Improvements to Monte-Carlo for Static Timing Analysis
A Survey

Haik Kalantarian
Electrical Engineering
UCLA

Arthur Matsumoto
Electrical Engineering
UCLA

I. INTRODUCTION

In this report, we provide a survey of six different papers that discuss the application of Monte-Carlo analysis to Statistical Static Timing Analysis (SSTA) as well as a method for Static Timing Analysis that incorporates correlation between parameters. These papers provide an overview of the major challenges in reducing the number of required samples through variance reduction techniques. Different methods are used to reduce the number of samples. From deterministic sample generation for use in Quasi-Monte-Carlo analysis, hybrid techniques that improve accuracy by combining different methods to ensure low-discrepancy sequences in multiple dimensions to Markov chains and importance sampling.

II. PRACTICAL, FAST MONTE CARLO STATISTICAL STATIC TIMING ANALYSIS: WHY AND HOW

Monte-Carlo analysis is very useful because it can be applied to almost any kind of problem, and is particularly useful in cases when an ideal analytical solution is not available or computation is too impractical. When the number of iterations for Monte-Carlo analysis increases, the results of the analysis will converge to the ideal analytical solution. One possible area in which Monte-Carlo analysis could be applied is Statistical Static Timing Analysis in integrated circuits. In present Static Timing Analysis algorithms, a linear dependence is assumed between gate delays and various circuit parameters such as channel length variations and threshold voltage variations for gates. Certain more recent works have developed extensions to nonlinear gate models and distributions, but these approaches do not scale well. This class of algorithms is known as “model-based” algorithms.

There are several major advantages for using Monte-Carlo analysis for SSTA. First, it does not require any simplifying assumptions such as the linear dependence of slew rate on processing parameters. Furthermore, the accuracy of this technique is independent of the number of circuit parameters and the algorithm is generally quite scalable. Additionally, these algorithms are parallelizable and well-suited for multicore machines.

Though Monte-Carlo analysis can be used for SSTA, it is instead only used to verify the results of existing techniques. This is because SSTA requires thousands of iterations to obtain an accurate result, and before this scheme can be adopted in STA algorithms, it is necessary to reduce the number of runs.

This can be done using Quasi Monte-Carlo analysis. This is a technique widely used to solve problems in computational finance, and is superior to traditional Monte-Carlo analysis because of its significantly faster convergence rate. This paper aims to prove that using variance-reduction techniques available in Quasi Monte-Carlo analysis, SSTA using Monte-Carlo can become possible.

The variance of a Monte-Carlo solution is as follows:

\[ \hat{\sigma}_{MC}^2 = \frac{\sigma^2(f)}{n} \]

Latin Hypercube sampling is a variance-reduction technique that improves Monte-Carlo analysis by reducing the contribution of the numerator to the variance. This algorithm works by generating n sample points uniformly distributed among the unit cube, and then applies an inverse transform to achieve the required distribution. Assuming \( P_i \) is a random permutation of \{1,2…n\} and \( u_{ij}(i=1…n; \ j = 1…s) \) are n*s independently distributed random variables over [0,1], then the jth coordinate of the ith point in an n-sample LHS is as follows:

\[ x_{ij} = n^{-1}(\pi_j(i) - 1 + u_{ij}), \ i = 1,…,n, \ j = 1,…,s. \]

Other Quasi Monte-Carlo techniques exist to improve Monte-Carlo simulations. Unlike traditional Monte-Carlo simulation which uses pseudo random-number generation, these techniques generate deterministic sequence with a more uniform distribution. The uniformity of a distribution can be represented mathematically through the concept of Star Discrepancy.

The discrepancy of a pseudo-random set is as follows:

\[ D_n \mid_{MC} = O\left(n^{-0.5}(\log \log n)^{-5}\right) \]

On the other hand, some deterministic sequences have the following discrepancies:

\[ D_n \mid_{QMC} = O\left(n^{-1}\log' n\right) \]

Some examples of these sequences include the Halton Set, Hammersley sequence, Sobol sequence, and Faur techniques.
The picture on the left shows the distribution of a pseudo-random sequence, while the picture on the right is a deterministic sequence with more uniformity that can achieve better results than traditional Monte-Carlo with fewer iterations.

Traditional Random Monte-Carlo methods estimate their errors by computing the results with the same number of iterations multiple times and comparing the deviations. This is not possible with deterministic sequences, because they provide the same input sequence every time. However, by scrambling the digits of the coordinate values of these sequences, one can also evaluate the error in the estimate by seeing how the estimation changes during different simulations.

The picture below shows the average number of points needed to obtain a 95.45% confidence interval using three techniques: Random Monte-Carlo, Latin Hypercube, and Quasi Monte-Carlo analysis. As we can see, variance reduction techniques can significantly decrease the number of iterations necessary, making Monte-Carlo a valid technique for STA.

The conclusion of this paper is that SSTA using Quasi Monte-Carlo analysis is a valid and practical technique, and that <5% error with 95.45% confidence is achievable with an average of 341 iterations, with runtimes ranging from 2 seconds to 90 seconds using very large benchmarks. Furthermore, these techniques are superior to existing ones because they do not make simplifying assumptions about gate models of wire models, and are highly scalable.

III. FAST STATISTICAL STATIC TIMING ANALYSIS USING SMART MONTE CARLO TECHNIQUES

As mentioned earlier, Monte-Carlo can be used in SSTA. However, it requires many samples and in some cases the number of samples can be impractical. This concern has led to various variance-reduction techniques and quasi-monte-carlo analysis. This paper identifies the main variance reduction techniques as: Latin hypercube sampling, stratified sampling, importance sampling, and control variates. One of the problems with sequences constructed to have a low discrepancy (known as LDSs) is known as pattern dependence. Many techniques generate a deterministic sequence to reduce the discrepancy of the set and provide more uniformity. However, LDSs are imperfect, and the uniformity degrades as the number of dimensions increase, which is particularly noticeable among higher coordinates of LDSs. Therefore, one method to mitigate this problem is to identify important and unimportant variables for monte-carlo analysis. Then, the lower-coordinates which have more uniformity can be assigned to the important variables, while the higher coordinates which have more variation can be assigned to the less important variables.

One technique that this paper describes is stratified sampling. This technique partitions the sample space into mutually exclusive strata, and then samples using any variance reduction technique within each strata. Essentially, a variable can be divided into several different bins of equal probability. As a result, timing behavior within strata would exhibit lower variation and be easier to estimate. This paper then describes the application of Latin Hypercube Sampling to SSTA and how the hypercube is constructed, which has already been discussed. Next, this paper proposes a process-variation model which accounts for spatially correlated variation by dividing the die into n x n grids, and assuming identical parameter variations within a grid. Therefore, each source of variation is represented by a set of variables for all grids.

The main contribution of this paper is SH-QMC: Stratification+Hybrid Quasi Monte-Carlo. In this approach, circuit timing criticality information is used to reduce the MC sample size. Previously, variables are ordered based on their criticality to circuit delay using a parameter $P_{crit}$. QMC, stratified sampling, and LHS are then applied to variables based on their convergence property. Due to the pattern dependencies exhibited by QMC described earlier, only a very limited number of variables are sampled using QMC, while Latin hypercube is used on non-critical variables which does not exhibit this problem but has a slower convergence.

The figure below shows how increasing the number of variables mapped to QMC can affect the 95% error when compared to a “golden” monte-carlo analysis with 40,000 samples which has the exact solution.
The main idea is that process variables are ordered based on importance (impact on timing). Based on the importance of these variables, QMC, LHS, and stratified sampling can be used for timing analysis. Because QMC has fast convergence but exhibits pattern behavior, they are used on only the most critical variables, and LHS can be used for the remaining variables. The two primary ways that variables can be ordered based on their criticality are nominal ordering and learning-based ordering. In nominal ordering, information from STA performed at the nominal process corner is used to order variables based on the timing criticality $P_{crit}$. The learning-based method is quite similar, but uses a subset of QMC samples to obtain more information about the statistical behavior of the circuit, which is used to obtain the boundaries of the probability distribution of that node’s timing slack. In other words, a minimum and maximum value is probabilistically obtained based on a few select samples.

The last contribution of this paper is about incremental evaluation of percentile delay. Static timing synthesis tools require efficient recomputation based on small changes of the design, which Monte-Carlo analysis does not deal well with. Naturally, it is inefficient to do an entirely new simulation on a circuit which has been changed just incrementally, so the authors of this paper propose a scheme in which incremental recomputation can be applied to SH-QMC. This is based on the idea that if we have two circuits C and C’, the Monte-Carlo samples do not need to be regenerated and can be reused for both circuits, meaning most samples do not need to be recomputed to calculate the delay. The only samples that must be calculated to compute the delay are the ones that have an arrival time which is close to the percentile delay of the circuit C which must be reevaluated. The basic idea is that if a circuit is on the very threshold of meeting timing constraints, it is not necessary to re-assess inputs in which the circuit easily met the constraints. Instead, the only points of interest are the ones surrounding the constraint threshold. This technique can save significant amounts of time and only requires reevaluating a select few samples.

The graph below shows that SH-QMC can provide a very close estimate to the golden circuit in its probability distribution, compared to traditional SSTA.

The graph below shows that SH-QMC has a dramatically reduced runtime compared to traditional SSTA, especially with larger grid-sizes where it scales significantly better.

IV. A ON EFFICIENT MONTE-CARLO BASED STATISTICAL TIMING ANALYSIS OF DIGITAL CIRCUITS

This paper attempts to improve upon the discrepancy of various Monte-Carlo techniques for static timing analysis, and proposes an algorithm which can improve the precision of delay estimations compared to Monte-Carlo, Quasi Monte-Carlo, and Latin Hypercube sampling. This paper first identifies the main problem with current SSTA (Statistical Static Timing Analysis) techniques: current techniques simplify or even ignore certain process parameters for simplification. Examples of these effects are the nonlinear relationships between gate delays and process parameters, nonzero skew of signal arrive times, and simplified interconnect delay models. This has led to the usage of Monte-Carlo techniques for SSTA, which can account for this complexity. This paper then provides a review of deterministic sequence generation, Latin Hypercube Sampling, and Quasi-Monte Carlo analysis which is already described in this survey.

To evaluate the effective dimension of the statistical mean and standard deviation of a circuit’s delay, a technique is used which uses shifted Legendre polynomial functions as orthogonal functional bases for integrand decomposition. The decomposition of function $f(x)$ is therefore:

$$f(x) = \sum_{r_1=0}^{\infty} \cdots \sum_{r_d=0}^{\infty} \prod_{j=1}^{d} C_r \phi_{r_j}(x(j))$$

where:

$$\phi_n(x) = \left[ \int_0^1 p_n^2 (2x-1) \, dx \right]^{-0.5}$$

$P_n(2x-1)$ is the n-order shifted and scaled polynomial. $C_r$ is simply a constant which is equal to:

$$C_r = \frac{1}{N(N-1)} \left( \sum_{k=1}^{N} f(x_k) \prod_{j=1}^{d} \phi_{r_j}(x_k(j))^2 \right) - \frac{1}{N} \sum_{k=1}^{N} f(x_k) \prod_{j=1}^{d} \phi_{r_j}^2(x_k(j))$$
The effective dimensions of benchmark circuit critical delays can therefore be estimated using this numerical method. Based on this technique, one can prove that Latin-Hypercube Sampling techniques show a large amount of uniformity in one dimension. However, this technique does not show superior 2D uniformity compared to Monte-Carlo analysis as shown below:

The figure below shows the 2D uniformity using Monte-Carlo, Latin Hypercube, CALHS, and QMC (Sobol) sampling. Note the lack of uniformity in the LHS.

Furthermore, the graph below shows the lack of uniformity when using Sobol (QMC) sampling in higher dimensions. This is a result of bad 2-D pairing for high-dimensional sequences.

![Graph showing 2D uniformity](image)

The next contribution of this paper was a modification to the Sobol algorithm which generates sequences with as many good pairings as possible to eliminate the problem depicted above. Furthermore, it is proven that these bad pairings can be detected in advance.

This paper first describes how to generate a Sobol Sequence of N samples and D dimensions.

\[ x_i^{(j)} = a_1 v_i^{(j)} \oplus a_2 v_{i+1}^{(j)} \oplus \cdots \oplus a_W v_{i+W}^{(j)} \]

The coefficients \( A1...Aw \) are generated from the Grays Code representation of i. For each dimension j, a series of integers \( M_k(j) \) is defined by the following q-term recurrence relation:

\[ m_i^{(j)} = 2b_1^{(j)} m_{i-1}^{(j)} \oplus 2^2 b_2^{(j)} m_{i-2}^{(j)} \oplus \cdots \oplus 2^{q-1} b_{q-1}^{(j)} m_{i-q+1}^{(j)} \oplus (2^{q} m_{i-q}^{(j)} \oplus m_{i-q}^{(j)}) \]

Where \( b_k^{(j)} \in \{0,1\} \), \( k = 1, \ldots, q - 1 \) being the coefficients for the q-degree polynomial specified for each dimension. Though the details of this sequence generation are quite complicated, the key idea is that a great amount of flexibility exists for choosing the initial values for \( M_k(j) \) and the remaining value for \( M(q) \) are generated from the equation listed above. To reduce the lack of uniformity in the 2D pairings described earlier, either more samples are needed or the initial points must be modified. The key goal then is to pick the appropriate initial values to minimize the number of bad pairings. This is done using a Simulated-Annealing algorithm, which minimizes bad pairings by flipping selected bits in the initial values. The figure below shows how modifying these initial value scan reduce the discrepancy significantly. In this figure, the top row is the distribution of \( L2 \) discrepancies using initial values, while the bottom row is the same using optimized initial values.
The main contribution of this paper is the proposal of a SSTA algorithm based on Monte-Carlo analysis. This technique uses Latin Hypercube Sampling because of its low-discrepancy in one dimension, and uses a Sobol sequence for higher dimensions. By modifying the initial values used in the Sobol generator, the number of bad pairings are significantly reduced and the 2D uniformity is increased.

Given that N, the number of samples, is \(N = (2^W) - 1\), a total of \(2^{(W-1)}\) samples would use Sobol sequences, while the remaining samples would use LHS samples. As a further optimization, parameters are assigned a criticality based on the following formula:

\[
c_t = \sum_{j=1}^{P} \psi_{i,j} \sum_{k=1}^{N_j} \exp \left\{ \alpha \cdot \left( \frac{\text{Slack}_j}{D_{\text{nom}}_j} \right)^2 \right\}
\]

Where \(\psi_{i,j}\) is the coefficient of the jth PC in the ith grid variable, and \(N_j\) is the number of logic cells in the jth grid. In this way, the different PCs are ordered based on their criticalities and assigned to the Sobol dimensions in sequence. If a cell has a smaller slack time, it has a higher probability of having parameters of that cell assigned to Sobol samples rather than LHS samples.

The graph below shows the ratio of runtime speedup of the hybrid technique over traditional Monte-Carlo for different confidence intervals.
V. STATISTICAL STATIC TIMING ANALYSIS USING MARKOV CHAIN MONTE CARLO

In modern circuits, chips are increasing in complexity and in the number of transistors. This introduces an increasing number of possible variations that can affect the timing of circuits. Therefore using statistical static timing analysis (SSTA) is becoming more difficult. If we want high yields for greater than 99.98% then we must analyze the failure regions closely.

With the increasing number of dimensions techniques that have significant improvement to standard Monte Carlo is difficult to obtain. This paper applied a method called Markov Chain Monte Carlo to the statistical timing analysis problem and shows over 10x reduction in computational effort over standard Monte Carlo for a specific setting.

A model is constructed that assumes independent varying gate delays for the circuit. They assume that rise/falling transitions or input combinations do not affect the gate delay. They also ignore the interconnect delays for simplicity.

Then from [4]

We associate a ‘gate delay’ $d_v$ with each gate $v$, the set of gates being denoted by $V_g$. Denote by $I$ the set of inputs and pseudoprimary inputs, and by $O$ the set of outputs and pseudo-primary outputs. Let $V = V_g \cup I \cup O$. Let $d_v, v \in V_g$ have a distribution $p_v(\cdot)$, denoted by $d_v \sim p_v(\cdot)$, and let $P$ be the corresponding product distribution. We denote the set of allowed gate delay vectors $\bar{d}$ of $(d_v, v \in V_g)$ by $S_d$. We define arrival time $D_v$ for each $v \in V$ corresponding to the longest path arrival time at $v$.

They define $D_{req}$ to be the required delay time of the entire circuit and $D$ to be the largest arrival time at the output. Therefore $D > D_{req}$. The Markov chain algorithm then estimates yield for a range of $D_{req}$ such that they get high yields.

The algorithm is as follows.

First define a density function $Q(\cdot)$ to be the density function of random variable $X$. Then uniquely associate $\bar{d}$ where the circuit arrival time is given as $D(\bar{d}) = D$. To get a distribution weighted more toward the slower end of the delay, they choose a ‘tilted’ distribution $\pi(\cdot)$ which is defined for this problem to be

$$
\pi_d(\bar{x}) = \frac{1}{Z(\beta)} P_d(\bar{x}) \exp(\beta D(\bar{d})), \quad \text{for all } \bar{x} \in S_d
$$

Where $\beta > 0$, and $Z(\beta)$ is a normalizing constant given by the equation

$$
Z(\beta) = \left( \int_{-\infty}^{\infty} \pi_D(x) \exp(-\beta x) \, dx \right)^{-1}
$$

And $\pi_D$ represents the density under $\pi(\cdot)$ given by

$$
\pi_D(x) = \frac{1}{Z(\beta)} P_D(x) \exp(\beta x)
$$

So the tail end or the worst delay portion of the delay distribution $PD(\cdot)$ can be inferred to be

$$
P(D \geq D_0) = Z(\beta) \int_{D_0}^{\infty} \pi_D(x) \exp(-\beta x) \, dx
$$

Now $PD(\cdot)$ can be estimated using the table below

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A baseline MCMC-based algorithm for SSTA</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

The MCMC-B is based on the Metropolis-Hastings Algorithm.

With this algorithm however a change in the delay at least once in steps 4 and 5 requires $V_g \ln(V_g)$ steps requiring a recalculation of the entire fanout cone to calculate $D$.

Instead the gates in $V_g$ are assigned to a level by grouping them so that the nodes with levels lower than $l$ are called predecessors and nodes greater than $l$ are called successors. $R$ is the numbers of levels which is also the depth of the circuit.

Now in step 4 and 5 they select a level at random instead of just a gate. This makes the computational effort per step linear so that

$$
\cdot R \ll |V_g| \Rightarrow R \ln R \ll |V_g| \ln |V_g|.
$$

The authors also implement another tweak called parallel tempering.

First create $k$ tilted distributions
\[ \pi_i(d) \propto \mathbb{P}(d) \exp(f_i(D)) \quad i = 1, \ldots, k \]

Where \( f_i \) are functions that are monotonic and non-decreasing in \( D \). Then for any \( j > i \)

\[ f_j(D_2) - f_j(D_1) \geq f_i(D_2) - f_i(D_1) \quad \forall D_2 > D_1 \]

Parallel tempering allows multiple Markov chains to be run for different values of \( i \). At each time step a ‘normal’ move or a ‘swap’ can be done. A normal move consists of updating each Markov chain according to a rule. A swap is for example in 2 dimensions moving from \((d_1, d_2)\) to \((d_2, d_1)\). Swapping helps greatly with mixing.

Finally another tweak made is to use concave exponents for the tilted distribution equation instead of a linear one because the Gaussian distribution is tilted.

This modified Markov chain algorithm they call (MCMC-PT) was tested on two different circuits. One is the ‘maccontrol’ circuit from Opencores Ethernet MAC design and the other is a router designed at the Concurrent VLSI Architecture group at Stanford.

The performance for the new algorithm was compared to standard Monte Carlo.

![Graphs showing delay tail estimates for 'Maccontrol' and 'Fifty' circuits.](image)

The MCMC-PT algorithm has slightly worse performance for yields under 99% but has much better performance for yield greater than 99%.

For 99.9% yield MCMC-PT has an average error of 10 while standard Monte Carlo has 25% error. For estimates for very high yield of 99.99% MCMC-PT gives and error of around 20% Standard Monte Carlo results are not given because it would take 15x more samples than the initial 17000 to match the results given by the Markov chain method.

The main problem with Monte Carlo is that it requires a large number of samples. Multiple approaches were suggested such as Quasi Monte Carlo (QMC) and LHS (Latin Hypercube). However QMC cannot be used if the problem has large number of dimensions (>12). Some alternative methods are to combine the methods such as QMC and LHS and include stratification. However for statistical timing analysis which has large deviations MCMC-PT gives significant gains over MC.

The disadvantage to this method is that some parameters must be chosen by hand and depends on the circuit. For example the number of tempering chains, \( k \). The selection of these parameters also requires sampling and mixing checks.

Some other disadvantages are that this algorithm may not perform well with too many levels or mixing may take too long to perform.

VI. FAST MONTE CARLO ESTIMATION OF TIMING YIELD WITH IMPORTANCE SAMPLING AND TRANSISTOR-LEVEL CIRCUIT SIMULATION

For most block based SSTA methods in order to simplify analysis certain assumptions are made. The basic SSTA algorithms ignore correlations between random variables and assume that all process parameters and gate delays have a Gaussian distribution. While this can make the SSTA algorithm very efficient however by making these simplifications, it can have detrimental effects on the accuracy and results obtained by SSTA.

Other methods that take into account correlations and non-Gaussian distributions have been created but at the same time computational complexity increases. Nonetheless these block based SSTA on abstract timing graphs are widely accepted.

For VLSI design, transistor level circuit simulations are performed on the critical path of a circuit using deterministic static timing analysis however it would be ideal to use a statistical method.

Monte Carlo can be used however the number of samples needed for good accuracy is prohibitively high. Therefore a method is needed to speed up Monte Carlo.

This paper uses a variance reduction technique called importance-sampling. The method presented in this paper is meant to complement not replace the fast block based SSTA methods.

A model is first created to model the inter and intra-die process variations in transistor parameters. They are modeled using a quad-tree model given by
\[ P = P_{\text{inter}} + \sum_{q=1}^{Q-1} R_{\text{intra}}(x, y) \]

Where \( P_{\text{inter}} \) models the perfectly correlated intra-die variations and \( R_{\text{intra}}(x, y) \) is the layout position \((x, y)\) of dependent random variables assigned to the level \( q \). \( Q \) is the total number of levels. \( P_{\text{inter}} \) and \( R_{\text{intra}}(x, y) \) are assumed to be independent gaussian random variables.

Next a gate delay model is constructed.

\[ d_{r}^{PDM} = (L_{r}, V_{tr}, F_{r}, \text{InS}_{r}) \]

Where delay is a function of the channel length \( L_{r} \), threshold voltage \( V_{tr} \), fanout \( F_{r} \), and input slope \( \text{InS}_{r} \). The \( r \) subscript denotes that these parameters are random.

They also defined an output slope contributed by the polynomial gate delay model (PDM).

\[ \text{Out}_{r}^{PDM} = (L_{r}, V_{tr}, F_{r}, \text{InS}_{r}) \]

The input slope of the next gate is the output of the previous gate. Then delay of the path is the sum of the delays from each gate. Critical paths of the circuit can be identified by running MC analysis to identify the critical path and those near it. The false paths are eliminated using a static sensitization test.

Now an importance sampling based estimation of timing yield can be constructed. First they define \( \text{Loss} \) to be the fraction of circuits that fail the timing test. The importance-sampling based estimator for loss is

\[ Loss_{IS}^{\text{MC}} = \frac{1}{N} \sum_{i=1}^{N} I^{\text{MC}}(T_{e}, X_{i}) f(X_{i}) \]

This function draws the samples \( X \) from another biasing distribution

\[ f^{\text{IS}}(X) = \frac{I^{\text{PDM}}(T_{e}, X) f(X)}{Loss^{PDM}_{\text{MC}}} \]

The lost estimate \( Loss^{PDM, e} \) and \( I^{\text{PDM}}(T_{e}, X) \) are computed using a cheap gate delay model. \( \epsilon \) is defined as a margin parameter introduced to guarantee that the biasing distribution is nonzero everywhere that \((T_{e}, X)\) is nonzero. The Loss equation can be simplified by substituting the biasing distribution equation into the the Loss equation to give

\[ Loss_{IS}^{\text{MC}} = \frac{Loss^{PDM, e}}{N} \sum_{i=1}^{N} I^{\text{PDM}}(T_{e}, X_{i}) \]

The algorithm to generate these loss values is given in the table below.

<table>
<thead>
<tr>
<th>Algorithm 1 CompLossMC.IS(NS, SM, T_{c})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Generate NS sample points {X_{1}, X_{2}, ..., X_{NS}} from ( f(X) ).</td>
</tr>
<tr>
<td>2. For each ( X_{i} ), compute ( d_{r}^{PDM}(X_{i}) ).</td>
</tr>
<tr>
<td>3. Let ( \mathcal{Y} = {Y_{1}, Y_{2}, ..., Y_{NS}} ) be the NS samples in decreasing order of ( d_{r}^{PDM}(Y) ).</td>
</tr>
<tr>
<td>4. ( i = 1, \mathcal{Z} = \emptyset, \text{SafeMargin} = \emptyset ).</td>
</tr>
<tr>
<td>5. ( \text{while} ((\text{SafeMargin} &lt; SM) \text{ and } i \leq NS) ) do</td>
</tr>
<tr>
<td>6. ( d_{r} = d_{r}^{PDM}(Y_{i}) ).</td>
</tr>
<tr>
<td>7. ( \text{if} (d_{r} &lt; T_{c}) \text{ then} )</td>
</tr>
<tr>
<td>8. ( \mathcal{Z} = \mathcal{Z} \cup {Y_{i}} ).</td>
</tr>
<tr>
<td>9. ( \text{if SafeMargin} = \emptyset ) then</td>
</tr>
<tr>
<td>10. ( \epsilon = T_{c} - 0.5(d_{r}^{PDM}(Y_{i}) + d_{r}^{PDM}(Y_{i-1})) ).</td>
</tr>
<tr>
<td>11. ( \text{end if} ).</td>
</tr>
<tr>
<td>12. ( \text{SafeMargin} = \text{SafeMargin} \cup {Y_{i}} ).</td>
</tr>
<tr>
<td>13. ( \text{else} ).</td>
</tr>
<tr>
<td>14. ( \text{SafeMargin} = \emptyset ).</td>
</tr>
<tr>
<td>15. ( \text{end if} ).</td>
</tr>
<tr>
<td>16. ( i = i + 1 ).</td>
</tr>
<tr>
<td>17. ( \text{end while} ).</td>
</tr>
<tr>
<td>18. ( \text{Let } N = i - SM - 1 ) and ( \text{SafeMargin} = {Y_{N+1}, ..., Y_{NS}} ).</td>
</tr>
<tr>
<td>19. ( \text{Let } \mathcal{W} = {Y_{1}, ..., Y_{N}} ).</td>
</tr>
<tr>
<td>20. ( \mathcal{Q} = \mathcal{W} \cap \mathcal{Z} ).</td>
</tr>
<tr>
<td>21. ( \text{Loss}^{\text{IS}} = \text{Loss}^{PDM}_{\text{MC}} \cdot \frac{</td>
</tr>
</tbody>
</table>

An important point here is determining the number of sample points, NS. The user provides the algorithm with a number that he expects to carry out approximately because this algorithm is intended to be a late stage yield estimator. If an initial estimate is not available \( \text{Loss}^{PDM}_{\text{MC}} \) can be used.

The key benefit of this algorithm is that transistor level (TL) simulations can be avoided for discarded samples.

The convergence error of the standard estimator is given by the equation

\[ \text{Error}^{TL}(N) = 2\sqrt{\text{Loss}^{TL} \cdot \text{Yield}^{TL}} \cdot \frac{1}{\sqrt{N}} \]

And the error of the importance-sampling method is given by

\[ \text{Error}^{IS}(N) = 2\sqrt{\text{Loss}^{TL} \cdot (\text{Loss}^{PDM, e} - \text{Loss}^{TL})} \cdot \frac{1}{\sqrt{N}} \]

Then the error ratio is just the ratio of these two equations. The speedup of the algorithm can be computed with

\[ \text{Speedup} = \frac{N_{TL}}{N_{IS}} = \frac{\text{Yield}^{TL} - \text{Loss}^{PDM, e}}{\text{Loss}^{PDM, e} - \text{Loss}^{TL}} \]

For experimental results, the ISCAS’85 benchmark results are used with the transistor lengths and threshold voltage being the parameters to be considered. For each circuit the statistically critical paths are found. Then two timing constraints are chosen that result in 10% and 5% loss denoted as \( T_{c,\text{low}} \) and \( T_{c,\text{high}} \). The accuracy and efficiency of the IS estimator to the standard MC estimator are measured.
The IS estimator gives on average two magnitude of speedup over the standard Monte Carlo method. The importance-sampling method is able to converge at a smaller number of transistor level simulations compared to standard Monte Carlo.

While this method reduces the number of transistor level simulations needed, in order to compute the algorithm it needs an initial Loss value that has to be computed ahead of time. Also it does not replace standard block based SSTA but compliments it.

VII. CORRELATION-PRESERVED STATISTICAL TIMING WITH A QUADRATIC FORM OF GAUSSIAN VARIABLES

The last paper we will review presents a method to account for correlation of random variables to improve block based static timing analysis. Traditional first order block based analysis performs static timing analysis by treating each gate/wire delay as a block and performing timing analysis block by block.

Previous history of blocks are not considered. However this can cause a problem because two delays might be affected by the same variation for example supply voltage, gate channel, or wire geometry variations.

The traditional delay timing model is

\[ D = \mu + \alpha R + \sum_i \beta_i G_i \]

Where G is the global variations and R are local variations. The delay is Gaussian since it is a linear combination of Gaussian random variables. When devices scale down, nonlinearity of delays as a function of global variations cannot be modeled accurately by the traditional delay equation since the delay will no longer be Gaussian.

This paper presents a “quadratic timing model” that adds a second order term to the delay model above

\[ D = m + \alpha R + \sum_i \beta_i G_i + \sum_{i,j} \Gamma_{ij} G_i G_j \]

Where \( \Gamma_{ij} \) are quadratic coefficients and m is a constant that may not necessarily be the mean of the delay.

To analyze the effect of non-linear sources on the delay the Taylor series expansion of the delay formula is used

\[ D(G_1, G_2, \ldots, G_p) \]

\[ = m + \alpha R + \sum_{j=1}^{\infty} \left( \frac{1}{j!} \left( \sum_{i=1}^{p} \frac{\partial D}{\partial G_i} \right)^j \right) \times D(G_1', G_2', \ldots, G_p') \]

For the quadratic model the tailor series is truncated at the second order term.

\[ D_g \approx m_g + \alpha R + \frac{\partial D_g}{\partial L} L + \frac{\partial D_g}{\partial V} V + \cdots \]

\[ + \frac{1}{2} \frac{\partial^2 D_g}{\partial L^2} L^2 + \frac{1}{2} \frac{\partial^2 D_g}{\partial V^2} V^2 + \cdots \]

Where L and V are the global variations in length and voltage. The coefficients are extracted from SPICE by obtaining a hyperplane of gate delays vs. variation parameters and fitting them to the Taylor expansion. This equation can be written in a compact form

\[ D_g = m_g + \alpha R + \beta_g \delta_g + \delta_g^T \Gamma_g \delta_g \]

Where

\[ \delta_g = [G_1, G_2, \ldots, G_p]^T \sim N(0, \Sigma_g) \]

And

\[ \beta_g(i) = \frac{\partial D_g}{\partial G_i} \text{ and } \Gamma_g(i, j) = \frac{1}{2} \frac{\partial^2 D_g}{\partial G_i \partial G_j} . \]
The quadratic gate delay model shows great accuracy improvements over the traditional timing model. The inverter delay is computed by using SPICE Monte Carlo method as well as the traditional and quadratic delay models.

![Inverter Delay Distribution](image1)

Wire delays will also be treated with the quadratic model. Using the distributive wire delay model and separating the wire into N segments of length L, each segment will have a width W and thickness T that can be considered as global variables. These variables are considered to be Gaussian but are not uncorrelated.

Wire delay can be calculated using the Elmore Delay model

\[
D_w = \frac{1}{W} \sum_{i=1}^{N} \sum_{j=1}^{N} r_s L^2 (c_w W_j + c_f T_j) / W_i T_i
\]

Similar to the gate delay formula the Elmore delay can be expanded by the Taylor series and truncated to the second order.

\[
D_w \approx m_w + \alpha R + \beta_w \delta_w + \delta_w \Gamma_w \delta_w
\]

Where \(\delta_w\) is a 2Nx1 global variation vector given by

\[
\delta_w = [W_1', W_2', \ldots, W_N', T_1', T_2', \ldots, T_N'] \sim N(0, \Sigma_w)
\]

The wire delay from SPICE, canonical and quadratic delay models are shown in the following graph. It is important to note that even if the thickness and wire widths are considered to be Gaussian, the overall delay will not be. Again the quadratic delay model shows accuracy improvements over the traditional delay model.

![Inverter Delay Distribution](image2)

Next spatial correlations must be considered. Global parameters such as length, supply voltage, and temperature are not independent since nearby devices might have similar variations. The correlation between them is modeled by a grid and each grid cell will be assigned a global parameter variation. Also correlations between grids decrease as the distance between them increases.

![Spatial Correlation Matrix](image3)

While a finer grid is desired for better accuracy it will result in a large covariance matrix which degrades the speed at which static timing analysis can be performed. Therefore a tradeoff must be made.

For timing analysis the signal arrival time is the cumulative effect of all gate/wire delays at its input cone. If the gate/wire delays are quadratic in form then the arrival delay time will also have a quadratic form of

\[
D_a = m_a + \alpha_a r_a + \beta_a \delta_a + \delta_a \Gamma_a \delta_a
\]

In block based static timing analysis determining arrival times involves two operations.
1) ADD: The output arrival time will be the input arrival time $X$ plus the gate delay time $Y$. So output arrival time, $Z = X + Y$.

For quadratic timing analysis $X$ and $Y$ are expressed by the quadratic form of arrival and delay times mentioned previously

$$Z = X + Y \sim Q(m_Z, \alpha_Z, \beta_Z, \Gamma_Z)$$

2) MAX: If two arrival times merge at a gate the new arrival time is the maximum of the two $Z = \max(x, y)$.

The max operator is non-linear but it will be approximated as a linear function. When doing this there will be some error which must be minimized.

If $Z = \psi(X, Y)$ is an approximating of $Z = \max(X, Y)$ then the error function is defined as

$$\Delta = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} [\max(x, y) - \psi(x, y)]^2 f_{XY}(x, y) \, dx \, dy$$

$$-E[Z^2] + E[\tilde{Z}^2] - 2E[Z\tilde{Z}]$$

If the function $Z = \psi(X, Y)$ is defined by parameters $\lambda_i$ there is some optimum value of $\lambda_i$ such that the error is minimized which are defined as

$$E \left\{ Z \frac{\partial \tilde{Z}}{\partial \lambda_i} \right\} - E \left\{ Z \frac{\partial \tilde{Z}}{\partial \lambda_i} \right\} = \frac{1}{2} \left( \frac{\partial^2 \tilde{Z}}{\partial \lambda_i^2} \right)$$

For this optimum case the max function can be approximated as linear by the equation $\tilde{Z} = aX + bY + c$. And the coefficients $a, b, c$ are such that they minimize the error function and are solutions to the following equations

$$E[Z] = aX + bY + c$$

$$\text{cov}(Z, X) = a \cdot \sigma_X^2$$

$$\text{cov}(Z, Y) = a \cdot \sigma_Y^2$$

Then the a max arrival time can be calculated as a function of the quadratic delay formulas with the parameters being

$$\alpha_Z = a\alphaX + b\alphaY$$

$$m_Z = a\alphaX + b\alphaY$$

$$\beta_Z = a\betaX + b\betaY$$

The quadratic SSTA algorithm has greater computational complexity which comes from computing the moments of the quadratic forms and updating the coefficient matrix $\Gamma$.

The new block based SSTA was compared with the first order model and Monte Carlo simulation with 10,000 repetitions. Using simple gates the delays were measured while varying different parameters as well as the global parameters, gate length, supply voltage, and temperature.
REFERENCES


