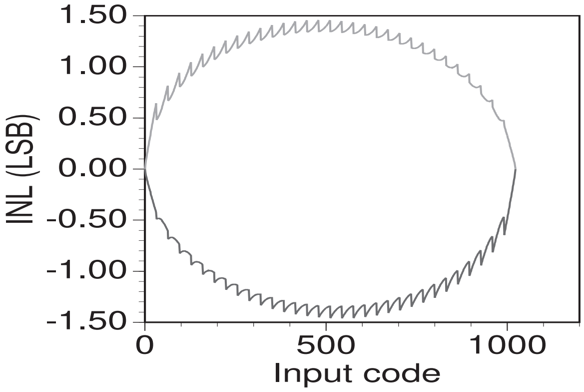

Figure 6: $\pm 3\sigma$ INL bounds

5 Conclusion

In analog system design, final verification in the presence of parasitic loading effects is crucial to guarantee functionality of the entire circuit. The traditional system verification strategy based on SPICE is too slow for large systems, especially when worst case analysis is used to simulate process variations. In our approach, we take advantage of the hierarchical decomposition of the system into components. As a result, we can extract each component individually and verify it under ideal bias conditions. Then, to verify the system in the presence of parasitic coupling between components, we linearize the components at the operating point, substitute the linearized component in the interconnect network, find the changes in the bias conditions, and estimate the performance deviation due to bias changes using a first order Taylor approximation. From experimental results, we have proven the accuracy of this verification approach for a 10 bit interpolative D/A to be within 0.005 LSB compared with SPICE. From CPU time analysis, this verification approach can be many orders of magnitude faster than SPICE.

References

[1] H. Chang, A. Sangiovanni-Vincentelli, F. Balarin, E. Charbon, U. Choudhury, G. Jusuf, E. Liu, E. Malavasi, R. Neff, and P. Gray. A top-down, constraint-driven design methodology for analog integrated circuits. In *Proc. IEEE Custom Integrated Circuits Conference*, pages 841–846, May 1992.

[2] H. Y. Koh, C. H. Séquin, and P. R. Gray. Automatic synthesis of operational amplifiers based on analytic circuit models. In *Proc. IEEE ICCAD*, pages 502–505, 1987.

[3] L. R. Carley and et. al. Acacia: The CMU analog design system. In *Proc. IEEE Custom Integrated Circuits Conference*, 1989.

[4] P. E. Allen and P. R. Barton. A silicon compiler for successive approximation A/D and D/A converters. In *Proc. IEEE Custom Integrated Circuits Conference*, pages 552–555, 1986.

[5] H. A. Mantooth and P. E. Allen. Behavioral simulation of a 3-bit flash ADC. In *Proc. IEEE Int. Symposium on Circuits and Systems*, pages 1356–1359, 1990.

[6] G. Ruan. A behavioral model of A/D converters using a mixed-mode simulator. *IEEE Journal of Solid State Circuits*, SC-26, No. 3, March 1991.

[7] L. O. Chua and Pen-Min Lin. *Computer aided analysis of electronic circuits: algorithms & computational techniques*. Prentice-hall, 1975.

[8] C. Michael and M. Ismail. Statistcial modeling of device mismatch for analog mos integrated circuits. *IEEE Journal of Solid State Circuits*, SC-27, No. 2, November 1992.

[9] M. Pelgrom, A. Duinmaijer, and A. Welbers. Matching properties of mos transistors. *IEEE Journal of Solid State Circuits*, SC-24, No. 5, October 1989.

[10] E. Liu, A. Sangiovanni-Vincentelli, G. Gielen, and P. Gray. A behavioral representation for nyquist rate A/D converters. In *Proc. IEEE ICCAD*, pages 386–389, November 1991.

[11] Analog Devices. *Data converter reference manual volume II*. Norwood, MA 02062-9106, 1992.

[12] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C. The Art of Scientific Computing*. Cambridge University Press, 1989.