Interval-Valued Reduced Order Statistical Interconnect Modeling

James D. Ma and Rob A. Rutenbar
Department of Electrical and Computer Engineering
Carnegie Mellon University
jdma@ece.cmu.edu and rutenbar@ece.cmu.edu

ABSTRACT

We show how recent advances in the handling of correlated interval representations of range uncertainty can be used to predict the impact of statistical manufacturing variations on linear interconnect. We represent correlated statistical variations in RLC parameters as sets of correlated intervals, and show how classical model order reduction methods – AWE and PRIMA – can be re-targeted to compute interval-valued, rather than scalar-valued reductions. By applying a statistical interpretation and sampling to the resulting compact interval-valued model, we can efficiently estimate the impact of variations on the original circuit. Results show the technique can predict mean delay with errors between 5-10%, for correlated RLC parameter variations up to 35%

1. INTRODUCTION

With continued technology scaling, semiconductor designs are increasingly sensitive to random manufacturing variations, and the scale of these unavoidable variations continues to increase relative to the so-called "nominal" outcomes we would prefer. This growing problem is compounded by the fact that different manufacturing steps can affect each die *globally* (perturbing all circuit components in the same way) or *locally* (perturbing spatially nearby clusters of components in the same way), and that these individual variations may be arbitrarily correlated [1].

Against this background, it is perhaps no surprise that static timing analysis is the design step most intensely studied for how it can be enhanced to handle today's mix of correlated variations [2, 3, 4, 5, 6]. First, static timing analysis is still "the" essential sign-off step in many designs; a statistical engine fits neatly into today's design flows. Second – and more relevant to our work in this paper – static timing depends on a very small number of essential statistical "operators". If we can efficiently represent the distribution of arrival times at each input of a gate, we need only be able to compute *sums* and *maximums* of these distributions to propagate a statistical model of worst-case timing through the network. Propagating a set of correlated *normal* delay distributions seems the most successful approach today [3, 4, 5, 6], in large part because the sum and maximum of a set of normal distributions may be calculated (or approximated) by another normal distribution.

The situation is much less satisfactory when we consider other important analysis tasks which require a richer palette of computations to produce a useful result. Consider, for example, interconnect delay analysis. It is well understood that wire delay is a vital component of overall circuit delay, affected in complex ways by manufacturing variations [7]. Model order reduction techniques pioneered in the last decade work remarkably well to approximate complex interconnect with low-order linear models [8, 9, 10]. However, unlike the case with static timing, it is not so easy to see how to represent the essential statistics. For example, it is easy to add two normal distributions; it is *not* easy to extract the dominant eigenvalues from a matrix whose entries are themselves correlated normal distributions, and represent this as yet *another* normal distribution.

Historically, there have been three different avenues of attack on the statistical interconnect modeling problem:

- **Monte Carlo Simulation:** with a sufficiently fast modeling tool, (e.g., RICE [11]), we can randomly sample the space of manufacturing variations and *directly* estimate the distribution for the overall interconnect delay using efficient sampling techniques. This can be very accurate, but very expensive with large designs and a large number of random parameters. We seek a faster method, and are willing to accept some concomitant loss of accuracy.

- **Interval Analysis:** rather than performing the calculations of model order reduction on real-valued scalars, instead use *intervals* on the real line to approximate each statistical variation as a *range* [12]. The idea is appealing, but has yet to be shown to be practical. Interval calculations are notoriously pessimistic, and attempts to date have been restricted to simplistic Elmore-style models that have a small closed-form.

- **Perturbation Theory:** clearly the most successful strategy to date, the idea here is to replace the matrix operations that form the heart of most reduction methods (e.g., for calculating invariant subspaces) with an explicit variational form based on ideas from classical *matrix perturbation theory*. As noted in [13], these ideas have been successfully employed in quantum mechanics for over 50 years. [13] showed how the PACT [9] and PRIMA [10] reduction methods can be rendered in a variational form, and how modest sampling in the space of manufacturing variations yields the data needed to form a set of linear equations whose solution parameterizes these variational models. This technique works with high accuracy on some important problems, e.g., clock tree analysis [7]. However, as presented, the approach can handle only global independent variations.

In this paper, we revisit the interval modeling idea, but show how recent advances in the handling of correlated interval computations provide the key piece of the puzzle missing in prior attempts, and greatly reduce the error of the final calculations. In contrast to [12], we replace the entire numerical "recipe" for AWE- and PRIMA-style model order reductions with interval calculations, and show what changes are needed in the algorithms to make this feasible.

0-7803-8702-3/04/$20.00 ©2004 IEEE.

Finally, borrowing an idea from [13], we also use sampling as an intrinsic part of our reduction method. However, in contrast to [13], our methods produce *interval-valued* transfer functions or eigensystems – interval poles and residues with an explicit correlation structure. Sampling these intervals produces a set of scalar-valued reduced order transfer functions or eigensystems which each approximate one sample of the variational behavior of the overall interconnect. The virtue of the approach is that the final low-order model has much fewer variational terms than the original large interconnect, and can be quickly sampled.

Of course, the technique is not without its own new set of problems. Mis-estimation of interval endpoints and correlations creates a new source of errors, rather like floating point roundoff errors, but more macroscopic. Stability and passivity are similarly problematic, since essentially every calculation is now interval-valued. Pathological combinations of scalar parameters may be easily masked inside interval representations which "usually" – but not always – give useful solutions. Approximating a statistical distribution with a range necessarily results in some loss of accuracy, and requires some assumptions and approximations. And comparing interval-valued quantities for essential decision steps in algorithms creates a wholly new difficulty. We describe these problems, and some empirical solutions, in the sequel.

The paper is organized as follows. Section 2 briefly reviews the correlated intervals model [14] we use to represent correlated statistical variations, its advantages and limitations. Section 3 details our interval-based AWE and PRIMA algorithms. Section 4 shows experimental results, comparing Monte Carlo estimates with estimates from interval-valued delay models. Finally, Section 5 offers concluding remarks.

2. MODELING WITH AFFINE INTERVALS

2.1 Basics of Affine Intervals and Arithmetic

Classical interval arithmetic was invented in the 1960s by Moore [15] to solve range estimation problems in the presence of uncertainties. It has been the subject of extensive investigation, ranging even to complex problems such as linear and nonlinear equations expressed and solved over interval-valued variables [16].

In classical interval analysis, the uncertainty of a variable x is represented by an interval $\overline{x} = [\overline{x}.lo, \overline{x}.hi]$. The true value of x is only known to satisfy $\overline{x}.lo \leqslant x \leqslant \overline{x}.hi$. Basic arithmetic is redefined to yield interval solutions from interval operands, e.g., for addition:

$$\overline{z} = \overline{x} + \overline{y} = [\overline{x}.lo + \overline{y}.lo, \quad \overline{x}.hi + \overline{y}.hi] \tag{1}$$

However, due to the lack of information about operand dependencies, a serious problem is *over-estimation*. To illustrate this, suppose $\overline{x} = [-1, 1]$, $\overline{y} = [-1, 1]$, and that x and y have the relationship as $y = -x$. If we compute $\overline{z} = \overline{x} + \overline{y}$, we can only obtain $\overline{z} = [-2, 2]$, while in reality $z = x + y = 0$. This is a classical example of *range explosion*, when interval computations are pushed through long chains of calculations.

This situation was not improved until a novel range arithmetic model – *affine arithmetic* – was proposed in [14]. In contrast to simple interval models, the affine arithmetic model preserves correlations among variables, in a form analogous to a first-order Taylor series. In this model, the uncertainty of a variable x is represented as a range in an affine form $\widehat{x}$, given by

$$\widehat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n \quad (-1 \leqslant \varepsilon_i \leqslant 1) \tag{2}$$

Each uncertainty symbol ε_i stands for an independent component of the total uncertainties of the variable x; the corresponding coefficient x_i gives the magnitude of that component. We note immediately that, in contrast to traditional interval methods defined by their endpoints, affine intervals are defined by their central point x_0, and a set of symmetric excursions about this point. Still taking addition as an example, if $\widehat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2$ and $\widehat{y} = y_0 + y_1\varepsilon_1 + y_3\varepsilon_3$,

$$\widehat{z} = \widehat{x} + \widehat{y} = (x_0 + y_0) + (x_1 + y_1)\varepsilon_1 + x_2\varepsilon_2 + y_3\varepsilon_3 \tag{3}$$

which is again in an affine form. More importantly, we can see that one symbol ε_i may contribute to the uncertainties of two or more variables, indicating dependence among them. When these variables are combined, uncertainty terms may actually be cancelled.

Returning to the previous example, suppose that x and y have affine forms $\widehat{x} = 0 + 1\varepsilon$ and $\widehat{y} = -\widehat{x} = 0 - 1\varepsilon$. In this case, the affine form of the sum $\widehat{z} = \widehat{x} + \widehat{y} = 0$ perfectly coincides with the actual range of the variable z. This is the unique feature of the model, and its central advantage over earlier interval methods.

Of course, the real problem is when an operation does *not* yield directly an affine result. For example, multiplication of affine intervals $\widehat{x}$ and $\widehat{y}$ gives the product

$$\begin{aligned}\widehat{x}\widehat{y} &= (x_0 + \sum_{i=1}^{n} x_i\varepsilon_i)(y_0 + \sum_{i=1}^{n} y_i\varepsilon_i) \\ &= x_0y_0 + \sum_{i=1}^{n}(y_0x_i + x_0y_i)\varepsilon_i + (\sum_{i=1}^{n} x_i\varepsilon_i)(\sum_{i=1}^{n} y_i\varepsilon_i)\end{aligned} \tag{4}$$

which is *not* in affine form any more, due to the quadratic uncertainty terms $(\sum_{i=1}^{n} x_i\varepsilon_i)(\sum_{i=1}^{n} y_i\varepsilon_i)$. According to [14], we can approximate the quadratic uncertainty terms by

$$(\sum_{i=1}^{n} x_i\varepsilon_i)(\sum_{i=1}^{n} y_i\varepsilon_i) \approx [R(\sum_{i=1}^{n} x_i\varepsilon_i)][R(\sum_{i=1}^{n} y_i\varepsilon_i)]\zeta \tag{5}$$

where ζ is a new uncertainty symbol that is also in $[-1, 1]$, but distinct from all the other uncertainty symbols $(\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n)$ that have already appeared in the same computation. R is the "radius operator", defined as

$$R(\sum_{i=1}^{n} x_i\varepsilon_i) = \sum_{i=1}^{n} |x_i| \tag{6}$$

which computes the upper error bound of an affine variable. Note that the substitution of $[R(\sum_{i=1}^{n} y_i\varepsilon_i)]\zeta$ for the quadratic terms implies a loss of information because ζ is assumed to be independent from $\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n$, while it is in fact a function of them. The result is in affine form again:

$$\begin{aligned}\widehat{x}\widehat{y} \approx & x_0y_0 + \sum_{i=1}^{n}(y_0x_i + x_0y_i)\varepsilon_i \\ & + [R(\sum_{i=1}^{n} x_i\varepsilon_i)][R(\sum_{i=1}^{n} y_i\varepsilon_i)]\zeta\end{aligned} \tag{7}$$

which is a conservative approximation in the sense that it always bounds the original true affine product given by Equation 4.

[14] develops suitable affine approximations for not only multiplication, but also division (assuming the denominator interval does not include '0'), square root (assuming the operand interval is strictly positive), and $exp(\cdot)$. The basic idea for handling non-affine intermediate forms (such as for our multiplication example) is to apply Chebyshev approximation or mid-range approximation theory [14]; roughly speaking, we select an appropriate set of secants of the nominal function curve and manipulate these to parameterize an affine form to minimize the maximum absolute error of the approximation.

2.2 From Intervals to Algorithms

With these basic operations defined for correlated intervals, we can do some remarkably complex numerical operations. A good example is polynomial root-finding. Suppose we have a polynomial over variable λ, with affine interval valued coefficients $\widehat{c_k}$ ($k = 0, 1, 2, \cdots, q$):

$$p(\lambda) = \widehat{c_0} + \widehat{c_1}\lambda + \cdots + \widehat{c_q}\lambda^q \quad (8)$$

We want to extract affine interval-valued roots. Several methods are available for the traditional scalar case, but following advice in [17, 18], we might choose *Laguerre's method*, which requires no explicit derivatives, and can be accomplished with a series of only basic interval arithmetic operations and evaluations of interval absolute value and square root. This method has favorable convergence properties for the scalar case, and is notably robust – it is guaranteed to converge monotonically to a scalar root from an arbitrary starting point. We can use the scalar version as a simple template for an interval-valued version.

for $j = 1, 2, \cdots, q$
  $\widehat{\alpha} = \widehat{c_q}, \widehat{\beta} = 0, \widehat{\gamma} = 0;$
  for $k = q-1, q-2, \cdots, 0$
    $\widehat{\gamma} = \widehat{z_0}\widehat{\gamma} + \widehat{\beta};$
    $\widehat{\beta} = \widehat{z_0}\widehat{\beta} + \widehat{\alpha};$
    $\widehat{\alpha} = \widehat{z_0}\widehat{\alpha} + \widehat{c_k};$
  end
  $\widehat{A} = -\widehat{\beta}/\widehat{\alpha};$
  $\widehat{B} = \widehat{A}^2 - 2\widehat{\gamma}/\widehat{\alpha};$
  $\widehat{C} = [\widehat{A} \pm \sqrt{(q-1)(q\widehat{B} - \widehat{A}^2)}]/q;$
  $\widehat{z_{new}} = \widehat{z_0} + 1/\widehat{C};$
  output $j, \widehat{z_{new}};$
  if $|\widehat{z_{new}} - \widehat{z_0}| < e$ **then stop;**
  $\widehat{z_0} = \widehat{z_{new}};$
end

Figure 1: Laguerre's method.

In Figure 1, variables like $\widehat{c_q}$ are intervals; variables like q are scalars. $\widehat{z_0}$ is an initial approximate root; $\widehat{z_{new}}$ are the interval-valued roots; e is a pre-specified error tolerance. Figure 1 shows the basic iteration, and also lets us highlight another crucial point. Replacing scalar arithmetic operations and function approximations like $\sqrt{(\cdot)}$ with affine intervals is necessary, but not sufficient for most algorithms. We also must deal with *decision* steps which compare values. This presents no problems for the scalar case: either x is bigger than y, or not, and so forth. But such questions are not so well posed for interval-valued operands $\widehat{x}$ and $\widehat{y}$, where specific choices for a scalar element of the range may throw the decision one way or the other.

To confront this, we make a simple, standard choice as in [19]: *we always use the central value of the interval for all decision steps.* Thus, if we need to compare $\widehat{x} = x_0 + \sum_{i=1}^{n} x_i\varepsilon_i$ against $\widehat{y} = x_0 + \sum_{i=1}^{n} y_i\varepsilon_i$, we do this by simply comparing scalar x_0 against scalar y_0. This is always well defined, and has another desirable feature. As noted in [13], it is highly desirable that, when statistical variations are set to zero, a variational analysis algorithm should be able to solve for the nominal case, i.e., to produce an answer that is *identical* to the deterministic solution for the nominal case. By using central values for all interval-valued decisions, we guarantee this useful property.

2.3 From Intervals to Statistics

One final piece of the approach we need to discuss is the assumed relationship between our interval models and the final, statistical models we seek. The affine interval model nicely handles first order linear correlations between variables, captured via shared uncertainty symbols. In their traditional development, these uncertainty symbols ε_i are independent and arbitrarily distributed within $[-1, 1]$, thus creating a set of symmetric excursions about each interval's midpoint. However, what is not present in any of the traditional development of affine arithmetic is any *specific* distribution assumption for these uncertainty terms.

This may seem odd, especially in comparison to recent work on statistical static timing, in which the normal delay distribution assumption is *central* to the computational plan. However, remember that affine intervals are, first and foremost, *intervals* – emphasis is on a conservative approximation to the *range* of the variational results we seek. Any specific probabilistic assumptions we add must be layered on top of this basic model (e.g., as in [20]); they are not part of the core computations as intervals move through each step of a long chain of numerical calculations.

Obviously, the easiest interpretation for each ε_i is as a *uniformly* distributed value within $[-1, 1]$. This is simple, but it is unfortunately not very realistic for modeling manufacturing variations. Hence, we choose the next most obvious interpretation, namely, that each ε_i is a *normal* random variable, with $\mu = 0$ and $\sigma = 1$. Note that, unlike the uniform case, this now means that while our calculations are done on "conservative" ranges, we evaluate any final interval-valued formula by sampling it with values that actually extend *outside* each of these ranges. In other words, we adopt a heuristic interpretation that the uncertainty terms each model the "bulk" of the distribution, between $\pm\sigma$. This is a simple, but surprisingly workable approximation, as we shall see in Section 4.

2.4 Modeling *RLC* Parameter Variations

Manufacturing process variations are random in nature and the true causes are complicated. In general, the variations can be classified into two categories: *global* variation and *local* variation, as in [1, 13]. Global variations, such as critical-dimension (CD) variations, are *inter-die* and can be assumed to affect all the devices and interconnect in a similar way within the same chip. Local variations, such as metal width, metal thickness, and inter-layer-dielectric (ILD) variations are *intra-die* and often exhibit spatial correlations, i.e., the device and interconnect parameters are affected similarly by a common source of variation when these physical elements are close enough to each other. Global variations used to dominate local variations. As semiconductor technology scales and die size grows rapidly, however, local variations are becoming as important as global variations [1].

In this paper, we assume linear combinations of both global and local variations for the interconnect parameters, i.e., resistance (R_i), capacitance (C_i), and inductance (L_i), using the basic affine form introduced earlier:

$$R_i = R_{i,0} + \sum_{j=1}^{l} \Delta R_{i,j}\varepsilon_j + \sum_{j=l+1}^{m} \Delta R_{i,j}\varepsilon_j + \sum_{j=m+1}^{n} \Delta R_{i,j}\varepsilon_j \quad (9)$$

$$C_i = C_{i,0} + \sum_{j=1}^{l} \Delta C_{i,j}\varepsilon_j + \sum_{j=l+1}^{m} \Delta C_{i,j}\varepsilon_j - \sum_{j=m+1}^{n} \Delta C_{i,j}\varepsilon_j \quad (10)$$

$$L_i = L_{i,0} + \sum_{j=1}^{l} \Delta L_{i,j}\varepsilon_j - \sum_{j=l+1}^{n} \Delta L_{i,j}\varepsilon_j - \sum_{j=m+1}^{n} \Delta L_{i,j}\varepsilon_j \quad (11)$$

$R_{i,0}$, $C_{i,0}$, and $L_{i,0}$ are the nominal parameter values. Each unique source of global or local variation is modeled by an uncertainty symbol ε_j. $\Delta R_{i,j}$, $\Delta C_{i,j}$, and $\Delta L_{i,j}$ are the magnitude of the

parameter variation due to a particular ε_j. Any individual source of uncertainty may contribute to more than one parameter and thus lead to direct correlations among these parameters. Correlations can have arbitrary polarity as well. The equations shown above simply emphasize the fact that any uncertainty symbol ε_j can appear with positive or negative proportional impact in any of the linear formulas for any R, C, and L. For example, when metal width increases from its nominal value, the metal resistance is *decreased* while the metal ground capacitance may be *increased*. This is a simple model, but it can capture global and local variations and correlations, and it maps perfectly onto our preferred affine interval model of computation.

3. INTERVAL-VALUED MODEL ORDER REDUCTION

In this section we develop interval-valued versions of the standard AWE [8] and PRIMA [10] model-order reduction algorithms. The overall strategy is easily summarized: we represent variational circuit element values as affine intervals, and then push these interval values through the numerical linear and nonlinear solution steps that comprise each algorithm. The goal – arrived at through different paths in these two algorithms – is a set of interval-valued poles and residues. Our expectation is that the large number of parameters in the original un-reduced circuit will be replaced by a small number of interval-valued coefficients in the final pole/residue form. We statistically sample these final "reduced" intervals to produce an estimate of the distribution of the original circuit's delay.

3.1 Interval-Valued AWE

Before beginning, it is worth commenting on why we include any discussion of AWE, when more stable approaches such as PRIMA already exist. The answer in our case is that the AWE "recipe" was numerically simpler, and thus an easier first target in our development of this approach. Most of the computations in AWE are standard matrix-vector operations, easily handled by basic affine interval arithmetic. The new challenges in AWE are the need for interval-valued LU decomposition for matrix solution, and interval polynomial root-finding. LU decomposition itself is nothing more than a set of matrix-vector loops, with a few critical decision steps, e.g., for pivoting. As we described earlier, all such decision steps on interval quantities simply compare the central values, thus guaranteeing that the central point of the outcome, at least, matches the nominal version of the algorithm. The same is true for the Laguerre-based root finder we described in Section 2.2.

Let us also introduce some consistent terminology. We shall again reserve the '$\widehat{\ }$' symbol for interval-valued quantities. Consider a linear interconnect circuit where the resistance, capacitance, and inductance are subject to both global and local variations with correlations, in the forms of Equations 9–11, and the voltage and current sources are assumed to be scalar, and deterministic. The interval-valued modified-nodal analysis (MNA) formulation in the frequency domain is

$$(\widehat{G} + s\widehat{C})\widehat{x} = b \qquad (12)$$

where the interval-valued conductance matrix $\widehat{G}$, interval-valued susceptance matrix $\widehat{C}$, and scalar-valued excitation vector b are defined as

$$\widehat{G} = \begin{bmatrix} \widehat{N} & E \\ -E^T & 0 \end{bmatrix} \widehat{C} = \begin{bmatrix} \widehat{Q} & 0 \\ 0 & \widehat{H} \end{bmatrix} b = -\begin{bmatrix} i \\ v \end{bmatrix} \qquad (13)$$

$\widehat{N}$, $\widehat{Q}$, and $\widehat{H}$ are the interval-valued matrices containing the stamps for variational resistors, capacitors, and inductors, respectively. E consists of constants ones, negative ones, and zeros, which correspond to the variables of current induced by inductance and voltage sources. The interval-valued solution vector $\widehat{x}$ contains node voltages appended by inductance and voltage source current. The output-selecting vector is denoted as $\widehat{l}$.

Analogous to AWE, interval-valued AWE (or in short, *int*AWE) assumes a moment expansion for $\widehat{x}$ and matches its first $2q$ interval-valued moments by *Padé approximation* where q is the order of model reduction. The $2q$ interval-valued moments are used to construct a qth-order interval-valued transfer function and compute the corresponding interval-valued poles and residues. Figure 2 gives a detailed description of the *int*AWE algorithm.

1. solve $\widehat{G}\widehat{x}_0 = b$ for $\widehat{x}_0$;
2. **for** $k = 1, 2, \cdots, 2q$–1
   solve $\widehat{G}\widehat{x}_k = -\widehat{C}\widehat{x}_{k-1}$ for $\widehat{x}_k$
   end
3. obtain the moments at the output node of interest;
4. set up the Hankel matrix and vector;
5. set the coefficients of the transfer function's denominator polynomial = – (Hankel matrix)$^{-1}$·(Hankel vector);
6. find the roots of the denominator polynomial as poles;
7. set up the Vandemonde matrix;
8. residues = – (Vandemonde matrix)$^{-1}$·(moment vector);

Figure 2: *int*AWE algorithm.

In Figure 2, the interval-valued matrix $\widehat{G}$ is first LU-factorized using a standard Markowitz pivoting scheme, such that the subsequent matrix solves can be conveniently performed by only changing the right-hand-side (RHS) vectors, and then one forward and backward substitution. LU decomposition is also used to find the inverse of an interval-valued matrix column by column, each time given the RHS vector as the column vector of identity matrix. Steps 1, 2, 5, and 8 use the interval-valued LU solve; step 6 uses the interval-valued root-finding from [17, 18]. As in the scalar case, we use *moment shifting*, which only adds some iterative loops to step 2, and *frequency shifting*, which scales interval-valued poles and residues by a scalar constant, to reduce the instability of our results. For clarity, they are omitted in Figure 2.

3.2 Interval-Valued PRIMA

Due to precision loss that occurs during moment calculations and the inherent instability of Padé approximation, AWE may produce inaccurate results or fictitious, unstable poles, even though the original interconnect circuit is actually stable. PRIMA is a projection-based algorithm that is able to generate provably passive/stable reduced-order models in the scalar case. Again, we use a standard version of the PRIMA "recipe" and replace with interval-valued computations. The MNA formulation is the same as that in Section 3.1.

Interval-valued PRIMA (or in short, *int*PRIMA) constructs an orthonormal interval-valued matrix $\widehat{X}$ that spans the interval-valued Krylov subspace defined as

$$Kr(\widehat{A}, \widehat{r}, q) = colsp(\widehat{r}, \widehat{A}\widehat{r}, \widehat{A}^2\widehat{r}, \cdots, \widehat{A}^{q-1}\widehat{r}) \qquad (14)$$

where $\widehat{A} = -\widehat{G}^{-1}\widehat{C}$ and $\widehat{r} = -\widehat{G}^{-1}\widehat{b}$, by successively generating q interval-valued moment vectors $\widehat{x}_k (k = 0, 1, \cdots, q-1)$, like *int*AWE, and filling in the columns of $\widehat{X}$ while maintaining its orthonormality. As a result, the central points of interval quantities in the basis of $\widehat{X}$ are numerically better conditioned than the central points of the interval-valued moment space in *int*AWE. *int*PRIMA

then projects the original variational interconnect system into a reduced-order interval-valued system in the interval-valued Krylov subspace spanned by $\widehat{X}$ via congruence transformation [9]. Finally, the interval-valued poles and residues of the reduced-order model are computed through eigendecomposition. Figure 3 gives the details of *int*PRIMA.

1. solve $\widehat{G}\widehat{x}_0 = b$ for $\widehat{x}_0$;
2. $\widehat{X}^{(0)} = \widehat{x}_0$ and orthonormalize $\widehat{X}^{(0)}$;
3. **for** $k = 1, 2, \cdots, q$-1
   solve $\widehat{G}\widehat{x}_k = -\widehat{C}\widehat{x}_{k-1}$ for $\widehat{x}_k$;
   append $\widehat{x}_k$ to $\widehat{X}^{(k-1)}$;
   orthonormalize $\widehat{X}^{(k)}$;
   end
4. compute $\widehat{G}' = \widehat{X}^T\widehat{G}\widehat{X}$ and $\widehat{C}' = \widehat{X}^T\widehat{C}\widehat{X}$;
   compute $\widehat{b}' = \widehat{X}^T\widehat{b}$ and $\widehat{l}' = \widehat{X}^T\widehat{l}$;
5. let $\widehat{A}' = -\widehat{G}'^{-1}\widehat{C}'$ and eigendecompose $\widehat{A}'$;
6. let the column vectors of $\widehat{S}$ be the eigenvectors;
   let $\widehat{\Lambda} = diag(\widehat{\lambda}_1, \widehat{\lambda}_2, \cdots, \widehat{\lambda}_q)$;
7. invert $\widehat{\lambda}_1, \widehat{\lambda}_2, \cdots, \widehat{\lambda}_q$ to obtain poles;
8. solve $\widehat{G}'\widehat{w} = \widehat{b}'$ for $\widehat{w}$;
9. obtain residues as $-\widehat{S}^T\widehat{l}'\widehat{S}^{-1}\widehat{w}$·(poles);

Figure 3: *int*PRIMA algorithm.

*int*PRIMA again makes heavy use of standard interval-valued matrix-vector operations, orthonormalization, and LU decomposition (see steps 1, 2, 3, 4, 7, 8, and 9 in Figure 3). We implement an interval-valued version of modified Gram-Schmidt (MGS) orthonormalization procedure to orthonormalize $\widehat{x}_k$ to all interval-valued vectors previously filled into $\widehat{X}$. However, the new challenge is the eigendecomposition in step 5. $\widehat{A}'$ is an interval-valued upper Hessenberg matrix, and we implement an interval-valued QR algorithm with implicit shifts to compute its all eigenvalues $(\widehat{\lambda}_1, \widehat{\lambda}_2, \cdots, \widehat{\lambda}_q)$, after balancing $\widehat{A}'$ to improve its numerical condition. All the eigenvectors are then found by inverse iteration. Again, eigendecomposition ends up in practice as a standard sequence of iterative matrix-vector operations, interspersed with some critical decision steps, based on central values. We use as templates the scalar versions of these algorithms from [17, 21].

3.3 Interval-Valued Poles and Residues

Both *int*AWE and *int*PRIMA produce q pairs of interval-valued poles and residues in the following form:

$$\widehat{p}_i = p_{i,0} + \sum_{j=1}^{n} p_{i,j}\varepsilon_j + \sum_{l=1}^{m} p_{i,l}\zeta_l \qquad (15)$$

$$\widehat{k}_i = k_{i,0} + \sum_{j=1}^{n} k_{i,j}\varepsilon_j + \sum_{l=1}^{m} k_{i,l}\zeta_l \qquad (16)$$

where $p_{i,0}$ and $k_{i,0}$ are the same nominal poles and residues as those computed by deterministic AWE and/or PRIMA. ε_j ($j = 1, 2, \cdots, n$) is an uncertainty symbol originally associated with the resistance, capacitance, and inductance in Equations 9–11. ζ_l ($l = 1, 2, \cdots, m$) is a *new* uncertainty symbol created during affine approximations and assumed to be independent of ε_j. Recall that the affine model is fundamentally a *linear* model of correlated range uncertainties. Higher order terms, e.g., the $(\sum_{i=1}^{n} x_i\varepsilon_i)(\sum_{i=1}^{n} y_i\varepsilon_i)$ of the multiplication example are replaced with newly created ζ_l error symbols which capture (in a conservative sense) the range implications, but sever the perfect connection to the problem's original uncertainty terms.

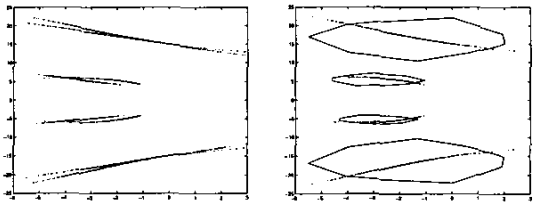
The interplay of the original ε_j and new, linearized ζ_l uncertainty terms is worth examining more closely, since it gives some insights into the advantages and disadvantages of the overall approach. When considering variations in a standard transfer function representation, the root locus is the most traditional way of illustrating the sensitivity of poles/zeros to parameter variation. As a useful illustration, let us consider a synthetic $4th$-order scalar-valued denominator

$$p(s) = c_0 + c_1 s + c_2 s^2 + c_3 s^3 + c_4 s^4 \qquad (17)$$

Now replace c_2 and c_3 with a pair of perfectly correlated affine intervals:

$$p(s) = c_0 + c_1 s + (c_2 + 0.2c_2\varepsilon)s^2 + (c_3 + 0.2c_3\varepsilon)s^3 + c_4 s^4 \qquad (18)$$

ε is a single common uncertainty term shared between the two intervals. Let $c_0 = 1.0000$, $c_1 = 0.1360$, $c_2 = 0.0357$, $c_3 = 0.0008$, $c_4 = 0.0001$, and vary ε with $\mu = 0$ and σ 20% relative to the central values.



(a) (b)

Figure 4: Comparison between scalar-valued root locus and interval-valued root-finding using ε only in (a), and using all uncertainty symbols in (b).

The four curved lines in Figures 4.(a) and (b) show the motion of the four poles in the complex plane if we solve numerically each scalar polynomial we create via this sampling. Next, we replace the coefficients c_2 and c_3 with real affine intervals as in Equation 18, and use the interval version of Laguerre's root finder to create interval-valued roots. The result is a new set of intervals, with not only the original ε term, but with 19 new ζ uncertainties, representing linearized estimates of higher-order interactions. Figure 4.(a) plots the pole motion (see the four curves) predicted if we ignore all but the original single ε value, varying it as before. The result is a simplistic estimate of the real root locus. But, this result fails in our original goal of providing an *interval-valued* solution which bounds the range of final answers. Figure 4.(b) shows the comparison result if we include all of the 19 new ζ uncertainties, sampling them all. We show the convex hull of the set of samples for each pole; a pair of correlated affine intervals by construction define a range bounded by a 2-dimensional central-symmetric convex polytope (see [14]). The good news here is that this result does, to a reasonable extent, bound the real motion of the poles. Unfortunately, this result also highlights the problem with all interval methods: the ranges are not as tight as we might like. Linearized simplifications of the high-order uncertainties are one major source of this discrepancy. Empirically, however, we find the case is rarely as bad as this simple example with perfect correlation between a few terms. In larger problems with more correlations, we usually see some fortuitous cancellation among terms, which helps mitigate the problem.

This simple example also provides insight into why questions of passivity and stability for interval-valued models are so difficult. In

its original form, AWE did not guarantee either passive or stable reduction; PRIMA remedied this difficulty. This problem for the matrix perturbation approach of [13] was resolved in [22, 23] via a careful coupling of the reduced variational model with a fast custom transistor level timing engine. Unfortunately, we cannot yet say anything similarly formal for our interval-valued results. The center points of our resulting intervals do behave as the nominal versions of each algorithm, by construction. But the conservative nature of range bounds, and in particular the difficulty of the linearized higher-order uncertainty approximations, means that any particular sample from these interval ranges may not yield a stable/passive result.

In practice, we use some simple heuristics to remove obviously erroneous results, e.g., unstable poles are simply omitted when they are sampled, as are pole values which differ in sign from their central value. The reason for this rule is more subtle. Affine intervals are, by construction, symmetric about their center points. Conservative range bounds sometime create cases where the overall interval range straddles both the positive and negative real line, even when one or other of these signs for the results is impossible (see [14] for a more careful treatment of the problem). This is another source of error for us in extracting useful interval-valued pole/residue pairs. We note that, while some theory for the stability of interval-valued transfer functions does exist [19] it does not target the more useful correlated affine model. For simple classical intervals, even the decision problem for stability is sometimes unsolvable. Thus, the real question we wish to answer is, empirically, how well the affine intervals yield usable results.

4. EXPERIMENTAL RESULTS

The *int*AWE and *int*PRIMA numerical algorithms, together with the affine arithmetic routines, were implemented in C/C++ and tested using four $RC(L)$ tree-like circuits with a unit step input on a UNIX 1.0GHz machine. The numbers of RLC elements of the four circuits are given in Table 1. We choose a number of uncertainty symbols ranging from 4 to 20 for Equations 9–11. Among these symbols, one is assumed to originate from global variation and is shared by all RLC elements. The rest of the symbols are assumed for local variations and are only shared by a cluster of RLC elements that are "close enough" to one another. In other words, for the case of, say, 4 uncertainty terms, we partition each net-list arbitrarily into 4 groups, and assign the same uncertainty term to every element in the group. Our algorithms and implementation can accommodate any number of global and local uncertainty symbols. Note also that these are only the *initial* sets of uncertainty symbols and more new, distinct symbols will be created during the interval-valued model order reduction process. Furthermore, we assume three combinations for the relative σ of global and local variations: 15%/5%, 5%/15%, and 5%/30%, given in the first columns of Tables 2 and 3.

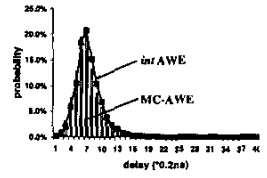
	R	C	L
design1	42	41	40
design2	186	185	185
design3	248	247	247
design4	638	637	0

Table 1: Numbers of R, C, and L elements for the four circuits.

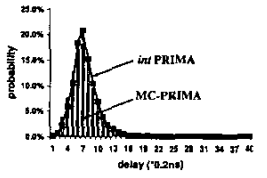
We run $8th$-order *int*AWE and *int*PRIMA to produce interval-valued pole/residue pairs, and then sample the uncertainty terms in these reduced models to create samples of the scalar-valued behavior of the circuit under variation. We analyze each resulting scalar-valued result to find the 50% delay. To measure accuracy, we compare with a MATLAB-based implementation of standard AWE and PRIMA. This allows maximum trust in the various numerical steps in each algorithm, though it clearly penalizes the runtime of the nominal algorithm, compared with a custom C/C++ implementation. For simplicity, we use 10,000 Monte Carlo samples for each combination of design case, global variation, local variation, and number of uncertainty terms (i.e., correlated spatial clusters). We show overall run times, mean and standard deviation (std) of the resulting delay distributions, and comparisons between the "ideal" MATLAB-based Monte Carlo solutions (MC-AWE and MC-PRIMA in Tables 2 and 3, respectively) and our own *int*AWE and *int*PRIMA approaches.

In columns 9–11 of Tables 2 and 3, the run-time speed-up, and the errors of mean delay and standard deviation are defined as the difference between the result of our interval-valued approach and that of Monte Carlo simulation, normalized to the latter. It is easy to see that both the mean delay and standard deviation errors for all the experiment are less than 10%, even with variations of up to 35%. On average, the mean delay error is 4.1% for *int*AWE and 4.9% for *int*PRIMA. Empirically all the mean delay errors are positive; *int*AWE and *int*PRIMA are providing conservative approximations. On average, the standard deviation errors are 5.6% and 5.9% for *int*AWE and *int*PRIMA, respectively.

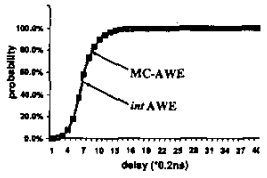
To pictorially demonstrate the accuracy of our interval-valued approaches, we plot the probability distribution function (PDF) and cumulative distribution function (CDF) for design4 with 5% global variation, 30% local variation, and 16 initial uncertainty symbols. In Figure 5, the PDF and CDF given by our interval-valued approaches match very well with those given by Monte Carlo simulation, even though on average, the estimation errors for design4 are bigger than for other designs. Interestingly enough, the distribution of 50% delay is asymmetrical with a positive skew (mean > median) and a long tail at the far end of distribution histogram (see Figures 5.(a) and 5.(c)), although the parameter uncertainties do follow normal distributions.



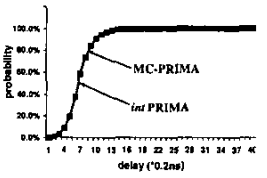
(a) PDF: MC-AWE v.s. *int*AWE



(c) PDF: MC-PRIMA v.s. *int*PRIMA



(b) CDF: MC-AWE v.s. *int*AWE



(d) CDF: MC-PRIMA v.s. *int*PRIMA

Figure 5: Comparison between interval-valued approaches and Monte Carlo simulation for design4.

*int*AWE achieves an average of 18X run-time speed-up over MC-AWE, and *int*PRIMA 19X over MC-PRIMA. The run-time of *int*AWE and *int*PRIMA increases with the number of uncertainty symbols, while the run time of Monte Carlo simulation almost remains the same, because 10,000 vectors were used to sample any number of uncertainty symbols (ranging from 4 to 20).

Two observed trends are worth mentioning here. First, we do not observe more error for the test cases with more uncertainty terms.

1	2	3	4	5	6	7	8	9	10	11
global/local variation	number of ε	MC-AWE			*int*AWE			error		run time speed-up
		mean (ps)	std (ps)	run time (s)	mean (ps)	std (ps)	run time (s)	mean	std	
design1										
15% / 5%	4	271.4	37.8	418	275.8	38.8	23	1.6%	2.6%	17
	8	271.9	37.7	419	275.9	38.6	28	1.5%	2.4%	14
	16	271.7	37.3	419	275.2	38.1	36	1.3%	2.1%	12
5% / 15%	4	272.6	37.7	417	274.0	38.6	22	0.5%	2.7%	18
	8	272.6	37.6	418	273.7	38.5	29	0.4%	2.4%	13
	16	270.5	37.2	419	274.1	38.1	36	1.3%	3.5%	11
5% / 30%	4	270.4	37.9	419	274.7	38.9	22	1.6%	2.6%	18
	8	270.2	37.8	420	274.0	38.8	29	1.4%	2.9%	14
	16	269.3	37.8	422	273.8	38.5	39	1.7%	1.9%	11
design2										
15% / 5%	5	555.6	78.3	3884	588.9	83.9	171	6.0%	7.2%	22
	9	550.6	76.9	3912	575.4	83.2	225	4.5%	8.2%	16
	18	551.9	75.4	3927	579.2	81.6	321	4.9%	8.2%	11
5% / 15%	5	564.1	76.2	3859	582.6	81.0	170	3.3%	6.3%	22
	9	557.3	75.6	3905	584.5	78.8	236	4.9%	4.2%	16
	18	561.4	74.0	3918	576.7	78.4	314	2.7%	5.9%	11
5% / 30%	5	565.7	79.6	3962	587.8	85.6	159	3.9%	7.5%	25
	9	560.0	78.5	3997	579.4	83.7	231	3.3%	6.6%	16
	18	560.8	76.2	3984	581.0	80.5	329	3.6%	5.6%	11
design3										
15% / 5%	5	694.2	98.2	7432	723.2	106.5	305	4.2%	8.5%	23
	10	702.8	97.6	7459	723.5	104.9	392	2.9%	7.5%	18
	20	700.3	97.5	7454	723.9	103.3	509	3.4%	5.9%	14
5% / 15%	5	708.7	97.8	7447	735.3	103.6	303	4.6%	5.9%	24
	10	699.1	96.4	7478	726.7	102.9	388	3.9%	6.7%	18
	20	709.0	95.2	7480	729.1	100.4	519	2.8%	5.5%	13
5% / 30%	5	705.3	99.1	7475	734.0	107.1	302	4.1%	8.1%	24
	10	704.4	98.3	7451	733.2	106.0	381	4.1%	7.8%	19
	20	707.2	98.2	7467	733.8	103.9	508	3.8%	5.8%	14
design4										
15% / 5%	4	1319.4	185.7	14794	1431.0	197.6	512	8.5%	6.6%	28
	8	1320.3	184.2	14815	1424.6	196.5	650	7.9%	6.7%	22
	16	1319.4	182.6	14823	1428.5	194.0	824	8.3%	6.2%	17
5% / 15%	4	1324.0	184.0	14682	1437.7	196.4	507	8.6%	6.7%	28
	8	1314.0	183.5	14797	1435.4	194.9	645	9.2%	6.2%	22
	16	1320.8	182.1	14823	1436.2	192.1	820	8.7%	5.5%	17
5% / 30%	4	1322.6	186.1	14851	1413.2	198.5	506	6.9%	6.7%	28
	8	1322.9	184.8	14942	1425.4	197.3	643	7.7%	6.8%	22
	16	1327.9	183.6	14915	1421.8	195.3	829	7.1%	6.4%	17

Table 2: Comparison between MC-AWE and *int*AWE.

One explanation is that the affine intervals effectively capture the main correlations, and allow for significant empirical cancellation among the uncertainty terms. Second, as expected, we do see more error as the size of the interconnect circuit increases. Just as with basic floating point calculations, a longer chain of calculations produces a less accurate result. The same is true for all interval-based approaches. An important component of ongoing work is trying to pin down more carefully where the most serious sources of mis-estimation occur, and what we might do about them. Nevertheless, we regard the results of Tables 2 and 3 as a very satisfactory outcome for the first *fully* interval-valued implementation of modern model order reduction methods.

5. CONCLUSIONS

The affine interval model, which allows us to represent and preserve first-order correlations among intervals, can be applied to classical model order reduction techniques for linear interconnect, yielding compact, interval-valued models of reduced circuits. A simple statistical interpretation of the resulting intervals allows us to statistically sample these small reduced circuits and estimate the distribution of delay for the original circuit. Preliminary results are extremely encouraging, and suggest that interval-valued models may be able to play a more central role in statistical design and analysis. Our ongoing work focuses on understanding more clearly the current sources of interval error in our approach, the best mapping from intervals to statistics, and the optimal "boundary" between where it is advantageous to compute with intervals, and where it is more efficient to stop, sample the intervals, and continue forward with a set of scalar valued models.

6. ACKNOWLEDGMENTS

We thank Claire Fang, who provided the affine interval arithmetic library and many useful insights about interval mechanics; Saurabh Tiwary, who provided MATLAB versions of AWE and PRIMA for our comparisons; and Larry Pileggi, Xin Li, and Yaping Zhan, for many useful insights into model order reduction methods and process variation. This work was funded in part by C2S2, the MARCO Focus Center for Circuit & System Solutions, under MARCO contract 2003-CT-888.

7. REFERENCES

[1] S. Nassif, "Modeling and analysis of manufacturing variations," in *CICC*, 2001.

[2] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," in *ICCAD*, 2003.

1	2	3	4	5	6	7	8	9	10	11
global/local variation	number of ϵ	MC-PRIMA mean (ps)	MC-PRIMA std (ps)	MC-PRIMA run time (s)	*int*PRIMA mean (ps)	*int*PRIMA std (ps)	*int*PRIMA run time (s)	error mean	error std	run time speed-up
design1										
15% / 5%	4	239.8	32.8	403	243.9	34.4	22	1.7%	4.9%	17
	8	240.6	32.4	405	243.7	34.3	28	1.3%	5.9%	14
	16	239.8	32.1	400	244.1	33.7	38	1.8%	5.0%	10
5% / 15%	4	240.2	32.4	411	244.9	33.9	22	2.0%	4.6%	18
	8	239.4	32.0	404	245.1	33.8	28	2.4%	5.6%	13
	16	240.5	30.9	416	244.9	32.4	39	1.8%	4.9%	10
5% / 30%	4	240.1	33.5	407	245.3	35.0	22	2.2%	4.5%	18
	8	242.0	33.1	410	245.4	35.0	29	1.4%	4.5%	13
	16	240.4	33.1	407	244.9	34.2	38	1.9%	3.3%	10
design2										
15% / 5%	5	561.4	78.0	3387	588.3	82.9	151	4.8%	6.3%	21
	9	554.3	77.5	3420	579.5	81.6	197	4.5%	5.3%	16
	18	560.8	76.2	3411	585.2	80.0	272	4.4%	5.0%	12
5% / 15%	5	569.0	77.6	3405	580.8	82.2	164	2.1%	5.9%	20
	9	562.8	76.9	3391	585.6	81.0	190	4.1%	6.2%	17
	18	563.1	76.0	3402	583.5	80.8	257	3.6%	6.3%	12
5% / 30%	5	556.5	78.3	3402	582.8	83.4	147	4.7%	6.5%	22
	9	554.7	78.0	3424	578.6	82.8	189	4.3%	6.2%	17
	18	554.6	77.5	3418	579.2	81.1	276	4.3%	4.6%	11
design3										
15% / 5%	5	691.5	97.6	6510	723.7	103.4	246	4.7%	5.9%	26
	10	700.2	97.1	6497	723.2	103.1	305	3.3%	6.2%	21
	20	699.4	96.0	6521	724.3	103.1	392	3.6%	7.3%	16
5% / 15%	5	696.9	96.3	6484	732.1	101.9	256	5.1%	5.8%	24
	10	704.3	96.0	6506	734.2	101.8	312	4.2%	6.0%	20
	20	696.2	94.8	6545	731.5	100.1	374	5.1%	6.6%	17
5% / 30%	5	697.7	98.0	6495	730.2	104.2	248	4.7%	6.3%	25
	10	698.1	96.9	6512	725.6	103.5	313	3.9%	6.8%	20
	20	696.0	96.4	6510	726.9	102.9	388	4.4%	6.7%	16
design4										
15% / 5%	4	1321.4	185.2	12538	1418.2	196.0	427	7.3%	5.8%	28
	8	1314.8	184.3	12607	1409.6	195.1	504	7.2%	5.4%	24
	16	1321.0	181.6	12663	1411.8	193.3	611	7.4%	6.4%	20
5% / 15%	4	1319.0	184.1	12475	1422.3	195.0	410	7.8%	5.9%	29
	8	1320.2	183.4	12513	1413.7	193.8	506	7.1%	5.7%	24
	16	1322.7	181.9	12745	1420.4	192.0	602	7.4%	5.6%	20
5% / 30%	4	1315.9	185.2	12600	1400.2	196.9	425	6.4%	6.3%	29
	8	1316.9	184.2	12794	1397.0	195.4	521	6.1%	6.1%	24
	16	1315.6	184.0	12682	1398.5	194.1	601	6.3%	5.5%	20

Table 3: Comparison between MC-PRIMA and *int*PRIMA.

[3] S. Bhardwaj, S. B. Vrudhula, and D. Blaauw, "*T*AU: Timing analysis under uncertainty," in *ICCAD*, 2003.

[4] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *ICCAD*, 2003.

[5] C. Visweswariah, K. Ravindran, and K. Kalafala, "First-order parameterized block-based satistical timing analysis," in *TAU*, 2004.

[6] J. Le, X. Li, and L. T. Pileggi, "STAC: Statistical timing analysis with correlation," in *DAC*, 2004.

[7] Y. Liu, S. R. Nassif, L. T. Pileggi, and A. J. Strojwas, "Impact of interconnect variations on the clock skew of a gigahertz microprocessor," in *DAC*, 2000.

[8] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *TCAD*, 1990.

[9] K. J. Kerns and A. T. Yang, "Stable and efficient reduction of large, multiport RC networks by pole analysis via congruence transformations," *TCAD*, 1997.

[10] A. Odabasioglu, M. Celik, and L. T. Pileggi, "PRIMA: Passive reduced-order interconnect macromodeling algorithm," *TCAD*, 1998.

[11] C. L. Ratzlaff and L. T. Pileggi, "RICE: Rapid interconnect circuit evaluation using AWE," *TCAD*, 1994.

[12] C. L. Harkness and D. P. Lopresti, "Interval methods for modeling uncertainty in RC timing analysis," *TCAD*, 1992.

[13] Y. Liu, L. T. Pileggi, and A. J. Strojwas, "Model order-reduction of RC(L) interconnect including variational analysis," in *DAC*, 1999.

[14] J. Stolfi and L. H. de Figueiredo, *Self-Validated Numerical Methods and Applications*. Brazilian Mathematics Colloquium Monograph, IMPA, Rio De Janeiro, Brazil, 1997.

[15] R. E. Moore, *Interval Analysis*. Prentice-Hall, 1966.

[16] A. Neumaier, *Interval Methods for Systems of Equations*. Cambridge University Press, 1990.

[17] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*. Cambridge University Press, 1992.

[18] D. Kincaid and W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*. Brooks/Cole, 2002.

[19] L. V. Kolev, *Interval Methods for Circuit Analysis*. World Scientific, 1993.

[20] C. F. Fang, R. A. Rutenbar, M. Püschel, and T. Chen, "Toward efficient static analysis of finite precision effects in DSP applications via affine arithmetic modeling," in *DAC*, 2003.

[21] G. H. Golub and C. F. Van Loan, *Matrix Computation*. The Johns Hopkins University Press, 1996.

[22] E. Acar, L. T. Pileggi, and S. R. Nassif, "A linear-centric simulation framework for parametric fluctuations," in *DATE*, 1999.

[23] E. Acar, S. R. Nassif, Y. Liu, and L. T. Pileggi, "Time-domain simulation of variational interconnect models," in *ISQED*, 2002.