An Accurate Sparse Matrix Based Framework for Statistical Static Timing Analysis

Anand Ramalingam Ashish Kumar Singh Sani R. Nassif
Gi-Joon Nam Michael Orshansky David Z. Pan

Department of Electrical and Computer Engineering, The University of Texas, Austin, TX 78712
Austin Research Laboratory, IBM, Austin, TX 78758
{anandram,asingh,dpan}@cerc.utexas.edu, {nassif,gnam}@us.ibm.com, and orshansky@mail.utexas.edu

ABSTRACT

Statistical Static Timing Analysis has received wide attention recently and emerged as a viable technique for manufacturability analysis. To be useful, however, it is important that the error introduced in SSTA be significantly smaller than the manufacturing variations being modeled. Achieving such accuracy requires careful attention to the delay models and to the algorithms applied. In this paper, we propose a new sparse-matrix based framework for accurate path-based SSTA, motivated by the observation that the number of timing paths in practice is sub-quadratic based on a study of industrial circuits and the ISCAS89 benchmarks. Our sparse-matrix based formulation has the following advantages: (a) It places no restrictions on process parameter distributions; (b) It embeds accurate polynomial-based delay model which takes into account slope propagation naturally; (c) It takes advantage of the matrix sparsity and high performance linear algebra for efficient implementation. Our experimental results are very promising.

1. INTRODUCTION

As technology has scaled, manufacturing variations have emerged as a major limiter of design performance. These variations exhibit themselves as systematic, spatial and random changes in the parameters of active (transistor) and passive (interconnect) components. Furthermore, these variations are increasing with each new generation of technology. Statistical Static Timing Analysis (SSTA) has been proposed to perform full-chip analysis of timing under such types of uncertainty, and has been the subject of intense research recently [1–18]. The result of SSTA is the prediction of parametric yield at a given target performance for a design.

SSTA algorithms can be classified into two major groups:

1. *Block-based* [1–6] approaches use a breadth-first traversal of the circuit to compute circuit delay [1]. The delay pdf is propagated from the primary inputs to the primary outputs. The major difficulty in block-based approaches is the introduction of the max operator at each block, and the need to accurately estimate the maximum of two random variables in the same form in which those two variables are defined.

2. *Path-based* [7–11] approaches rely on an enumeration of all or a large number of the most critical paths in the circuit [7]. Considering the case where all paths are enumerated, the max operator is deferred to the end of the analysis (i.e. taking the maximum of all the paths) and therefore does not introduce any inaccuracy in the computation. A major problem with path-based approaches is the *perception* that typical circuits have an exponential number of paths, making the computational requirement for such approaches impractical.

While there has been much work on the algorithms for SSTA, there has been somewhat less work on the accuracy issues. Some of the sources of inaccuracy in SSTA are: (a) The basic assumptions underlying static timing analysis, such as treating a gate as a node without considering the functionality which gives rise to false paths, (b) The delay models used for gates and wires, and (c) The model for process variations and their spatial and/or temporal distributions.

The *algorithmic* error introduced by SSTA algorithms can be traced back to the application of the max operator, which is an approximation to the behavior of true circuits, and which is further approximated in SSTA algorithms [13–15]. While a direct assessment of that error is difficult, we propose that minimizing the number of max operations would aid in reducing the error. The algorithm we propose in this work reduces the number of max operations to one per circuit.

The *model* error has been widely recognized and a number of researchers have made important contributions. The original *parameterized delay form* expressed delays and arrival times as explicit linear functions of the process parameters [5]. It was later expanded to handle quadratic delay models that are able to improve the accuracy of delay estimates [12–15]. A related source of error, namely the modeling and handling of the slope of signals, has not received as much attention. In fact, current published approaches typically make a worst-case estimate of the slope or propagate the latest arriving slope [5] which can lead to significant error [19]. The polynomial models we propose in this work

ICCAD'06, November 5-9, 2006, San Jose, CA USA

allow high accuracy by using higher order models, and naturally handle the slope and its propagation.

The *distribution* error, i.e. the error caused by lack of generality in the modeling of the statistical properties of the process variations has been the most difficult to deal with, due to the lack of published realistic manufacturing variability data. Earlier approaches assumed that process variables followed normal distributions [7], but recent work has shown how more general distributions can be handled, and how spatial and systematic correlation can be accommodated [18]. In this work, we make *no assumptions* about the character or distribution of any process parameter.

This paper proposes a new approach to parameterized path-based SSTA. The proposed method starts with a preprocessing step of path enumeration and delay computation of all the paths in a parameterized form, which we then efficiently represent using a *sparse matrix*. We model the delay and slope of each component in the circuit using a general parameterized polynomial form which can include the influence of: (a) Input waveforms and output loading, (b) Manufacturing variations in parameters like threshold voltage and channel length, and (c) Operating environment variations in parameters like power supply voltage and temperature. Next, the path delays in this same parameterized form are computed by a natural extension to the gate delay formulation. Given a sample of values from the distribution of manufacturing variations, this computation is shown to be simply a matrix/vector multiply to produce a vector of delays for each path in the circuit. Finally, the maximum circuit delay is obtained by applying the max operator on the path delays. The major attributes of this work are:

1. We show that the number of paths in practice is subquadratic in number of gates by evaluating the number of paths in the ISCAS89 benchmarks as well as two different families of industrial circuits.

2. It can handle global, spatial and intra-die variations in one unified framework.

3. It can compute the delay based on an accurate propagation of slope along all paths.

4. It minimizes the impact of the error caused by approximating the max function commonly used in SSTA.

5. It is independent of the underlying distribution of the process parameters, and is not restricted to the usual Gaussian distribution.

The remainder of this paper is organized as follows. In Section 2 we first motivate the path based approach by showing that it is indeed practical for many circuits. We then show our higher order (more than linear) delay models in Section 3, and describe our approach to the delay modeling of practical static CMOS gates. With those models in hand, we then describe our matrix based formulation for STA and SSTA without slope in Section 4 and SSTA with slope in Section 5. We demonstrate its application to the ISCAS family of sequential benchmark circuits in Section 6 and conclude in Section 7.

2. A CASE FOR PATH BASED SSTA

CAD folklore holds that the number of latch to latch paths in an arbitrary network can be exponential in the number of gates. This is indeed a theoretical upper bound predicted by graph theory. A key observation in this paper, however, is that for the vast majority of practical circuits, the number of actual paths is far less than this theoretical upper bound, and is quite manageable. With the easy availability of large amounts of memory in modern computers, storing and manipulating million of paths is eminently practical.

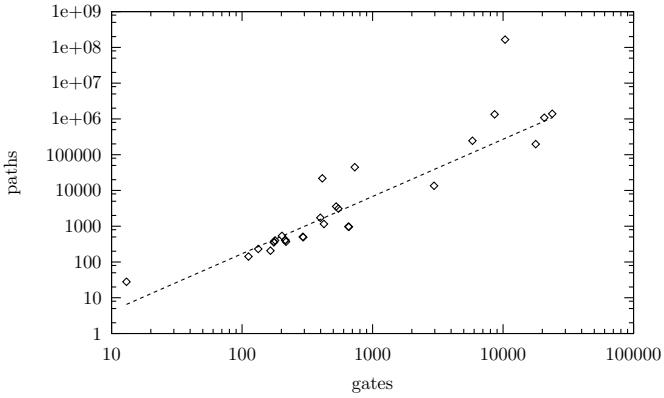

Figure 1: The number of paths versus the number of gates in ISCAS'89 benchmarks. By linear regression we get the following relationship: paths $\approx 0.04 \times \text{gates}^{1.8}$**.**

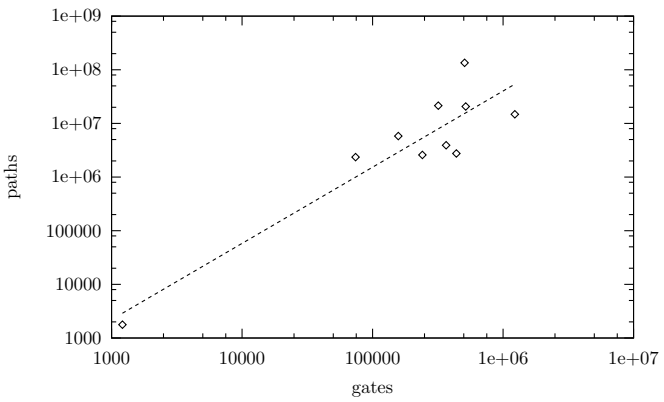

Figure 2: The number of paths versus the number of gates for one family of 10 **industrial benchmarks. By linear regression we get the following relationship:** paths $\approx 0.12 \times \text{gates}^{1.42}$**.**

To test our conjecture, we enumerated *all* the latch to latch, primary input to latch, and latch to primary output paths in the ISCAS sequential circuit benchmarks [20] (see Figure 1), and found that the paths $\approx 0.04 \times \text{gates}^{1.8}$. This is hardly the type of explosive growth that might cause one to completely discount a family of algorithms. But since the ISCAS benchmarks are small compared to modern designs, we further extended our analysis to 2 different families of industrial benchmarks, one for large circuits (much larger than the ISCAS benchmarks), and one for moderate sized circuits (comparable to the ISCAS benchmarks).

We enumerated all paths for the circuits in those two families. For the first and larger family, shown in Figure 2, we

saw that the number of paths $\approx 0.12 \times \text{gates}^{1.42}$. For the second and smaller family, we found paths $\approx 0.43 \times \text{gates}^{1.17}$.

Clearly, the demonstration above should not be taken as sufficient license to propose a purely path-based SSTA algorithm. However, it does demonstrate that such an algorithm can be practical for a significant number of cases. In the broader picture, one can imagine a pairing of path-based and block-based algorithms with one being applied when the enumeration of paths results in a manageable number of paths, while the other gets applied to those circuits where the number of paths exceeds some suitable threshold.

3. PARAMETERIZED GATE DELAY MODELING

The advantage of path-based SSTA is that it can naturally handle accurate nonlinear delay models. In this section, we present a parameterized gate delay model which explicitly takes slope propagation into account. In current published approaches, typically worst-case estimate of the slope or the latest arriving slope is propagated [5] which can lead to significant error [19]. By modeling the input slope in the gate delay equation we avoid this modeling error.

In order to generate the cell delay model for every gate in the library, we simulate each gate varying the process parameters *uniformly* in the range $\mu \pm 3\sigma$ with $3\sigma = 0.2\mu$. The load capacitance C_{L} and input slope S_{in} were also varied. The samples of S_{in} were generated in the range of 10 to 100 ps and samples of C_{L} were generated in the range of 1 to 10 fF. Then the values were fit to the delay equation given below:

$$D = a_0^d + a_1^d L + a_2^d L^2 + a_3^d V_{\text{th}} + a_4^d V_{\text{th}}^2 + C_{\text{L}}\left(b_1^d L + b_2^d L^2 + b_3^d V_{\text{th}} + b_4^d V_{\text{th}}^2\right) + \alpha^d C_{\text{L}} + \beta^d S_{\text{in}} + \gamma^d S_{\text{in}} C_{\text{L}} \quad (1)$$

Similarly, the output slope was also fit to the same canonical form as delay. Note that both the output delay equation Eq. (1) and the output slope equation are explicitly dependent on input slope S_{in}. It should be noted that our formulation does not restrict the model order in any way, and higher order models are possible with no change to our methodology.

4. SPARSE MATRIX BASED SSTA WITHOUT SLOPE PROPAGATION

In this and next sections, we present the sparse-matrix based SSTA formulation. First, we calculate the path delays without considering slope propagation and in the next section we take the slope into account. Let the delay of gate j from input a to the gate output be $d_{j_a} \in \mathbb{R}$. Later we will generalize the gate delay as a function of parameters $\mathbf{z}$, $d_{j_a} = f(\mathbf{z})$.

4.1 Sparse-Matrix Based Static Timing Analysis (STA)

Consider the circuit shown in Figure 3. This circuit has 4 paths and 6 gates. Three of the gates have two inputs which we will denote by a and b. We define an incidence matrix where each row represents a path, and each column represents a gate input. The columns are sorted by gate topological order. The *path-gate* incidence matrix for the example is given by:

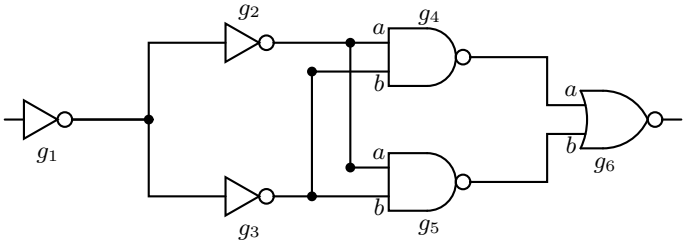

Figure 3: Example circuit for illustrating the matrix formulation. For 2-input gates, the input pins are identified by the labels a and b.

$$\mathbf{A} = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{ccccccccc} g_1 & g_2 & g_3 & g_{4_a} & g_{4_b} & g_{5_a} & g_{5_b} & g_{6_a} & g_{6_b} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \quad (2)$$

Since each path only consists of a small number of gates, matrix $\mathbf{A}$ is a very sparse. The delay of the gates can be written as *gate delay* vector:

$$\mathbf{d}_{\text{gate}} = \begin{bmatrix} d_1 & d_2 & d_3 & d_{4_a} & d_{4_b} & d_{5_a} & d_{5_b} & d_{6_a} & d_{6_b} \end{bmatrix}^\top \quad (3)$$

where d_{4_b} is the delay from input pin b of gate 4 to its output. The delay of a path is given by the addition of gate delays along that path. Thus the path delays is given simply by the multiplication of the path-gate incidence matrix with the gate delay vector:

$$\mathbf{d}_{\text{path}} = \mathbf{A}\mathbf{d}_{\text{gate}} \quad (4)$$

The overall circuit delay is given by the max of all path delays:

$$d_{\text{circuit}} = \max(\mathbf{d}_{\text{path}}) \quad (5)$$

Eq. (5) represents path based Static Timing Analysis (STA). We note that STA in this form is essentially a sparse matrix-vector multiplication, and that it requires only a single max operator to find the circuit delay. There are many data structures and algorithms developed for efficient sparse matrix manipulation which we can exploit [21]. Now we turn our attention to the Statistical STA (SSTA).

4.2 Sparse Matrix based Statistical Static Timing Analysis (SSTA)

In this section, we drop the input specific delay for the sake of convenience. Let the delay of gate j be a function of r parameters $\mathbf{z}_j \in \mathbb{R}^r$. Thus $d_j = f(\mathbf{z})$ is a symbolic function of parameters instead of a real number.

$$d_j = \sum_{k=1}^{r} c_{jk} z_{jk} = \mathbf{c}_j^\top \mathbf{z}_j \quad (6)$$

Note that $\mathbf{z}$ need not consist only of linear parameters. For example, a possible second order gate delay model in channel

length L and load $C_{\rm L}$ might be:

$$\mathbf{z}_j = \begin{bmatrix} 1 & L & L^2 & C_{\rm L} & C_{\rm L}L \end{bmatrix}^\top \tag{7}$$

The same formulation can trivially handle a mixed model such as:

$$\mathbf{z}_j = \begin{bmatrix} 1 & \sqrt{L} & L & C_{\rm L} & C_{\rm L}e^L \end{bmatrix}^\top \tag{8}$$

The gate delays of the circuit in Figure 3 can be written as,

$$\begin{bmatrix} d_1 \\ \dots \\ d_6 \end{bmatrix} = \mathbf{diag}(\mathbf{c}_1^\top, \dots, \mathbf{c}_6^\top) \begin{bmatrix} \mathbf{z}_1 \\ \dots \\ \mathbf{z}_6 \end{bmatrix}$$
$$\mathbf{d}_{\rm gate} = \mathbf{C}^\top \mathbf{Z} \tag{9}$$

The path delays are obtained by multiplying the path-gate incidence matrix in Eq. (2) and the gate delay vector in Eq. (9)

$$\begin{aligned} \mathbf{d}_{\rm path} &= \mathbf{A}\mathbf{d}_{\rm gate} \\ &= \mathbf{A}\mathbf{C}^\top \mathbf{Z} \end{aligned} \tag{10}$$

With Eq. (10) we have now extended the path delay calculation in Eq. (4) to include the dependence of delay on process parameters. Assuming that these process parameters are random variables with some well defined joint probability density function from which we can sample, our goal is to show how we can generalize this result to calculate the distribution of path delays, and by using the traditional max function, the distribution of overall circuit delay.

If the gate delay vector due to kth random sample is given by $\mathbf{d}_{\rm gate}^{(k)}$ then path delay vector in Eq. (10) is given by

$$\mathbf{d}_{\rm path}^{(k)} = \mathbf{A}\mathbf{C}^\top \mathbf{Z}^{(k)} \tag{11}$$

Now if we take ℓ samples then Eq. (11) can be generalized as

$$\begin{bmatrix} \mathbf{d}_{\rm path}^{(1)} & \dots & \mathbf{d}_{\rm path}^{(\ell)} \end{bmatrix} = \mathbf{A}\mathbf{C}^\top \begin{bmatrix} \mathbf{Z}^{(1)} & \dots & \mathbf{Z}^{(\ell)} \end{bmatrix} \tag{12}$$

To get the circuit delay distribution, we apply Eq. (5) to Eq. (12)

$$\begin{bmatrix} d_{\rm circuit}^{(1)} & \dots & d_{\rm circuit}^{(\ell)} \end{bmatrix} = \begin{bmatrix} \max(\mathbf{d}_{\rm path}^{(1)}) & \dots & \max(\mathbf{d}_{\rm path}^{(\ell)}) \end{bmatrix} \tag{13}$$

This is essentially a Monte Carlo simulation expressed in matrix form. A histogram of the circuit delay vector in Eq. (13) produces the circuit delay distribution. Thus Eq. (13) represents path based Statistical Static Timing Analysis (SSTA) ignoring slope. In this form, SSTA is a natural extension of STA as written in Eq. (5) and is simply in the form of a matrix-matrix multiplication. We make a few remarks about the matrices. It is important to note that $\mathbf{A}\mathbf{C}^\top$ matrix is a sparse matrix, which allows for efficient storage as well as fast computation. The $\mathbf{Z}$ vector, though dense, depends only on the number of gates and not on the number of paths.

5. SPARSE MATRIX BASED SSTA WITH SLOPE PROPAGATION

We now extend our delay model to include slope propagation. It is important to note that the output slope of gate j cannot be specified unless we know which path it belongs to. For example, in Figure 3, gate g_4 will have two different slopes namely:

1. s_{14} due to path 1 ($g_1 \rightarrow g_2 \rightarrow g_{4_a} \rightarrow g_{6_a}$), and
2. s_{44} due to path 4 ($g_1 \rightarrow g_3 \rightarrow g_{4_b} \rightarrow g_{6_a}$).

We use the same canonical form to express both delay and slope, but we restrict the dependence of delay and output slope on the input slope to be *linear*. This linearity is required in order to preserve the canonical form as delays are accumulated along a path. We use the superscripts d and s to distinguish among them. We delineate the input slope to a gate by the subscript *in*. The gate delay d_{ij} and the output slope s_{ij} of gate j in path i is given by:

$$d_{ij} = \lambda_j^d s_{\rm in} + \omega_j^d \tag{14}$$
$$s_{ij} = \lambda_j^s s_{\rm in} + \omega_j^s \tag{15}$$

Where the ω terms represent the terms not related to the slope in the canonical form of Eq. (6), i.e. $\omega = \mathbf{c}^\top \mathbf{z}$. From Eq. (14), one can see that the input slope at all the gates is required to calculate the gate and path delays. One way to solve for the input slope is to look at each path p separately and obtain the slope of each gate in an individual path. This method is illustrated using the Figure 4, and this simple circuit consists of inverters which allows us to conveniently drop the input-pin specific subscripts.

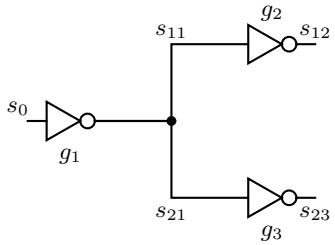

Figure 4: A simple circuit to illustrate SSTA with slope propagation. Here s_0 denotes the slope at the primary input. The output slope at gate g_1 in path 1 is denoted as s_{11} and in path 2 is denoted as s_{21}.

Let $\mathbf{s}_p$ be the column vector in which the values of slopes along path p are listed. Assume the values are listed in the topological order of the gates along path p.

To illustrate, consider the path $p = 1$, through gates g_1 and g_2 in Figure 4. The column vector $\mathbf{s}_1$ is given by

$$\mathbf{s}_1 = \begin{bmatrix} s_0 \\ s_{11} \\ s_{12} \end{bmatrix} \tag{16}$$

and related by Eq.(15) as

$$\begin{bmatrix} s_0 \\ s_{11} \\ s_{12} \end{bmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ \lambda_1^s & 0 & 0 \\ 0 & \lambda_2^s & 0 \end{pmatrix} \begin{bmatrix} s_0 \\ s_{11} \\ s_{12} \end{bmatrix} + \begin{bmatrix} \omega_0^s \\ \omega_1^s \\ \omega_2^s \end{bmatrix}$$
$$\mathbf{s}_1 = \mathbf{\Lambda}_1^s \mathbf{s}_1 + \boldsymbol{\omega}_1^s \tag{17}$$

where $s_0 = \omega_0^s$.

In general Eq. (17) is valid for any arbitrary path containing t gates. Thus, $\mathbf{s}_1 \in \mathbb{R}^{t+1}$, $\mathbf{\Lambda}_1^s \in \mathbb{R}^{(t+1)\times(t+1)}$ is *lower diagonal* matrix, and $\boldsymbol{\omega}_1^s \in \mathbb{R}^{t+1}$. If the circuit has p paths,

then the Eq. (17) for all the p paths can be succinctly captured into one single equation shown below:

$$\begin{bmatrix} \mathbf{s}_1 \\ \dots \\ \mathbf{s}_p \end{bmatrix} = \mathbf{diag}(\mathbf{\Lambda}_1^s, \dots, \mathbf{\Lambda}_p^s) \begin{bmatrix} \mathbf{s}_1 \\ \dots \\ \mathbf{s}_p \end{bmatrix} + \begin{bmatrix} \boldsymbol{\omega}_1^s \\ \dots \\ \boldsymbol{\omega}_p^s \end{bmatrix}$$
$$\mathbf{s} = \mathbf{\Lambda}^s \mathbf{s} + \boldsymbol{\omega}^s \quad (18)$$

From Eq. (18) we can solve for the slope $\mathbf{s}$ in the circuit

$$\mathbf{s} = (\mathbf{I} - \mathbf{\Lambda}^s)^{-1} \boldsymbol{\omega}^s \quad (19)$$

Lemma 1. *The matrix $(\mathbf{I} - \mathbf{\Lambda}^s)^{-1}$ is non-singular. Thus its inverse exists.*

Proof. The proof is omitted due to space constraints. □

The equation for gate delays is similar to Eq. (18) and can be generalized to make the gate delay a function of parameters as in Eq. (6), and Eq. (9),

$$\mathbf{d}_{\text{gate}} = \mathbf{\Lambda}^d \mathbf{s} + \boldsymbol{\omega}^d = \mathbf{\Lambda}^d (\mathbf{I} - \mathbf{\Lambda}^s)^{-1} \boldsymbol{\omega}^s + \boldsymbol{\omega}^d$$
$$= \left(\mathbf{\Lambda}^d (\mathbf{I} - \mathbf{\Lambda}^s)^{-1} (\boldsymbol{C}^s)^\top + (\boldsymbol{C}^d)^\top \right) \mathbf{Z} = \mathbf{D}\mathbf{Z} \quad (20)$$

Once the gate delays are calculated, we can find the path delays using Eq. (11). The circuit delay is given by the max of all path delays. Since delay and slope are a function of process parameters, by taking ℓ samples of process parameters one can generate ℓ samples of circuit delay. A histogram on these samples gives us the circuit delay distribution. Thus SSTA can be performed considering the slope and process variations.

6. EXPERIMENTAL RESULTS

We implemented our algorithm using a combination of awk/perl scripts and C++. We report the results of experiments run on the ISCAS89 benchmarks using a 64-bit Linux machine with 16 GB RAM and running at 3.4 GHz. The delay models were generated using the 90 nm Berkeley Predictive Technology Model [22]. In the experiments only latch-to-latch paths were considered for timing. Thus in Table 1 only the latch-to-latch paths and the number of gates between the latches are listed. We modeled the effect of variations in channel length and threshold voltage, and assumed that the variance of these parameters was such that $3\sigma = 0.2\mu$. We modeled the impact of spatial correlation on parameter variations, and therefore required placement information for the circuits, which we obtained by placing the circuits using Dragon [23]. To properly account for random die-to-die (global) and within-die (intra) variations along with the spatial component mentioned above, we modeled each process parameter $z_{g,i}$ as:

$$z_{g,i} = \sqrt{0.5}\, z_{g,i}^{\text{global}} + \sqrt{0.25}\, z_{g,i}^{\text{intra}} + \sqrt{0.25}\, z_{g,i}^{\text{spatial}} \quad (21)$$

We performed 10000 Monte Carlo simulations for each of the ISCAS benchmark circuits. The results are summarized in Table 1. The table contains the number of gates and paths along with the runtime taken by the algorithm to compute the delay statistics of the circuit. Also shown is the breakdown of effort among (a) Path enumeration (implemented in awk), (b) Sparse matrix generation (implemented in perl), and (c) Matrix multiplication (implemented in C++). Note that the path generation step takes a modest portion of the overall runtime, less than 21% for smaller benchmarks and nearly 5% for bigger benchmarks, while the parameterized path delay generation, which builds the various matrices, and the matrix multiplication take the bulk of the runtime.

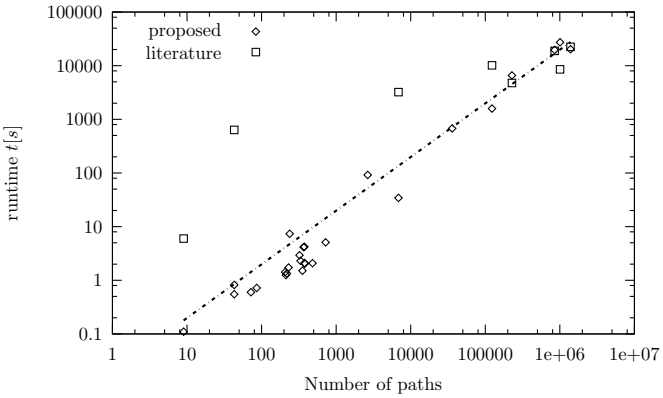

Figure 5: The runtime of the proposed algorithm with respect to the number of paths in the circuit. By linear regression we get the following relationship: runtime $\approx 0.006 \times \text{paths}^{1.09}$. The runtimes for Monte-Carlo simulations of the ISCAS89 benchmarks as reported in [18] are also plotted. The runtimes for bigger benchmarks are comparable.

The runtime versus the number of paths is shown in Figure 5. The relationship between number of paths and runtime is approximately

$$\text{runtime} \approx 0.006 \times \text{paths}^{1.09}$$

which is nearly linear in the number of paths, and shows that our algorithm is scalable. We include in Figure 5 the runtime for Monte-Carlo simulations reported in [18] noting that the machine specification are comparable, and that the number of Monte-Carlo samples is identical. The runtimes are quite close, especially for the larger benchmarks, inspite of (a) the use of a simpler linear delay model, (b) not accounting for slope propagation, and (c) a complete compiled code (C++) implementation in [18].

The number of simulations performed in our experiment (10000) was set high in purpose to establish an accurate result. But a run with one tenth (1000) the number of samples would normally be sufficient to calculate the mean and variance of each circuit delay to engineering accuracy. Furthermore, one can devise an adaptive strategy where non-critical paths non-critical paths are pruned early and skipped from future sample generation and matrix multiplication phases. We believe that we can easily achieve two order of magnitude speedup over the run times quoted, but will defer further discussion to future work.

7. CONCLUSION AND FUTURE WORK

This paper demonstrates that it is possible and practical to perform path based statistical static timing analysis, and that such an analysis can be written compactly in matrix notation, allowing the use of standard highly optimized linear algebra techniques. The major advantage of this formulation is that it places no restrictions on process parameter distributions. It embeds accurate polynomial-based delay model which takes into account slope propagation naturally. With

Table 1: Path-gate statistics of ISCAS89 benchmarks and runtime for 10000 simulations.

circuit	gates	paths	sparsity [%]	runtime [s]				percentage runtime [%]			time per
				generating		matrix	total	generating		matrix	matrix
				paths	matrix	multiply		paths	matrix	multiply	multiply [s]
s27	8	9	46.13	0.02	0.02	0.07	0.11	18	18	63	7.00e-06
s1196	73	43	11.61	0.15	0.07	0.60	0.82	18	8	73	6.00e-05
s1238	73	43	11.61	0.12	0.04	0.39	0.55	21	7	70	3.90e-05
s208	50	72	10.48	0.07	0.04	0.49	0.60	11	6	81	4.90e-05
s386	92	86	8.56	0.08	0.07	0.57	0.72	11	9	79	5.70e-05
s820	187	207	3.42	0.15	0.13	1.14	1.42	10	9	80	1.14e-04
s298	98	212	4.97	0.10	0.11	1.04	1.25	8	8	83	1.04e-04
s832	188	219	3.41	0.15	0.12	1.07	1.34	11	8	79	1.07e-04
s510	162	230	4.02	0.15	0.19	1.39	1.73	8	10	80	1.39e-04
s641	237	238	12.82	0.41	2.14	4.80	7.35	5	29	65	4.80e-04
s344	154	323	6.21	0.17	0.43	2.33	2.93	5	14	79	2.33e-04
s349	155	333	6.11	0.21	0.40	1.70	2.31	9	17	73	1.70e-04
s382	133	353	4.21	0.16	0.17	1.18	1.51	10	11	78	1.18e-04
s1488	307	366	3.36	0.31	0.51	3.33	4.15	7	12	80	3.33e-04
s1494	306	375	3.37	0.36	0.52	3.33	4.21	8	12	79	3.33e-04
s526n	172	377	2.66	0.15	0.20	1.75	2.10	7	9	83	1.75e-04
s526	171	379	2.67	0.15	0.12	1.76	2.03	7	5	86	1.76e-04
s444	160	482	4.18	0.21	0.27	1.60	2.08	10	12	76	1.60e-04
s953	328	723	2.54	0.40	0.76	3.93	5.09	7	14	77	3.93e-04
s713	250	2650	17.54	5.13	36.47	50.42	92.02	5	39	54	5.00e-03
s5378	1938	6858	0.66	4.44	9.62	20.28	34.34	12	28	59	2.00e-03
s1423	566	35990	5.61	37.42	255.02	384.52	676.96	5	37	56	3.80e-02
s35932	14773	122997	0.16	109.93	436.57	1041.31	1587.81	6	27	65	1.03e-01
s9234	5158	227837	0.74	319.25	2030.37	4236.31	6585.93	4	30	64	4.20e-01
s38584	15351	850422	0.25	1121.40	6925.62	11612.28	19659.30	5	35	59	1.15
s13207	7070	1005680	0.63	1575.20	10385.31	15364.06	27324.57	5	38	56	1.52
s38417	21633	1389348	0.13	1349.61	5779.38	13128.55	20257.54	6	28	64	1.30

the exception of the need to have the slope appear linearly, fairly arbitrary models can be trivially handled using this framework.

Data was presented to show that many practical circuits have a bounded number of paths, making such an analysis possible. It should be noted that this demonstration should not be taken as sufficient license to propose a purely path-based SSTA algorithm. For example, the s15850 ISCAS89 benchmark circuit had $\geq 150 \times 10^6$ paths and could not be handled. We plan to explore efficient non-critical path removal techniques to reduce the matrix sizes. In addition, we plan to study further speedup techniques, extend the formulation to handle wires, and show how incremental computation may be done in the framework.

Acknowledgment

This work is partially supported by SRC, IBM Faculty Award, Fujitsu, Sun, and Intel equipment donation.

8. REFERENCES

[1] Jing-Jia Liou, Kwang-Ting Cheng, Sandip Kundu, and Angela Krstic. Fast statistical timing analysis by probabilistic event propagation. In *DAC '01: Proceedings of the 38th conference on Design automation*, pages 661–666, 2001.

[2] Aseem Agarwal, David Blaauw, Vladimir Zolotov, and Sarma Vrudhula. Computation and refinement of statistical bounds on circuit delay. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 348–353, 2003.

[3] Anirudh Devgan and Chandramouli Kashyap. Block-based static timing analysis with uncertainty. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, pages 607–614, 2003.

[4] Hongliang Chang and Sachin S. Sapatnekar. Statistical timing analysis under spatial correlations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(9):1467–1482, 2005.

[5] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 331–336, 2004.

[6] Jiayong Le, Xin Li, and Lawrence T. Pileggi. STAC: statistical timing analysis with correlation. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 343–348, 2004.

[7] Anne E. Gattiker, Sani R. Nassif, Rashmi Dinakar, and Chris Long. Timing yield estimation from static timing analysis. In *ISQED '01: 2nd International Symposium on Quality of Electronic Design*, pages 437–442, 2001.

[8] Jing-Jia Liou, Angela Krstic, Li-C. Wang, and Kwang-Ting Cheng. False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation. In *DAC '02: Proceedings of the 39th conference on Design automation*, pages 566–569, 2002.

[9] Aseem Agarwal, David Blaauw, Vladimir Zolotov, Savithiri Sundareswaran, Min Zhao, Kaushik Gala, and Rajendran Panda. Path-based statistical timing analysis considering inter and intra-die correlations. In *ACM/IEEE International Workshop on Timing Issues*, 2002.

[10] J. A. G. Jess, K. Kalafala, S. R. Naidu, R. H. J. M. Otten, and C. Visweswariah. Statistical timing for parametric yield prediction of digital integrated circuits. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 932–937, 2003.

[11] Michael Orshansky and Arnab Bandyopadhyay. Fast statistical timing analysis handling arbitrary delay correlations. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 337–342, 2004.

[12] Yaping Zhan, Andrzej J. Strojwas, Xin Li, Lawrence T. Pileggi, David Newmark, and Mahesh Sharma. Correlation-aware statistical timing analysis with non-gaussian delay distributions. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 77–82, 2005.

[13] Lizheng Zhang, Weijen Chen, Yuhen Hu, John A. Gubner, and Charlie Chung-Ping Chen. Correlation-preserved non-gaussian statistical timing analysis with quadratic timing model. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 83–88, 2005.

[14] Vishal Khandelwal and Ankur Srivastava. A general framework for accurate statistical timing analysis considering correlations. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 89–94, 2005.

[15] Hongliang Chang, Vladimir Zolotov, Sambasivan Narayan, and Chandu Visweswariah. Parameterized block-based statistical timing analysis with non-gaussian parameters, nonlinear delay functions. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 71–76, 2005.

[16] Khaled R. Heloue and Farid N. Najm. Statistical timing analysis with two-sided constraints. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM international conference on Computer-aided design*, pages 829–836, 2005.

[17] Debjit Sinha and Hai Zhou. A unified framework for statistical timing analysis with coupling and multiple input switching. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM international conference on Computer-aided design*, pages 837–843, 2005.

[18] Jaskirat Singh and Sachin S. Sapatnekar. Statistical timing analysis with correlated non-gaussian parameters using independent component analysis. In *ACM/IEEE International Workshop on Timing Issues*, 2006.

[19] David Blaauw, Vladimir Zolotov, and Savithri Sundareswaran. Slope propagation in static timing analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(10):1180–1192, 2002.

[20] Franc Brglez, David Bryan, and Krzysztof Koźmiński. Combinational profiles of sequential benchmark circuits. In *Proc. of ISCAS*, pages 1929–1934, 1989.

[21] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[22] Yu Cao, Takashi Sato, Michael Orshansky, Dennis Sylvester, and Chenming Hu. New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation. In *Proceedings of Custom Integrated Circuits Conference*, pages 201–204, 2000.

[23] Maogang Wang, Xiaojian Yang, and Majid Sarrafzadeh. Dragon2000: standard-cell placement tool for large industry circuits. In *ICCAD '00: Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 260–263, 2000.