Fast Interval-Valued Statistical Interconnect Modeling And Reduction

James D. Ma and Rob A. Rutenbar
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213
jdma@ece.cmu.edu and rutenbar@ece.cmu.edu

ABSTRACT

Correlated interval representations of range uncertainty offer an attractive solution for approximating computations on statistical quantities. The key idea is to use finite intervals to approximate the essential mass of a *pdf* as it moves through numerical operators; the resulting compact interval-valued solution can be easily interpreted as a statistical distribution and efficiently sampled. This paper describes improved interval-valued algorithms for AWE/PRIMA model order reduction for tree-structured interconnect with correlated RLC parameter variations. By moving to a faster interval-valued linear solver based on path-tracing ideas, and making more optimal trade-offs between interval- and scalar-valued computations, we can extract delay statistics roughly 10X faster than a classical Monte Carlo simulation loop, with accuracy to within 5%.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids, Simulation B.8.2 [Performance and Reliability]: Performance Analysis and Design Aids

General Terms: Algorithms, Design, Performance

Keywords: Affine Arithmetic, Interval-Valued Statistical Interconnect Analysis, Manufacturing Variation

1. INTRODUCTION

At 90nm and below, it is no longer realistic to regard device and interconnect parameters as deterministic. The increasingly atomic scale of manufacturing means that all important design parameters are statistically distributed with complex correlations. The new problem is how to *efficiently* analyze critical devices, interconnects, and layouts in this new regime. We see the first practical solutions in recent progress on statistical static timing tools [1, 2, 3, 4, 5]. By representing all circuit delays and arrival times as correlated Gaussian distributions, it is possible to perform delay analysis by "pushing" these statistics through the machinery of static timing. The problem is that static timing requires only a very limited palette of basic operations: addition and maximum of two distributions. We cannot apply the statistical ideas successful in this application to other, arbitrary problems; e.g., if our interest is in *interconnect* analysis, these ideas do not tell how to invert a matrix of Gaussians, or find its statistically distributed eigenvalues. Solving such problems for RLC interconnects in particular, and for circuits in general, is a vigorous new research area. Approaches range from perturbation/symbolic methods [6, 7], to control-theoretic [8, 9], to low order analytical forms [10], among others. Despite recent progress, Monte Carlo analysis still seems to be the most widely used – and widely trusted – method. It provides as much accuracy as we may desire – but often at unacceptable cost.

Our interest is to find more general techniques for representing correlated statistics "inside" algorithms. We believe that recent advances in correlated interval representations of range uncertainty [11] offer an attractive solution that is both "trustworthy" and "widely applicable". Our idea [12] is to approximate each statistical distribution as a finite interval, and to use an appropriate algebra of interval arithmetic to replace each conventional scalar calculation in a given numerical algorithm. The essential assumption is that the mechanics of range calculation for each finite interval are a reasonable – though clearly imperfect – approximation of how statistics actually move through the same computations.

In [12] we showed how the complete numerical algorithms for AWE [13] and PRIMA [14] linear model order reduction could be recast in this interval-valued form, and used to predict mean interconnect delay with errors between 5-10% for correlated RLC parameter variations up to 35%. This paper shows how to improve upon these early results to build a new interval-valued model order reduction strategy that is (a) 10X faster than our tools from [12] for the common case of tree-structured interconnect, and (b) accurate to within 5% of classical Monte Carlo simulation-based analysis.

The first key idea is to understand that intervals, like floating point representations, inevitably accrue more errors as they push through deep chains of calculation. Thus, we replace the interval-valued modified-nodal analysis (MNA) formulation and LU decomposition of [12] with interval-valued versions of the standard path-tracing algorithms [15] which significantly reduces both the number of operations for (orthonormalized) moment generation, and the overall estimation error.

The second key idea is to recognize that it is not essential to employ intervals in every step of our algorithms. In [12] we used intervals pervasively – from RLC circuit inputs to pole/residue outputs – to prove the feasibility of our approach. In this paper, we switch from interval-based computation to scalar-valued Monte Carlo sampling right after the original variational system is reduced to a low order model, but before the reduced system requires further nonlinear solves (e.g., root finding and/or eigendecomposition) for poles and residues. The right trade-off between interval- and

ISPD'05, April 3–6, 2005, San Francisco, California, USA.

scalar-valued computation is a new degree of freedom in these algorithms; we show how to exploit this to make the best trade-off between accuracy and efficiency.

The paper is organized as follows. Section 2 briefly reviews the interval model we use to represent correlated statistical variations. Section 3 describes our statistical AWE and PRIMA algorithms based on interval-valued path-tracing. Section 4 shows experimental results comparing our interval methods against conventional Monte Carlo analysis using a fast linear interconnect simulator (RICE [15, 16]) in the loop.

2. MODELING WITH INTERVALS

2.1 Basics of Affine Intervals and Arithmetic

Classical interval arithmetic [17] was invented to solve range estimation problems in the presence of uncertainties. 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 re-defined 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 interval model – *affine arithmetic* – was proposed in [11]. 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 $\widehat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n$ $(-1 \leqslant \varepsilon_i \leqslant 1)$. 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. 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 $\widehat{y} = y_0 + y_1\varepsilon_1 + y_3\varepsilon_3$,

$$\widehat{z} = \widehat{x} + \widehat{y} = (x_0 + y_0) + (x_1 + y_1)\varepsilon_1 + x_2\varepsilon_2 + y_3\varepsilon_3 \tag{2}$$

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

$$\begin{aligned}\widehat{z} &= \widehat{x}\widehat{y} \\ &= (x_0 + \sum_{i=1}^{n} x_i\varepsilon_i)(y_0 + \sum_{i=1}^{n} y_i\varepsilon_i) \\ &= x_0y_0 + \sum_{i=1}^{n} (y_0x_i + x_0y_i)\varepsilon_i + (\sum_{i=1}^{n} x_i\varepsilon_i)(\sum_{i=1}^{n} y_i\varepsilon_i)\end{aligned} \tag{3}$$

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 [11], we can approximate these terms by $[R(\sum_{i=1}^{n} x_i\varepsilon_i)][R(\sum_{i=1}^{n} y_i\varepsilon_i)]\zeta$. 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 $\sum_{i=1}^{n} |x_i|$, which computes the upper error bound of an affine variable. Note that the substitution of $[R(\sum_{i=1}^{n} x_i\varepsilon_i)][R(\sum_{i=1}^{n} y_i\varepsilon_i)]\zeta$ 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:

$$\begin{aligned}\widehat{z} \approx & x_0y_0 + \sum_{i=1}^{n} (y_0x_i + x_0y_i)\varepsilon_i + [R(\sum_{i=1}^{n} x_i\varepsilon_i)][R(\sum_{i=1}^{n} y_i\varepsilon_i)]\zeta \\ = & z_0 + \sum_{i=1}^{n} z_i\varepsilon_i + R(\widehat{x}\widehat{y})\zeta\end{aligned} \tag{4}$$

which is a conservative approximation in the sense that it always bounds the range of the original true product given by Equation 3.

[11] develops workable affine approximations for not only multiplication, but also division, square root, $exp(\cdot)$, $log(\cdot)$, etc. The twin goals of this fundamental work were (a) to create a practical portfolio of interval operators with which to replace scalars in standard numerical codes, and (b) to guarantee conservative bounding for the range uncertainty represented by each affine interval computation. In our work, we use (a), but we need to abandon (b) to move from intervals to statistics.

2.2 From Intervals to Statistics

The essential assumption we make to move from intervals to a practical statistical interpretation is that the mechanics of range calculation for each *finite* interval are a reasonable approximation of how *statistics* actually move through the same computations. We cannot afford the expense necessary to calculate the exact statistics resulting from various arithmetic operations. We can, however, perform such operations fairly efficiently with affine interval representations. But the affine model does not prescribe any particular distribution to the individual uncertainty symbols. To do requires us to extend the model.

The most useful statistical interpretation is to assume that each ε_i is a *normal* random variable with zero mean and unit variance. This means that we have abandoned the idea that any range uncertainty has finite support, but use the interval to capture the *central mass* of an infinite, continuous distribution. In other words, we employ the bounding mechanics of the affine interval model, but interpret any final affine result as a combination of normal random variables.

It is important at this point to make an observation. This is indeed a rather simple probabilistic interpretation for the intervals. Yet, as we shall see, it is a surprisingly successful one. But more generally, this is *not* the only interpretation one can choose. In this paper, we use a statistical interpretation that emphasizes speed over accuracy. It is possible to alter the fundamental calculation and interpretation of the affine uncertainty terms to produce much more accurate distribution estimates – at the cost of additional CPU time. This is not the focus of this paper; see [18] for details.

We do need to mention, however, one important new way in which the work described in this paper diverges from the classical bounding formulation for the affine model, and from the interval-valued methods we implemented in [12]. Refer again to the multiplication example of Equation 3. Observe that any operation other than addition/subtraction will create a new "linearized" uncertainty term ζ to account for the higher-order combinations of uncertainty

symbols we cannot represent in affine form. Large problems need to create and manage a large number of these linearized uncertainty terms, which is inefficient. Can we do this faster?

The answer is "yes", and to do this we take some inspiration from statistical static timing methods [3, 4] which "match variance" across a fixed set of normal random variables as delay is computed gate by gate. Thus, our idea is *not* to create the standard lumped linearized uncertainty ζ, but instead, to distribute its "effect" across the existing uncertainty symbols. So, returning to the multiplication example of Equation 4, we still compute the coefficient $R(\widehat{xy})$ of ζ, but distribute its effect proportionally among the original $\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n$. Specifically,

$$\widehat{z} \approx z_0 + \sum_{i=1}^{n} z_i [1 + \frac{z_i}{\sum_{i=1}^{n} |z_i|} R(\widehat{xy})] \varepsilon_i \quad (5)$$

which is also a conservative approximation that always bounds the true range of $\widehat{z}$.

Furthermore, if $R(\widehat{xy})$ is much less than $\sum_{i=1}^{n} |z_i|$, we can simply truncate the quadratic terms in Equation 3 and obtain the most efficient approximation of affine multiplication as below:

$$\widehat{z} \approx z_0 + \sum_{i=1}^{n} z_i \varepsilon_i \quad (6)$$

In reality, a threshold ratio of $R(\widehat{xy})$ over $\sum_{i=1}^{n} |z_i|$ can be set to determine whether to use Equation 6. Note that this approximation is no longer conservative and does not always bound the true range of $\widehat{z}$. However, since our own statistical interpretation does not require perfect conservatism on the bounds, and abandons the finite support assumption altogether, this turns out to be an attractive trade-off. We have implemented similar simplifications for division and for the other useful nonlinear operations we need to perform on our intervals.

2.3 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 [19, 6]. 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 and thickness 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 [19].

In this paper, we assume linear combinations of both global and local variations for RLC interconnect parameters, using the basic affine form:

$$R_i = R_{i,0} + \sum_{j=1}^{l} \Delta R_{i,j} \varepsilon_j + \sum_{j=l+1}^{m} \Delta R_{i,j} \varepsilon_j + \sum_{j=m+1}^{n} \Delta R_{i,j} \varepsilon_j \quad (7)$$

$$C_i = C_{i,0} + \sum_{j=1}^{l} \Delta C_{i,j} \varepsilon_j + \sum_{j=l+1}^{m} \Delta C_{i,j} \varepsilon_j - \sum_{j=m+1}^{n} \Delta C_{i,j} \varepsilon_j \quad (8)$$

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

$R_{i,0}, C_{i,0}$, 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 linearized parameter variations due to ε_j, a particular source of variation [20]. Any individual source of uncertainty may contribute to more than one parameter and thus lead to direct correlations among these parameters. The equations simply emphasize the fact that any uncertainty symbol ε_j can appear with positive or negative proportional impact in any of the linearized formulas for any R, C, and L. For example, when metal width increases from its nominal value, the metal ground capacitance may be *increased* while the metal resistance is *decreased*. 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. FAST INTERVAL-VALUED MODEL ORDER REDUCTION

In this section we first develop an efficient interval-valued version of standard path-tracing algorithms [15, 16]. Consider a linear interconnect circuit where the resistance, capacitance, and inductance are subject to both global and local variations with correlations and the input driver voltage sources are assumed to be scalar and deterministic. We represent variational circuit element values as affine intervals, in the forms of Equations 7–9, and then replace the entire "recipe" of the path-tracing algorithm by pushing these interval values through the computation steps. The result is a reduced, small set of either interval-valued *moments* (for interval AWE [13]) or interval-valued reduced *system matrices and vector* (for interval PRIMA [14], to be defined later on), compared with the large number of parameters in the original, un-reduced circuits. Then we statistically sample these "reduced" intervals to complete each model order reduction algorithm and produce an estimate of the delay distribution for the original variational interconnect.

3.1 Interval-Valued Path-Tracing for Moment Generation

In [12], we developed an interval-valued AWE based on an MNA formulation for interconnect circuits of general topologies. The interval-valued moments of variational interconnect were computed by interval-valued LU decomposition. In principle, this is equivalent to computing the interval-valued moments through successive DC analyses of the interconnect circuit where capacitors are substituted with interval-valued DC current sources and inductors with interval-valued DC voltage sources [15]. Initially, the input driver is set to a DC source of its final (deterministic) value, all the capacitors to zero-valued current sources, and all the inductors to zero-valued voltage sources. After the first DC analysis, the resultant interval-valued voltages across each capacitor-current source form the first generation of interval-valued capacitor moments, and the interval-valued currents through each inductor-voltage source form the first generation of interval-valued inductor moments. The succeeding generations of interval-valued moments are computed by setting the driver to zero and replacing each capacitor and inductor source with the product of its previous interval-valued moment and respective interval value of capacitance or inductance. Figures 1.(b) and (c) illustrate how to compute the first two generations of interval-valued moments of a simple interconnect circuit example shown in Figure 1.(a).

The success of the interval-valued LU decomposition algorithms that underlie the interval AWE/PRIMA algorithms of [12] speaks well for the general utility of the interval approach. However, LU decomposition is not the preferred approach for one extremely important class of circuits: *tree-structured* circuits. For such circuits, the linear solve can be realized by a much more efficient algorithm:

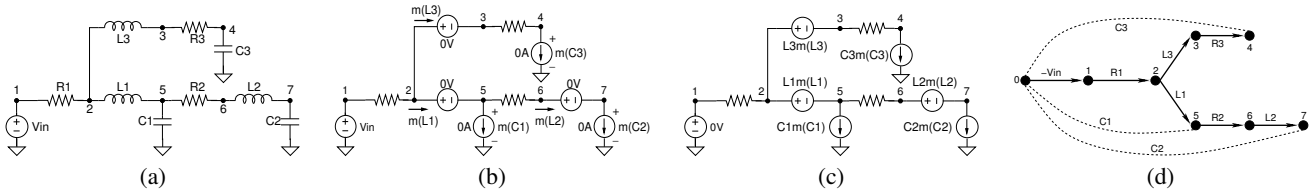


Figure 1: (a) An RLC interconnect circuit example. (b) Equivalent DC circuit for the first moment generation. (c) Equivalent DC circuit with source values changed for the second moment generation. (d) Graph and spanning tree representations for the example.

path-tracing [15]. Indeed, the fastest state-of-the-art AWE/PRIMA implementations (most notably RICE 5 [16]) achieve some of their speed-up by using path-tracing for tree-like interconnects. Our goal is to create an interval-valued counterpart of path-tracing[1].

We define a "tree-like" topology as the following: if a spanning tree $\mathcal{T}$ of the interconnect circuit graph $\mathcal{G}$ can be constructed to include all the voltage sources, resistors, and inductors, and exclude all the capacitors and current sources, then the interconnect circuit is tree-like. $\mathcal{T}$ is defined as a connected sub-graph of $\mathcal{G}$ that contains all nodes of $\mathcal{G}$ with no loops. The ground node is often denoted as the root of $\mathcal{T}$. The edges in $\mathcal{T}$ are denoted as *tree-branches* and all the other edges in $\mathcal{G}$ (not in $\mathcal{T}$) are denoted as *links*. By the definition of a tree-like interconnect circuit, all capacitors and current sources must be links, while all resistors and inductors must be tree-branches. The interconnect circuit shown in Figure 1.(a) is tree-like and its graph and spanning tree representations are given in Figure 1.(d).

Now we are ready to present an interval-valued version of the standard path-tracing algorithm [15]. Once a spanning tree for the interconnect circuit graph is constructed, we start at any leaf node of the tree and perform a reverse depth-first search (DFS) for all the nodes. As each node is visited, its incident interval-valued tree-branch and link currents are summed. This sum is also the total interval-valued current for the tree-branch from the predecessor node. Reverse DFS guarantees that a node is not visited until all the interval-valued currents for all the branches from its descendant nodes are known. The process is terminated at the ground node. For each inductor tree-branch, the resultant interval-valued current is the new interval-valued inductor moment to be used for the next interval-valued moment generation. The interval-valued currents of tree-branch resistors and their corresponding interval resistance values are then used to compute the resistor branch interval-valued voltages.

Next, starting at the ground node, a forward DFS is performed to visit all the nodes. The interval-valued voltage of each node is computed by subtracting the interval-valued voltage of the predecessor tree-branch from the interval-valued voltage of the predecessor node. Forward DFS guarantees that a node is not visited until the interval-valued voltage of its predecessor is known. The interval-valued node voltages are then used to compute the interval-valued voltage of each capacitor, which is the new interval-valued capacitor moment to be used for the next interval-valued moment generation.

The algorithm then proceeds as outlined in the first paragraph of this sub-section. It is stopped when the required number of interval-valued moments is obtained[2]. Note that the topology of an interconnect circuit does not change over successive moment computations. Therefore, a DFS of the spanning tree needs only to be performed once, and the tree-branch/link elements can be "remembered" in the proper computational order, which makes path-tracing very fast. Furthermore, just as in the scalar case, interval-valued path-tracing requires many fewer (interval-valued) calculations than a full LU step, and is thus both faster and more accurate.

3.2 Fast Interval AWE

In [12], we employed affine interval arithmetic *exclusively* – from RLC variational circuit elements all the way through until a final, reduced set of interval valued poles/residues was obtained. Let us revisit the assumption that we must use the interval mechanics exclusively. Consider stopping the interval calculation as soon as we have generated the required $2q+s$ interval-valued moments. At this point, we sample the interval moments, which are themselves affine forms made up of weighted linear sums of zero-mean unit-variance normal random variables. The result is a much larger set of random samples of the distribution of the final $2q+s$ *scalar* moments. Then each set of scalar moment samples is used to construct a qth-order scalar-valued transfer function and compute the corresponding scalar-valued poles and residues, just like a standard AWE Algorithm [13]. The interconnect circuit delay can be obtained via time-domain transient analysis. Enough random samples produce a delay distribution for the variational interconnect. Monte Carlo sampling over moment intervals is quite efficient, since it primarily involves scalar computations for the *reduced* interconnect model. Figure 2 describes the overall fast interval AWE algorithm with interval-valued path-tracing. It will become more clear in Section 3.4 why we push affine interval arithmetic only to interval-valued moments rather than interval-valued poles and residues, a later step in the chain of numerical computations.

3.3 Fast Interval PRIMA

For clarity, we shall reserve the '$\widehat{\ }$' symbol for interval-valued quantities. In [12], a standard version of the PRIMA "recipe" is replaced with interval-valued computations, based on an MNA formulation. Interval-valued LU decomposition is used to construct an orthonormal interval-valued matrix $\widehat{\boldsymbol{X}}$ that spans the interval-valued Krylov subspace defined as

$$Kr(\widehat{\boldsymbol{A}}, \widehat{\boldsymbol{r}}, q) = colsp(\widehat{\boldsymbol{r}}, \widehat{\boldsymbol{A}}\widehat{\boldsymbol{r}}, \widehat{\boldsymbol{A}}^2\widehat{\boldsymbol{r}}, \cdots, \widehat{\boldsymbol{A}}^{q-1}\widehat{\boldsymbol{r}}) \qquad (10)$$

where $\widehat{\boldsymbol{A}}$=$-\widehat{\boldsymbol{G}}^{-1}\widehat{\boldsymbol{C}}$ and $\widehat{\boldsymbol{r}}$=$-\widehat{\boldsymbol{G}}^{-1}\widehat{\boldsymbol{b}}$. Then the original variational interconnect system, characterized by $\widehat{\boldsymbol{G}}$, $\widehat{\boldsymbol{C}}$, and $\boldsymbol{b}$, is projected into a reduced-order interval-valued system in the interval-valued Krylov subspace spanned by $\widehat{\boldsymbol{X}}$ via congruence transformation: $\widehat{\boldsymbol{G}}'$=$\widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{G}}\widehat{\boldsymbol{X}}$, $\widehat{\boldsymbol{C}}'$=$\widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{C}}\widehat{\boldsymbol{X}}$, and $\widehat{\boldsymbol{b}}'$=$\widehat{\boldsymbol{X}}^T\boldsymbol{b}$. Please see [12] for the definitions of $\widehat{\boldsymbol{G}}$, $\widehat{\boldsymbol{C}}$, and $\boldsymbol{b}$.

Again, instead of interval-valued LU factorization, a much faster interval-valued version of the standard path-tracing algorithm in [16] can be used to successively generate the interval-valued moment vectors $\widehat{\boldsymbol{x}}_k$ (k = 0, 1, $\cdots$, q-1), as in Section 3.1, and filling in

[1] [15, 16] also show how to attack non-tree circuits using these ideas; we restrict ourselves here to tree-like circuits.

[2] For AWE, if q is the order of the reduced interconnect model, $2q+s$ moments are needed, where s is the order of *moment shifting*, as in the scalar case [15].

/⋆ interval-valued path-tracing ⋆/
1. **for** $k = 1, 2, \cdots, 2q+s$
   1.1 reverse DFS to compute inductor current moments;
   1.2 forward DFS to compute capacitor voltage moments;
   end
2. obtain the moments at the output node of interest;

/⋆ Monte Carlo sampling of the reduced model ⋆/
3. **repeat:** normally sample moment intervals (μ=0, σ=1);
4. **for** each sample of scalar moments
   4.1 set up the Hankel matrix and vector;
   4.2 set the coefficients of the transfer function's denominator = – (Hankel matrix)$^{-1}$·(Hankel vector);
   4.3 find the roots of the denominator polynomial as poles;
   4.4 set up the Vandemonde matrix;
   4.5 residues = – (Vandemonde matrix)$^{-1}$·(moments);
   4.6 compute the interconnect delay;
   end
   until pre-specified number of samples are generated
5. compute variational interconnect delay distribution;

Figure 2: Fast interval AWE algorithm.

the columns of $\widehat{\boldsymbol{X}}$ while maintaining its orthonormality (by Gram-Schmidt orthonormalization).

Next, unlike [12], we follow the ideas in [16] and show how the algorithm can be augmented, only minimally, to *implicitly* compute the interval-valued reduced system (characterized by $\widehat{\boldsymbol{G}}'$, $\widehat{\boldsymbol{C}}'$, and $\widehat{\boldsymbol{b}}'$) *without* any explicit matrix construction or manipulation. Furthermore, this transformation is performed *concurrently* with interval-valued moment generation and orthonormalization.

For each generation of normalized interval-value moment vector $\widehat{\boldsymbol{x}}_k$ (k = 0, 1, $\cdots$, q-1), every element of $\widehat{\boldsymbol{C}}\widehat{\boldsymbol{x}}_k$ is exactly the product of corresponding normalized interval-valued moment and interval value of capacitance or inductance. According to Section 3.1 and Figure 1, once the present interval-valued moments are computed, the product is *immediately* available for generating the next interval-valued moments. And $\widehat{\boldsymbol{C}}\widehat{\boldsymbol{X}}$= $[\widehat{\boldsymbol{C}}\widehat{\boldsymbol{x}}_0\ \widehat{\boldsymbol{C}}\widehat{\boldsymbol{x}}_1\ \cdots\ \widehat{\boldsymbol{C}}\widehat{\boldsymbol{x}}_{q-1}]$.

To find $\widehat{\boldsymbol{G}}\widehat{\boldsymbol{X}}$, we need some vector manipulation. For Gram-Schmidt orthonormalization,

$$\widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{x}}_{k-1} - \sum_{l=1}^{k-1} \widehat{\boldsymbol{x}}_l(\widehat{\boldsymbol{x}}_l^T \widehat{\boldsymbol{x}}_{k-1}) \quad (k = 1, 2, \cdots, q-1) \quad (11)$$

we multiply interval-valued $\widehat{\boldsymbol{G}}$ to both sides of Equation 11:

$$\widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_{k-1} - \sum_{l=1}^{k-1} \widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_l(\widehat{\boldsymbol{x}}_l^T \widehat{\boldsymbol{x}}_{k-1}) \quad (k = 1, 2, \cdots, q-1) \quad (12)$$

Recall that $\widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_0$=$\boldsymbol{b}$ and $\widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_l$=$-\widehat{\boldsymbol{C}}\widehat{\boldsymbol{x}}_{l-1}$ ($1 \le l \le q$-1), where $\widehat{\boldsymbol{C}}\widehat{\boldsymbol{x}}_{l-1}$ has also been computed previously. So $\widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_k$ can also be obtained. And $\widehat{\boldsymbol{G}}\widehat{\boldsymbol{X}}$=$[\widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_0\ \widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_1\ \cdots\ \widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_{q-1}]$.

Now that we know $\widehat{\boldsymbol{X}}$, $\widehat{\boldsymbol{G}}\widehat{\boldsymbol{X}}$, and $\widehat{\boldsymbol{C}}\widehat{\boldsymbol{X}}$, it is straightforward to construct the interval-valued reduced system. Most of the computations here are matrix-vector operations, easily handled by our basic affine interval arithmetic.

For fast interval PRIMA, we again normally sample each uncertainty symbol (μ=0 and σ=1) of the interval-valued elements of $\widehat{\boldsymbol{G}}'$, $\widehat{\boldsymbol{C}}'$, and $\widehat{\boldsymbol{b}}'$ to produce a scalar-valued reduced system (in terms of $\boldsymbol{G}'$, $\boldsymbol{C}'$, and $\boldsymbol{b}'$). Then a standard version of the PRIMA algorithm [14] can be applied to eigendecompose this scalar system, obtain the poles and residues, and finally analyze the interconnect circuit delay. Figure 3 gives the details of the fast interval PRIMA algorithm.

/⋆ interval-valued path-tracing ⋆/
1. **for** $k = 1, 2, \cdots, 2q+s$
   1.1 reverse DFS to compute inductor current moments;
   1.2 forward DFS to compute capacitor voltage moments;
   1.3 append $\widehat{\boldsymbol{x}}_k$ to $\widehat{\boldsymbol{X}}^{(k-1)}$;
   1.4 orthonormalize $\widehat{\boldsymbol{X}}^{(k)}$;
   1.5 compute $\widehat{\boldsymbol{C}}\widehat{\boldsymbol{x}}_k$ and $\widehat{\boldsymbol{G}}\widehat{\boldsymbol{x}}_k$;
   end
2. compute $\widehat{\boldsymbol{G}}' = \widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{G}}\widehat{\boldsymbol{X}}$ and $\widehat{\boldsymbol{C}}' = \widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{C}}\widehat{\boldsymbol{X}}$;
   compute $\widehat{\boldsymbol{b}}' = \widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{b}}$ and $\widehat{\boldsymbol{l}}' = \widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{l}}$;

/⋆ Monte Carlo sampling of the reduced model ⋆/
3. **repeat:** normally sample $\widehat{\boldsymbol{G}}'$, $\widehat{\boldsymbol{C}}'$, and $\widehat{\boldsymbol{b}}'$ (μ=0, σ=1);
4. **for** each sample of scalar reduced system ($\boldsymbol{G}'$, $\boldsymbol{C}'$, and $\boldsymbol{b}'$)
   4.1 let $\boldsymbol{A}' = -\boldsymbol{G}'^{-1}\boldsymbol{C}'$ and eigendecompose $\boldsymbol{A}'$;
   4.2 let the column vectors of $\boldsymbol{S}$ be $\boldsymbol{A}'$'s eigenvectors; let $\boldsymbol{\Lambda} = diag(\lambda_1, \lambda_2, \cdots, \lambda_q)$;
   4.3 invert $\lambda_1, \lambda_2, \cdots, \lambda_q$ to obtain poles;
   4.4 solve $\boldsymbol{G}'\boldsymbol{w} = \boldsymbol{b}'$ for $\boldsymbol{w}$;
   4.5 obtain residues as $-\boldsymbol{S}^T\boldsymbol{l}'\boldsymbol{S}^{-1}\boldsymbol{w}$·(poles);
   end
   until pre-specified number of samples are generated
5. compute variational interconnect delay distribution;

Figure 3: Fast interval PRIMA algorithm.

3.4 Interval/Scalar Trade-offs

Faster basic interval operations (that avoid new uncertainty terms) and replacing LU decomposition with path-tracing provide much of the speed improvement we shall show in the following section. But the other source of improvement is the use of a *hybrid* interval/scalar strategy. We discuss the motivating trade-offs in more detail here.

First, approximation of interval endpoints and correlations creates errors rather like floating point roundoff errors, but more macroscopic and not so easy to "ignore". Also as with basic floating point calculations, the longer the chain of computation is, the more errors may be created. Therefore, replacing LU decomposition with path-tracing for (orthonormalized) moment computation increases not only efficiency but also accuracy, by greatly reducing the number of operations that need approximate affine interval arithmetic, e.g., multiplication. Replacing scalar-valued LU decomposition by scalar-valued path-tracing does not have this significant accuracy boost as additional considerable benefit.

Second, recall that the affine model is fundamentally a *linear* model defined by each interval's midpoint, and a statistical interpretation on the uncertainty symbols creates a set of symmetric excursions about each interval's central point. A pair of correlated affine intervals by construction define a range bounded by a 2-dimensional "central-symmetric convex polytope" [11]. This does an excellent job for computations dominated by linear operations – additions and subtractions. This may provide a more pessimistic over-estimation on the bound, however, if the computation is highly nonlinear and the real distribution is asymmetric. For example, the root locus of a polynomial with correlated varying coefficients can have too high a degree of curvature for any central-symmetric convex polytope to bound tightly (see [12]). Thus, one idea to consider is replacing the interval-valued computations as soon as the resulting "intermediate" problem representa-

tion is *small* enough that standard, scalar Monte Carlo sampling can be done efficiently. This is especially attractive for AWE/PRIMA-style algorithms, which employ a "front-end" of very large matrix computations, and a "back-end" of more nonlinear operations (root finding, eigendecomposition) on very small, reduced representations of the problem. Thus, in the development of algorithms for fast interval AWE and PRIMA, we made the following trade-offs:

• For fast interval AWE, conduct interval-valued computation to reduce the original large variational system to a much smaller set of interval-valued moments; stop interval-valued computation here, and switch to scalar-valued Monte Carlo sampling over the moments to minimize the errors of affine approximation of nonlinear polynomial root finding for poles (Step 4.3 in Figure 2).

• For fast interval PRIMA, conduct interval-valued computation to reduce the original large variational system to a much smaller set of interval-valued system matrices and vector; stop interval-valued computation here, and switch to scalar-valued Monte Carlo sampling over the reduced system to minimize the errors of affine approximation of nonlinear eigendecomposition (Step 4.1 in Figure 3).

4. EXPERIMENTAL RESULTS

The fast statistical interval-valued algorithms for AWE and PRIMA of Figures 2 and 3 have been implemented as a pair of tools called *stat*AWE and *stat*PRIMA, respectively. The tools, together with the underlying affine arithmetic library, were implemented in C/C++ and benchmarked on a 1.0GHz UNIX machine, using three $RC(L)$ tree-like interconnect circuits. The numbers of RLC elements of the three circuits (design0, design1, and design2) are given in Table 1. It is worth pointing out that design2 is a synthetic un-buffered balanced H-shape clock tree similar to that in [21]. We choose a number of uncertainty symbols ranging from 6 to 21 for Equations 7–9. 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 wire width variations, linearized by first-order Taylor series expansion, 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, 6 uncertainty terms, we partition each net-list into 6 groups, and assign the same uncertainty term to every element in one group. Our algorithms and implementation can accommodate any number of global and local uncertainty symbols that originate from most types of variation sources. Furthermore, we assume three combinations for the relative σ of global and local variations: 20%/10%, 10%/20%, and 5%/30%, given in the first columns of Tables 2 and 3.

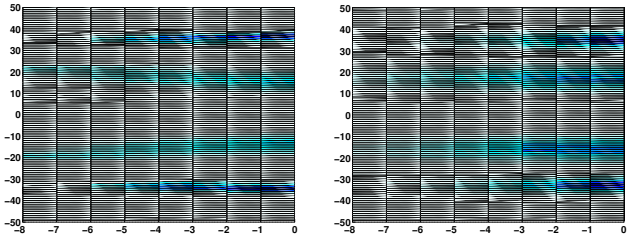
	R	C	L
design0	41	41	40
design1	638	637	0
design2	1121	1120	0

Table 1: Numbers of RLC elements for the three benchmarks.

We find 50% delay distribution of $4th$ order *stat*AWE and *stat*PRIMA for design1 and design2. For comparison of accuracy and speed, we use a fast interconnect simulator based on model order reduction (RICE4 for AWE [15] and RICE5 for PRIMA [16]) in a simple Monte Carlo loop[3]. For fairness and efficiency, we determine how many samples for each tool, parameter setting, and experiment using standard confidence interval methods [22]. We use samples sufficient to guarantee a 99% confidence level with 1% accuracy. (For almost all our experiments, this turns out to be between 3,000 and 3,500 samples.) We show overall run times, mean and standard deviation (std) of the resulting delay distributions, and comparisons between the straightforward Monte Carlo simulation runs (MC-AWE and MC-PRIME in Tables 2 and 3, respectively) and our interval methods, *stat*AWE and *stat*PRIMA.

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 experiments are less than 6%, even with variations of up to 35%. On average, the mean delay error is 1.7% for *stat*AWE and 2.5% for for *stat*PRIMA; the standard deviation error is 1.8% for *stat*AWE and 2.6% for *stat*PRIMA.

Figures 4 and 5 offer visual evidence of the quality of the results of our interval-based algorithms. Figure 4 shows pole density plots for the RLC circuit design0. False color shading indicates greater pole probability with a darker color. Assuming 5% global variation, 30% local variation, and 9 uncertainty terms for the RLC parameters, we compare the variation in the four dominant poles for an $8th$ order AWE reduction for the standard Monte Carlo simulation analysis, and our own *stat*AWE extraction. The correspondence is extremely good. As is obvious from the figure, unstable right half plane "false poles" are pruned in the standard manner [15, 12]. Assessing the stability/passivity of interval valued computation remains a challenging problem, though there are some earlier useful results for the simpler case of classical non-affine intervals [23]. Figure 5 plots the probability density function (*pdf*) and cumulative density function (*cdf*) for design1 with 5% global variation, 30% local variation, and 11 initial uncertainty symbols. The *pdf* and *cdf* given by our interval-valued approaches match very well with those given by Monte Carlo simulation. 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) Pole distribution of MC-AWE (b) Pole distribution of *stat*AWE

Figure 4: Comparison of pole distribution in false color shading between MC-AWE and *stat*AWE for design0. For both figures, X axis is real axis and Y axis is imaginary axis. More poles are distributed in darker regions.

To further demonstrate the trade-offs of our proposed approaches, we compare the following four AWE-style approaches for interval-valued interconnect modeling:

I: *stat*AWE proposed in this paper: interval-valued path-tracing for interval-valued moments and then Monte Carlo sampling;

II: *int*AWE proposed in [12]: LU-decomposition-based interval-

[3]Note that we use the RICE tools in a simple simulator-in-a-loop Monte Carlo experiment, and not the extensions of [6] which can compute variational models directly, for uncorrelated uncertainties.

1	2	3	4	5	6	7	8	9	10	11
global/local variation	number of ε	MC-AWE			*stat*AWE			error		run time
		mean (ps)	std (ps)	run time (s)	mean (ps)	std (ps)	run time (s)	mean	std	speed-up
design1										
20% / 10%	6	1313.2	184.8	196	1347.8	189.8	19	2.6%	2.7%	10
	11	1316.6	185.3	189	1347.2	189.7	18	2.3%	2.4%	11
	21	1314.5	185.0	185	1347.8	189.8	18	2.5%	2.6%	10
10% / 20%	6	1343.1	189.0	197	1346.6	189.6	18	0.3%	0.3%	11
	11	1350.0	190.0	185	1350.4	189.3	17	0.0%	0.4%	11
	21	1352.5	190.3	202	1354.0	189.8	20	0.1%	0.3%	10
5% / 30%	6	1345.3	189.3	199	1352.0	190.3	19	0.5%	0.5%	10
	11	1352.1	190.2	193	1348.8	189.9	18	0.2%	0.2%	11
	21	1358.1	191.1	191	1357.9	190.4	20	0.0%	0.4%	10
design2										
20% / 10%	6	847.4	119.3	302	879.3	123.2	28	3.8%	3.3%	11
	10	845.1	118.9	305	875.0	123.2	29	3.5%	3.6%	11
	19	845.2	118.9	297	878.0	123.6	31	3.9%	4.0%	10
10% / 20%	6	864.1	121.6	304	878.6	123.6	27	1.7%	1.6%	11
	10	866.8	122.0	299	866.3	121.9	27	0.1%	0.1%	11
	19	866.1	121.9	292	870.8	122.5	26	0.5%	0.5%	11
5% / 30%	6	864.1	121.6	301	892.9	125.6	28	3.3%	3.3%	11
	10	869.2	122.3	298	892.5	125.6	27	2.7%	2.7%	11
	19	875.1	123.1	297	867.9	122.1	29	0.8%	0.8%	10

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

1	2	3	4	5	6	7	8	9	10	11
global/local variation	number of ε	MC-PRIMA			*stat*PRIMA			error		run time
		mean (ps)	std (ps)	run time (s)	mean (ps)	std (ps)	run time (s)	mean	std	speed-up
design1										
20% / 10%	6	1307.3	184.0	184	1336.5	188.1	17	2.2%	2.2%	11
	11	1310.3	184.4	178	1348.1	190.1	18	2.9%	3.1%	10
	21	1312.3	184.7	176	1348.6	190.2	17	2.8%	3.0%	10
10% / 20%	6	1269.9	178.7	189	1338.1	188.6	16	5.4%	5.5%	12
	11	1283.7	180.7	175	1341.8	189.2	17	4.5%	4.7%	10
	21	1291.3	181.7	184	1340.5	189.0	19	3.8%	4.0%	10
5% / 30%	6	1290.3	185.1	177	1339.1	188.5	18	3.8%	1.8%	10
	11	1307.1	179.9	180	1298.2	182.3	16	0.7%	1.3%	11
	21	1316.8	181.3	179	1337.0	188.2	17	1.5%	3.8%	11
design2										
20% / 10%	6	845.9	119.0	269	869.9	122.3	25	2.8%	2.8%	11
	10	845.1	118.9	265	870.7	122.6	25	3.0%	3.1%	11
	19	847.1	119.2	267	871.6	122.7	24	2.9%	2.9%	11
10% / 20%	6	863.5	121.5	254	872.4	122.8	22	1.0%	1.1%	12
	10	870.2	122.5	279	861.5	121.2	27	1.0%	1.1%	10
	19	866.5	121.9	260	845.7	119.0	24	2.4%	2.4%	11
5% / 30%	6	864.2	121.6	262	877.0	123.4	23	1.5%	1.5%	11
	10	865.1	121.7	256	874.3	123.0	26	1.1%	1.1%	10
	19	867.8	122.1	263	852.2	120.0	22	1.8%	1.7%	12

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

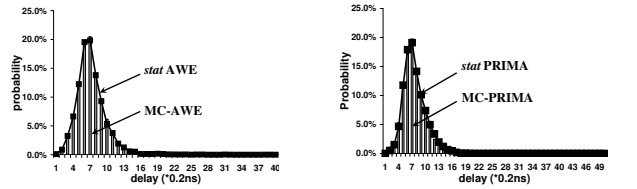
valued computation for interval-valued poles and residues and then Monte Carlo sampling;

III: Path-tracing-based interval-valued computation for interval-valued poles and residues and then Monte Carlo sampling;

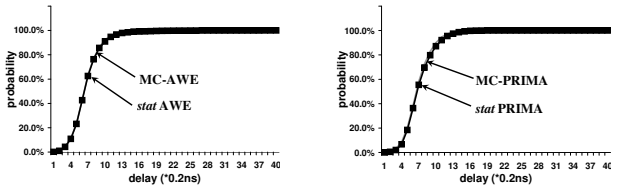
IV: LU-decomposition-based interval-valued computation for interval-valued moments and then Monte Carlo sampling.

We test the above four approaches on the same combinations of global variation, local variation, and number of uncertainty terms for design1 as those in Tables 2 and 3. For approaches I and III, the mean delay and standard deviation errors are with respect to RICE-based Monte Carlo simulation. For approaches II and IV, the errors are with respect to MATLAB-based Monte Carlo simulation, using MNA formulation and LU decomposition. We use the same confidence interval techniques to determine the proper number of samples. Figure 6 plots the results of run time versus mean delay and standard deviation errors for all the approaches. We can see that *stat*AWE provides the best efficiency-accuracy trade-off among all the four approaches: it achieves smaller errors for both mean delay and standard deviation estimation than all the other approaches, while its run time is only slightly longer than that of approach III and much shorter than those of approaches II and IV. Again, as discussed in Section 3.4, this is attributable to (1) many fewer interval operations in interval-valued path-tracing versus interval-valued LU decomposition, and (2) smarter choices on where to switch from interval computation to scalar Monte Carlo sampling – after model order reduction yet before highly nonlinear solves for the reduced system.

With respect to efficiency, our new interval-valued tools are roughly 10X faster than our earlier interval methods from [12], and also roughly 11X faster than the straightforward Monte Carlo simulation experiments conducted using standard confidence interval

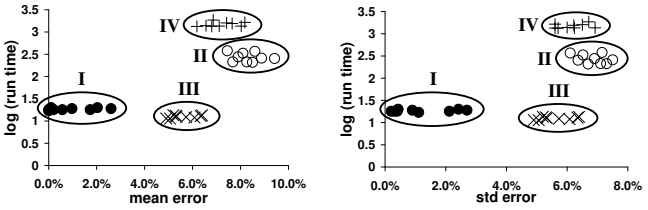


(a) *pdf*: MC-AWE v.s. *stat*AWE (c) *pdf*: MC-PRIMA v.s. *stat*PRIMA



(b) *cdf*: MC-AWE v.s. *stat*AWE (d) *cdf*: MC-PRIMA v.s. *stat*PRIMA

Figure 5: Comparison between interval-valued approaches and Monte Carlo simulation for design1.



(a) Run time v.s. mean error (b) Run time v.s. std error

Figure 6: Comparison among the four approaches in terms of (a) run time and mean delay error and (b) run time and standard deviation error. For both figures, X axis is the error and Y axis is the run time in logarithmic scale.

ideas. Stated differently, a typical interval-valued variational analysis conducted with our tools takes the same time as simulating about 300 different interconnect configurations (samples) in a brute-force manner, with a fast interconnect simulator. We see more speed-up for larger circuits. The run-time for the interval-valued path-tracing component of our tools is less than 5 seconds for all cases; Monte Carlo sampling of the much smaller, interval-valued reduced system, and the subsequent scalar-valued computations on these samples, comprise the majority of the total run time.

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 variational linear interconnect analysis. A simple statistical interpretation of the resulting intervals allows us to statistically sample these small reduced circuits and estimate the delay distribution for the original variational circuit. We have extended our work in [12] by developing more efficient and accurate interval-valued path-tracing algorithms, and have answered a key question posed in [12] on the optimal "boundary" between where it is advantageous to compute with intervals, and where it is more accurate yet still efficient enough to stop and sample the intervals to produce a set of scalar-valued models. Our new algorithms are roughly 10X faster than our first generation efforts of [12], yet still able to achieve accuracies to within 5% of standard Monte Carlo simulation. It is still an open question which, among the many competing strategies being pursued for efficient and accurate variational analysis for circuits and systems will be the winner. Based on the results in this paper, we believe that interval-valued models will find a practical and useful role in this important and evolving new area.

6. ACKNOWLEDGMENTS

At CMU, we thank Claire Fang for many useful insights into affine arithmetic, and Larry Pileggi and Xin Li for providing RICE4/5 and many useful insights into fast model reduction. 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] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," in *ICCAD*, 2003.

[2] S. Bhardwaj, S. B. Vrudhula, and D. Blaauw, "$\mathcal{T}$AU: Timing analysis under uncertainty," in *ICCAD*, 2003.

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

[4] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *DAC*, 2004.

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

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

[7] X. Li, J. Le, P. Gopalakrishman, and L. T. Pileggi, "Asymptotic probability extraction for non-normal distributions of circuit performance," in *ICCAD*, 2004.

[8] J. M. Wang, P. Ghanta, and S. Vrudhula, "Stochastic analysis of interconnect performance in the presence of process variations," in *ICCAD*, 2004.

[9] L. Daniel, O. C. Siong, L. S. Chay, K. H. Lee, and J. White, "Multi-parameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models," *TCAD*, 2004.

[10] K. Agarwal, D. Sylvester, D. Blaauw, F. Liu, S. Nassif, and S. Vrudhula, "Variational delay metrics for interconnect timing analysis," in *DAC*, 2004.

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

[12] J. D. Ma and R. A. Rutenbar, "Interval-valued reduced order statistical interconnect modeling," in *ICCAD*, 2004.

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

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

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

[16] A. Odabasioglu and L. T. Pileggi, "RICE 5: Rapid interconnect circuit evaluation version 5," *Reference Manual*, 1997.

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

[18] C. F. Fang, *Probabilistic Interval-Valued Computation: Representing and Reasoning about Uncertainty in DSP and VLSI Designs*. PhD thesis, CMU, 2005.

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

[20] V. Mehrotra, S. Nassif, D. Boning, and J. Chung, "Modeling the effects of manufacturing variation on high-speed microprocessor interconnect performance," in *IEDM*, 1998.

[21] 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.

[22] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo approach for power estimation," *TVLSI*, 1993.

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