Practical, Fast Monte Carlo Statistical Static Timing Analysis: Why and How

Amith Singhee[1], Sonia Singhal[2], Rob A. Rutenbar[3]
[1]IBM T J Watson Research Center, NY, [2]Synopsys Inc., CA, [3]Carnegie Mellon University, PA
asinghe@us.ibm.com, soniasin@ece.cmu.edu, rutenbar@ece.cmu.edu

Abstract

Statistical static timing analysis (SSTA) has emerged as an essential tool for nanoscale designs. Monte Carlo methods are universally employed to validate the accuracy of the approximations made in all SSTA tools, but Monte Carlo itself is never employed as a strategy for practical SSTA. It is widely believed to be "too slow" – despite an uncomfortable lack of rigorous studies to support this belief. We offer the first large-scale study to refute this belief. We synthesize recent results from fast quasi-Monte Carlo (QMC) deterministic sampling and efficient Karhunen-Loéve expansion (KLE) models of spatial correlation to show that Monte Carlo SSTA need *not* be slow. Indeed, we show for the ISCAS89 circuits, a few hundred, well-chosen sample points can achieve errors within 5%, with no assumptions on gate models, wire models, or the core STA engine, with runtimes less than 90 s.

1 Introduction

Tools for nanometer-scale circuits must manage increasing amounts of statistical variation across all phases of design. Variation sources may be global (e.g., wafer-level process problems [14]) or local (e.g., random dopant fluctuation [13]), and possess a complex spatial or temporal correlation structure. These problems have generated a wave of statistically "aware" tools and methods, in particular, Statistical Static Timing Analysis (SSTA) [1][2][4].

SSTA tools typically exploit some simplifications to make the computation tractable. The most adopted implementations [1][2] typically assume a linear dependence of gate delays (or slews) on the statistical parameters (e.g., channel length L, threshold voltage V_t) for gates. This linear model enforces a severe approximation of the max() operator used to compute the worst case delay of any gate, to maintain the linear dependence of the circuit delay on all statistical parameters. Early implementations also required the assumption of normally distributed statistical parameters. This, of course, leads to errors in the estimates of the circuit delay (or slack) distributions. Several authors [4][8][9] have proposed extensions to nonlinear gate models and/or non-normal distributions. These extensions usually result in higher computation cost that might not scale cheaply with an increasing number of parameters – a trend expected for upcoming technologies. As a result these extensions seem not to have yet found wide adoption in practice. We will refer to this entire class of SSTA approaches as being *model-based*.

On the other hand, we have Monte Carlo simulation [11], which is universally used to validate the accuracy of all practical SSTA tools, but never used actually to *implement* practical SSTA tools. On the positive side, Monte Carlo would seem to offer several desirable advantages:

- Monte Carlo requires no simplifying assumptions about the statistical parameter distributions, the form of any gate delay/slew models, or the +/max() operators.
- Monte Carlo accuracy is independent of the number of statistical parameters, or the circuit size [5]. The root-mean-square error in the estimate of any metric decreases as $O(n^{-0.5})$, where n is the Monte Carlo sample size used. Hence, the method is highly scalable.
- Monte Carlo's direct relationship between the sample size and the error gives users a flexible knob to completely control the error. Such trade-off flexibility is not provided by the prevailing model-based SSTA methods.
- Monte Carlo algorithms are naturally parallelizable, a feature increasingly attractive in the coming era of many-core processors.
- Monte Carlo methods give us actual samples of the different statistical outcomes for a circuit, not just estimates of distributions or sensitivities. Such samples, in the context of SSTA, would allow easy computation of circuit criticalities [2], and other fine-grain analyses of arbitrary paths.

On the negative side, the main hurdle to adoption of Monte Carlo for SSTA has been the high cost of thousands of STA runs. We note here the work in [10], which proposes a fast sparse matrix-based formulation for Monte Carlo SSTA, but is constrained to a path-based formulation, which may not scale well to large circuits. Thus, Monte Carlo has been relegated to a supporting role as the "gold standard" for validating the accuracy of today's SSTA tools [1][2][4][8].

What would be required to make Monte Carlo SSTA truly competitive with existing SSTA methods? The answer is simple: a method requiring dramatically fewer random STA runs, without compromising accuracy. For example, in validating standard SSTA tools, we see typical sample sizes of 10,000 STA runs. This is much too slow; but, if we could reduce this to a few 100s of sample points, Monte Carlo SSTA would become vastly more attractive.

In this paper, we show how to achieve exactly this reduction in Monte Carlo SSTA sample size. We synthesize two different, recently introduced ideas: reduced sampling via deterministic quasi-Monte Carlo (QMC) methods [5], and dimensionality reduction via a Karhunen-Loéve expansion (KLE) of a random field model of intra-die variation [4][7].

Quasi-Monte Carlo sampling has been used most spectacularly in the world of computational finance [11], where it is the workhorse method for evaluating complex financial instruments. Such evaluations involve high-dimensional, nonlinear statistical integrals of the same type found in modern statistical circuit problems, a connection first made explicit in [5]. [5] showed how QMC's fast convergence properties – faster than $O(n^{-0.5})$ – could be exploited for custom circuits. Unfortunately, QMC requires a careful mapping of "important" statistical variables to the individual dimensions of the sampling process, for effective use in high dimensions. [5] suggested using either designer expertise or rank correlation coefficients for this purpose. In the case of SSTA, we have the twin obstacles that the inherent dimensionality is extremely large (thousands to millions), and neither designer expertise nor rank correlation are tractable for measuring statistical importance of each variable.

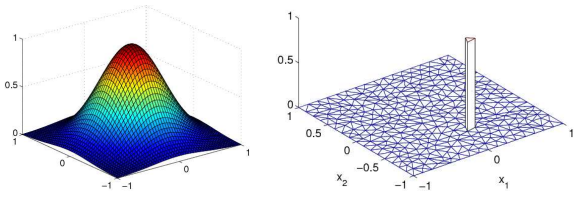
978-1-4244-2820-5/08/$25.00 ©2008 IEEE

This is where a recent advance in KLE-based spatial correlation modeling [4], made practical by [7], is essential. The KLE of a random field model of intra-die statistical variation can take an extremely large model, with many random variables (RVs) to represent gate-level statistics, and reduce these to few tens of uncorrelated RVs. [7] showed that this reduction already improves the performance of conventional Monte Carlo samples. But, in this paper, we show that one can do even better, in particular, we show that the KLE model is *precisely* what is needed to make ultra-fast QMC sampling scale to practical SSTA problems. It is the pair of novel strategies – KLE dimension reduction + QMC sample reduction – that form the missing ingredients needed to make Monte Carlo SSTA tractable.

The paper is organized as follows. Sec. 2 briefly reviews the random field model and the application of KLE to reduce the dimensionality of our SSTA problems. Sec. 3 then reviews standard sampling strategy from conventional Monte Carlo, and its application to our random field model. Sec. 4 and 5 review Latin hypercube sampling and our proposed "best" sampling method, quasi-Monte Carlo, respectively, and also provide some theoretical insight into why these offer different advantages in the SSTA setting. Sec. 6 presents our experimental setup and results, and Sec. 7 offers concluding remarks.

2 Reducing the Problem Dimension: the Grid-less Spatial Correlation Model

In conventional modeling practice, we take our logic network, place it, partition the placement surface into a coarse grid, and model intra-die spatial correlations as coarse grid-to-grid correlations. This has several flaws, as discussed in [3]. In brief, it is not clear if the grid-based covariance matrix will always be valid, and it is difficult to estimate the best (or near-best) grid structure. Typically, the grid is chosen in an *ad hoc* manner. Thus, we begin with a *grid-less stochastic process* model for spatially correlated intra-die variation, as proposed in [4], which does not rely on arbitrary partitioning of the chip area as in [1][15].



(a) (b)

Figure 1: (a) A double exponential (Gaussian) covariance kernel. (b) Triangular partition of chip area D and one basis function.

Let $K(\mathbf{x},\mathbf{y})$ be a function that returns the covariance of the parameter p (e.g., L, V_t, t_{ox}) at locations $\mathbf{x}$ and $\mathbf{y}$ on the normalized chip area $D = [-1,1] \times [-1,1]$ ($\mathbf{x},\mathbf{y} \in D$). We refer to this function as the *covariance kernel* for p. Having normalized the parameters, the covariance is equal to the correlation. We anticipate the correlation to drop off monotonically as we move away from any given point $\mathbf{x}$ on the chip. A valid covariance kernel on a domain $D \times D$ must be *non-negative definite* [7]. [3] shows how to extract valid kernels from measurement data. One valid kernel is the Gaussian kernel $K(\mathbf{x},\mathbf{y}) = \exp(-c||\mathbf{x}-\mathbf{y}||_2^2)$, shown in Fig. 1(a).

Consider the channel length (L) variation over a die. We can imagine that for every *outcome*, θ, we get one die, and hence one 2-D function $L(\mathbf{x})$ defining the value of L at any location $\mathbf{x}$ on the die. We can represent any such function as $p(\mathbf{x},\theta)$, where $\mathbf{x} \in D$ and $\theta \in \Omega$, the set of all possible outcomes. If the values of p across the domain D (the die) follow the covariance kernel K, then p is a *stochastic process* with covariance kernel K. Hence, we can represent any intra-die variation parameter as a stochastic process. Given these definitions, the following is true:

Theorem 1 (Karhunen Loéve Expansion) *Let the stochastic process $p(\mathbf{x},\theta)$ on a closed domain D have bounded variance over D and a covariance kernel $K(\mathbf{x},\mathbf{y})$ that is continuous at $(\mathbf{x},\mathbf{x}), \forall \mathbf{x} \in D$. Then p has the orthogonal decomposition*

$$p(\mathbf{x},\theta) = \sum_{j=0}^{\infty} \sqrt{\lambda_j}\xi_j(\theta) f_j(\mathbf{x}) \tag{1}$$

where λ_j is the j-th largest eigenvalue of the covariance kernel K and $f_j(\mathbf{x})$ is the corresponding eigenfunction of K.

The eigenpairs (λ_j, f_j) are solutions of the integral equation

$$\int_D K(\mathbf{x},\mathbf{y}) f(\mathbf{y}) d\mathbf{y} = \lambda f(\mathbf{x}), \tag{2}$$

where f_j are orthonormal and the RVs ξ_j are uncorrelated. For a proof, see [12]. From (1), we see that the j-th eigenvalue λ_j is a measure of the contribution of the j-th RV ξ_j to the overall variance of the process. KLE does not place restrictions on the distribution of the stochastic process.

[7] shows that the stochastic behavior of the thousands to millions of gates on a chip can be represented using a much *reduced* set of *r uncorrelated* RVs by truncating (1) at the first r eigenpairs. For the ISCAS89 circuits, [7] showed that using $r = 25$ eigenpairs for each statistical parameter yielded errors of only 0.35% on average. [7] uses a Galerkin technique to solve (2) and quickly compute the first r eigenpairs, for arbitrary K. It proposes a generic algorithm and then gives a specific implementation. The former involves representing a solution f of (2) as a linear combination of some n_b orthogonal basis functions $\psi_i(\mathbf{x})$: $f(\mathbf{x}) \approx \sum_{i=1}^{n_b} d_i \psi_i(\mathbf{x})$. Writing the corresponding residual as

$$r_n(\mathbf{x}) = \sum_{i=1}^{n} d_i \left\{ \int_D K(\mathbf{x},\mathbf{y}) \psi_i(\mathbf{y}) d\mathbf{y} - \lambda_n \psi_i(\mathbf{x}) \right\}, \tag{3}$$

the unknowns d_i and λ are computed to satisfy the Galerkin criterion

$$\int_D r_n(\mathbf{x}) \psi_k(\mathbf{x}) d\mathbf{x} = 0, \quad k = 1, \ldots, n. \tag{4}$$

This ensures that any residue behavior is unexplainable using our finite basis set. (3) and (4) lead to a matrix formulation, which we state for the specific implementation as follows. Discretize the domain D using a mesh of n_b triangles, and take ψ_i as the function equal to 1 over triangle i and 0 everywhere else (Fig. 1(b)). If $\triangle_i$ is the i-th triangle, a_i its area and $\mathbf{x}_{\triangle_i}$ its centroid, (3) gives a matrix formulation,

$$(\Psi^{-1}\mathbf{K})\mathbf{d} = \lambda_{n_b}\mathbf{d}, \quad (\Psi^{-1}\mathbf{K})_{ik} = K(\mathbf{x}_{\triangle_i}, \mathbf{x}_{\triangle_k}) a_k, \tag{5}$$

which is an easily solvable eigenvalue problem (EP). λ_{n_b} is the approximation to λ, and $\mathbf{d} = (d_1, \ldots, d_{n_b})$ is the corresponding eigenvector. This technique applied (1) is a generalization of PCA on regular grids [1]. We can write (5) as

$$\mathbf{KAd} = \lambda_{n_b}\mathbf{d}, \quad \mathbf{K}_{ik} = K(\mathbf{x}_{\triangle_i}, \mathbf{x}_{\triangle_k}),\ \mathbf{A}_{ik} = \delta_{ik} a_i, \tag{6}$$

where $\mathbf{K}$ is the covariance matrix for locations at the centroids of the triangles and δ_{ik} is the Kronecker delta. For PCA over regular grids, the discretization of D consists of squares with equal area a, and we solve the EP

$$\mathbf{K}a\mathbf{I}_{n_b}\mathbf{d} = \lambda_{n_b}\mathbf{d} \Leftrightarrow \mathbf{Kd} = (\lambda_{n_b} a^{-1})\mathbf{d}, \tag{7}$$

Algorithm 1 Generate n points for Monte Carlo

given gate locations $\{\mathbf{g}_i\}_{i=1}^{N_g}$
for all stat. parameters p_j **do**
  $\mathbf{K}_j \leftarrow \text{CovMatrix}(K_j, \{\mathbf{g}_i\})$
  $\mathbf{U}_j \leftarrow \text{CholeskyUpperFactor}(\mathbf{K}_j)$
  $\mathbf{P}_j \leftarrow \text{RandNormal}(n, N_g) \cdot \mathbf{U}_j$ // impose correlation
end for

where the new eigenpairs are the ones used in PCA [1][7]. Note that other implementations of the generic algorithm may not be such direct generalizations of PCA.

Because our SSTA algorithm works with the reduced set of r RVs, ξ_j, the values for the original statistical parameter $\mathbf{p}(\mathbf{x},\theta)$ (e.g., L) must be reconstructed for use in the gate models. Define the matrix $\mathbf{D}_\lambda = \mathbf{D}_r\sqrt{\Lambda_r}$: Λ_r is a diagonal matrix containing the eigenvalues, the i-th column of $\mathbf{D}_r$ is the i-th eigenvector of (5), and the square-root is taken element-wise. Then, for a point $\bar{\xi}$ in the reduced r-dimensional RV space,

$$\mathbf{p}_\triangle = \mathbf{D}_\lambda \bar{\xi}, \tag{8}$$

where the i-th element $p_{\triangle_i}$ of the vector $\mathbf{p}_\triangle$, approximates the value of $\mathbf{p}(\mathbf{x},\theta)$ in $\triangle_i$ as a constant over $\triangle_i$. Note that $\mathbf{D}_\lambda$ depends on the manufacturing process and needs to be computed only once for use on any number of designs.

3 Standard Sampling: Random Monte Carlo

At this point, we have reduced the number of statistical variables (r) that define our SSTA problem; the next challenge is to sample among the RVs in the most efficient manner. As a starting point, we review the simplest sampling strategy. Standard Monte Carlo applied to SSTA is shown as Alg. 1, where the number of random variables generated is equal to the number of gates N_g. Alg. 2 modifies it to work in the reduced r-dimensional space – the number of RVs generated is only r (25 in [7] and here).

Using the points from Monte Carlo, we typically compute some metric like yield, or the 99-th percentile delay. Such a computation is essentially a numerical approximation of some integral

$$Q = \int_{[0,1]^s} f(\mathbf{x}) d\mathbf{x}, \tag{9}$$

over the unit cube in s dimensions, where s is the *total* number of RVs [1]. For example, if we are estimating the mean of the worst circuit delay, f is the circuit delay as a function of the statistical parameters of all the gates. See [5][6] for illustrative explanations. The numerical estimate computed from an n-point Monte Carlo is

$$Q_n = n^{-1}\sum_{i=1}^{n} f(\mathbf{x}_i), \tag{10}$$

where $\mathbf{x}_i$ are the Monte Carlo points. Suppose we use Monte Carlo to estimate the standard deviation of the worst circuit delay in the presence of process variations. Different n-point Monte Carlo runs will give us slightly different estimates. It is well known [6][16] that the variance of this estimate decreases asymptotically as

$$\sigma^2_{MC} = \frac{\sigma^2(f)}{n}, \quad \sigma^2(f) = \int_{[0,1]^s} (f(\mathbf{x}) - Q)^2 d\mathbf{x}, \tag{11}$$

where $\sigma^2(f)$ is the variance of the underlying integrand f in (9). This gives us an r.m.s. error rate of $O(n^{-0.5})$, irrespective of the problem dimensionality. Hence, we expect both Algs. 1 and 2 to show similar convergence behavior in terms of the sample size n. Note, however, that the actual runtime of Alg. 2 can be much lower since it generates many fewer RVs than Alg. 1 and does not have the expensive matrix operations for correlation imposition. [7] shows significant speedups. However, in this paper, we show that there is more performance here to harvest if we couple the KLE dimensionality reduction with precisely the right Monte Carlo sampling strategy. We now look at two other, smarter possibilities for a Monte Carlo type approach to SSTA, Latin hypercube sampling (LHS) and quasi-Monte Carlo (QMC), that promise better performance in many cases.

[1] Note that s is the total dimensionality of the problem. Say we have 2 statistical parameters. Then, $s = 2\times$ (Num. gates N_g). If we use KLE with r eigenpairs for each statistical parameter, then $s = 2r$.

Algorithm 2 Generate n points for KLE-based Monte Carlo

given gate locations $\{\mathbf{g}_i\}_{i=1}^{N_g}$
for all stat. parameters p_j $(L, W, V_t, \dots)$ **do**
  $\Xi_j \leftarrow \text{RandNormal}(n, r)$ // reduced RV space
  $\mathbf{P}_{j\triangle} \leftarrow \mathbf{D}_\lambda \Xi_j$ // reconstrn. (8) gives columns of $\mathbf{P}_{j\triangle}$
  for $i = 1$ to N_g **do**
    $t \leftarrow \text{IndexOfContainingTriangle}(\mathbf{g}_i)$
    $\text{Row}(i, \mathbf{P}_j) \leftarrow \text{Row}(t, \mathbf{P}_{j\triangle})$
  end for
end for

4 Better Sampling: Latin Hypercube

One can easily do better than the simple uniform random sampling of the previous section. The most commonly employed technique for statistical circuit problems is Latin hypercube sampling (LHS). LHS, introduced in [17], is a variance reduction technique based on stratification. It essentially reduces the contribution of the $\sigma^2(f)$ numerator in (11). It first generates a sample of n points uniformly distributed over the unit cube, and then applies an inverse transform to achieve the required distribution – for example, the inverse cumulative normal function gives the normal distribution. Suppose we want to generate n such points in the unit cube in s dimensions. LHS tries to ensure good uniformity along each dimension. Define s independent, random permutations π_i of $\{1,2,\dots,n\}$. Also, let $u_{ij} (i = 1,\dots,n; j = 1,\dots,s)$ be ns i.i.d. random variables distributed uniformly over $[0,1)$. Then the j-th coordinate of the i-th point in an n-point LHS sample is given as

$$x_{ij} = n^{-1}(\pi_j(i) - 1 + u_{ij}), \quad i = 1,\dots,n, \quad j = 1,\dots,s. \tag{12}$$

To obtain normally distributed points, compute

$$x_{ij} = \Phi^{-1}(x_{ij}), \tag{13}$$

where Φ is the cumulative normal distribution function. This construction replaces `RandNormal()` in Algs. 1 and 2.

By using (12), in effect we create n equal slices or *strata* of $[0,1)$ along each coordinate, select a random value within each stratum for all the coordinates and then randomly match up the coordinate values to generate n s-dimensional points. In this manner, LHS ensures high uniformity of the point set along any dimension. This, however, does not guarantee similar uniformity in two (or higher) dimensional projections of the point set. This feature of LHS is the key to its variance reduction property. Write the underlying integrand as

$$f(\mathbf{x}) = f_1(\mathbf{x}) + f_{>1}(\mathbf{x}), \quad f_1(\mathbf{x}) = \sum_{i=1}^{s} f_{\{i\}}(x_i), \tag{14}$$

where $f_{\{i\}}(x_i)$ is that part of f which depends exclusively on the i-th variable. Hence, f_1 is the purely *one-dimensional* part of f, and $f_{>1}$ is the purely multi-dimensional part of f. LHS, as expected, integrates f_1 very accurately because of its high uniformity in one-dimensional projections, and shows Monte Carlo performance on $f_{>1}$. [18] shows that the variance of the LHS estimate is

$$\sigma^2_{LHS} = n^{-1}\sigma^2_{>1} + o(n^{-1}), \tag{15}$$

where $\sigma^2_{>1}$ is the variance of $f_{>1}$, similar to $\sigma^2(f)$ for f. If f is largely one-dimensional, we get large variance reduction and lower errors than Monte Carlo, otherwise we get error behavior similar to Monte Carlo. See [6] for examples and a detailed discussion. Hence, comparison of the convergence of LHS with the convergence of Monte Carlo for any circuit can reveal such characteristics of the computed metrics (e.g., yield or 99-th percentile delay).

5 Best Sampling: Quasi-Monte Carlo

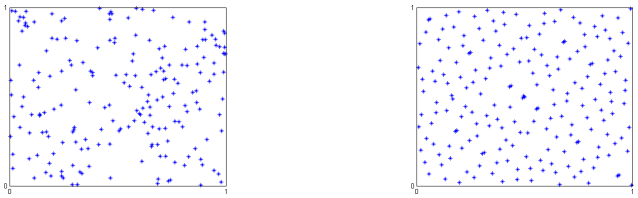
Random samples typically used for Monte Carlo (Fig. 2(a)) suffer from clusters and empty spaces in their distribution over the sampling region (unit cube): they are not very uniformly spread out, especially for not too large n. Although the details are beyond the scope of this paper, this non-uniformity of any set of n points can be represented mathematically by a measure called *star discrepancy* (D^*_n). A powerful theorem (Koksma-Hlawka [19]) shows that lower discrepancy sample sets can lead to lower Monte Carlo integration errors, specifically, the D^*_n discrepancy bounds the error. See [5] for details. The discrepancy of n uniformly distributed random points in s dimensions is [20]

$$D^*_n|_{MC} = O\left(n^{-0.5}(\log\log n)^{-.5}\right), \tag{16}$$

echoing the $O(n^{-0.5})$ convergence of standard Monte Carlo error. Point sequences with asymptotically superior discrepancy exist:

$$D^*_n|_{QMC} = O\left(n^{-1}\log^s n\right), \tag{17}$$

and are called *low discrepancy sequences* (LDSs). This discrepancy behavior suggests an asymptotic integration error rate of $O(n^{-1})$, which is much faster than the $O(n^{-0.5})$ of standard Monte Carlo methods. Several constructions of such sequences have been proposed, by Halton, Sobol', Faure, Niederreiter among others [11], all giving *deterministic* sequences, in contrast to the random sequences of standard Monte Carlo. Monte Carlo performed using samples from these deterministic LDSs is called *quasi-Monte Carlo* (QMC). Inspired by the widespread success of QMC in computational finance, [5] studied it for custom circuit problems and showed significant gains over standard Monte Carlo.



(a) (b)

Figure 2: In two dimensions, low discrepancy points (b) are more uniformly distributed than typical uniform pseudo-random points (a).

The overall idea of QMC is simple: rather than randomly sampling the space, try to "fill" the space with points that are as geometrically, homogeneously equidistant as possible (e.g., Fig. 2). For small dimensionality s, such uniformity is possible with practical sample sizes n. For large s, however, the number of points needed for such uniformity in all dimensions can be extremely large [6]. This is reflected by (17), where a large value of n is needed to make $\log^s n \ll n$, if s is large. For practical sample sizes, this incomplete "filling" is manifested as undesirable patterns of empty regions in the low dimensional projections of the point set [5][11]. Large empty regions can lead to large errors in the integration. However, current best LDSs are able to achieve uniform filling in the *early* dimensions and suffer from patterns mainly in the projections of the *later (higher)* dimensions. For example, a set of 10K Sobol' points in 100 dimensions will have well distributed points in dimensions 1-10, and undesirable patterns in dimensions 91-100.

[5] shows that if we are careful to map the "important" input variables of f to the early dimensions of the Sobol' sequence, we can achieve better than Monte Carlo convergence. The measure of importance used there is the strength of the relationship between f and any input, estimated using designer input or Spearman's rank correlation coefficient. In the case of SSTA, such a measure is not easy to obtain before actually running SSTA. However, we can exploit the KLE-based grid-less model here. KLE gives us the RVs ξ_j in decreasing order of their variance contribution to the stochastic process. We can use this variance contribution as a measure of variable importance: variables contributing less to the variance of the stochastic process will have negligible impact on the circuit delay. Improper sampling of these unimportant variables by the patterned higher dimensions of a Sobol' point set will not affect the integration error much. See [5][6] for supporting theoretical arguments.

In practice, we interlace the RVs for each statistical parameter while doing this mapping. For example, suppose we consider W and L variations and use 5 KLE RVs each, $\{\xi_{W1},\ldots,\xi_{W5}\}\{\xi_{L1},\ldots,\xi_{L5}\}$, respectively. Then, we use a 10 dimensional Sobol' sequence, with the following interlaced variable ordering: $\{\xi_{W1},\xi_{L1},\ldots,\xi_{W5},\xi_{L5}\}$. This ensures that the most important RVs for all statistical parameters are mapped to the earliest, more uniform dimensions.

We note here a parallel of QMC with LHS, which is also an advantage over LHS. LHS achieves excellent uniformity on one-dimensional projections. Sobol' points, and other competent LDSs, achieve good uniformity also in higher-dimensional projections, especially in the early dimensions (the first 10 or so for Sobol' points [22]). This allows QMC to achieve better performance in general, as long as we are careful to use the appropriate variable-dimension mapping. [6] provides detailed examples and discussions.

Since QMC is deterministic, it does not provide us a convenient way to evaluate the error in the estimate if we do not already know the exact value of the integral. A typical approach is to randomize the QMC sequence by *scrambling* the digits of the coordinate values – essentially reordering the sequence of values in each dimension independently of the other dimensions – while maintaining the discrepancy properties of the sequence. Then we can run, say, 30 differently scrambled QMC runs and compute the standard deviation of the estimate across these runs, to be used as a probabilistic measure of error. This also allows comparison with the standard deviations of Monte Carlo and LHS estimates. A rich, but expensive scrambling was proposed in [23]. We use a less powerful, but more scalable approach described in [24].

6 KLE+QMC for SSTA: Experiments

Our experimental setup is the same as [7]. We use a Gaussian kernel ($K(\mathbf{x},\mathbf{y}) = \exp(-c||\mathbf{x}-\mathbf{y}||_2^2)$, Fig. 1(a)) for our tests. Using measurement data, [14] suggests a near linear isotropic kernel (assuming normalized chip sides). The isotropic linear kernel, however, can be invalid [3]. [7] suggests using a Gaussian kernel with c computed for best fit to an isotropic linear kernel in 2-D with correlation distance equal to half the normalized chip length (a cone with a base radius of half chip length). This correlation distance is similar to the one extracted in [14]. To analyze timing, we use a simple, custom core STA engine which computes signal

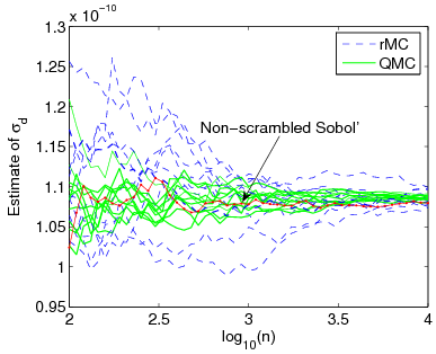


Figure 3: Estimated σ_d with increasing sample size, for 10 MC runs (dashed blue), 10 randomized QMC runs (solid green) and 1 QMC run (dot-solid red) for circuit s38417.

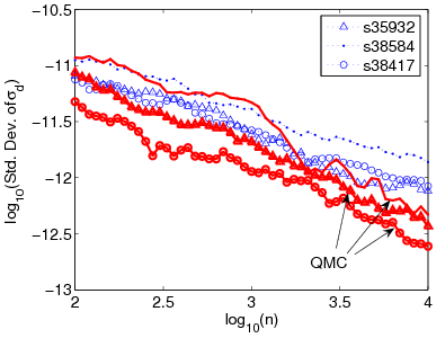


Figure 4: Std. dev. of σ_d estimated using rMC and QMC for the three largest ISCAS89 benchmarks, with increasing sample size.

delays at all the circuit nodes, using the Elmore delay metric [25] for wire delay, the PERI technique [26] with the Bakoglu metric [27] for wire slew, and rank-one quadratic functions [28] to model gate delay and gate output slew. The gate output slew and gate delay are modeled as functions of the input slew and 4 normally distributed statistical parameters: L, W, V_t and t_{ox}. Note that there is no restriction imposed by our technique on the type of gate model used. We use standard logic netlist benchmarks from the ISCAS89 sets. All the test circuits were placed using the Capo placer [29] (MetaPlacer), and half-perimeter wirelength was used to model the wire loads. The gates and wires were implemented using the 90nm Cadence Generic PDK. All statistical parameters were assumed to have Gaussian distributions.

In the following, we estimate the standard deviation of the worst circuit delay (σ_d) in the presence of intra-die variations, using four algorithms:

- **MC**: 10 runs of standard Monte Carlo, as in Alg. 1. Only 10 runs are used because the runtimes are very large, and hence, this algorithm is not a real contender for a Monte Carlo based SSTA implementation.
- **rMC**: 30 runs of standard Monte Carlo using a KLE-*reduced* set of 25 RVs per statistical parameter, as in Alg. 2.
- **LHS**: 30 runs of LHS using a KLE-reduced set of 25 RVs per statistical parameter.
- **QMC**: 30 runs of randomized QMC on a KLE-reduced set of 25 RVs per statistical parameter. We use the Sobol' points; see [5] for details on construction.

Each run uses a sample size of 10K full STA analyses. All algorithms are implemented using compiler-optimized C++ code on a 64-bit 2.8GHz dual-core Opteron, with no multi-threading. We compare the performance of these methods in three ways.

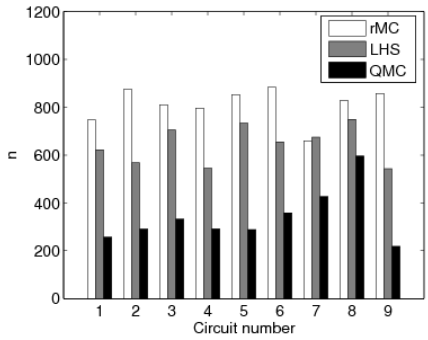


Figure 5: Estimated number of points needed by rMC, LHS and QMC to achieve $\leq 5\%$ error with a 95.45% confidence level.

1) Plot the estimate of σ_d from each run with increasing sample size n. Fig. 3, for example, plots the estimates from 10 runs of rMC (dashed blue lines) and 10 runs of QMC (solid green lines for scrambled Sobol'; solid-dot red line for non-scrambled Sobol') on a log scale of increasing n, for the largest benchmark s38417. We can see that the QMC runs converge faster. A log scale is chosen to highlight the behavior for small n, since we are not very interested in running too many STA runs. The non-scrambled QMC run does converge faster than most other runs.

2) For each algorithm, plot the standard deviation (σ_{σ_d}) of the estimate across all runs, with increasing n. This will show the error convergence behavior. Fig. 4, for example, plots the standard deviation of rMC and QMC on a log-log scale, for the three biggest benchmarks (s35932, s38584, s38417). We can see that QMC has lower error (standard deviation) than rMC for these cases. An exponential convergence of n^{-a} will show as a straight line with a slope of $-a$ on a log-log scale. Hence, we can use least-squared-error linear fits to these curves and estimate the overall convergence exponent for each algorithm on each benchmark. These estimated convergence exponents are shown in Table 1. We can see that MC, rMC and LHS all show similar convergence rates, close to the theoretical Monte Carlo exponent of -0.5. (The estimated rates for MC show some inconsistency because only 10 extremely expensive runs are used to compute them, resulting in large variance.) QMC, however, consistently achieves faster convergence than Monte Carlo. These results do not show the important impact of the y-intercept of the linear fits. The next comparison method, however, does.

Circuit		MC	rMC	LHS	QMC
1	s27	-.62	-.50	-.50	-.55
2	s1196	-.49	-.47	-.53	-.63
3	s5378	-.47	-.56	-.48	-.67
4	s9234	-.60	-.49	-.48	-.56
5	s13207	-.44	-.50	-.50	-.62
6	s15850	-.46	-.48	-.52	-.60
7	s35932	-.49	-.58	-.47	-.65
8	s38584	-.40	-.49	-.50	-.75
9	s38417	-.59	-.50	-.45	-.58

Table 1: Estimated error convergence exponent a for a rate of n^{-a}.

3) Assume that the exact value of σ_d is the estimate computed using the 100,000 total points from all 10 10K-point MC runs. Suppose we want any estimate to lie within $p\%$ of the exact σ_d with a confidence level of 95.45%. Using the Central Limit Theorem [30], this interval is $[\sigma_d \pm 2\sigma_{\sigma_d}]$, where σ_{σ_d} is the standard deviation of the estimate, as plotted in Fig. 4. For the given accuracy criterion then, $\sigma_{\sigma_d} \leq p/200$. Say, we accept errors of up to 5%, then $\sigma_{\sigma_d} = 0.025$ or less. The linear convergence fits can be used

to compute the required sample size n for this value of σ_{σ_d}. Fig. 5 shows the required sample sizes for rMC, LHS and QMC, using our KLE-reduced dimension models. The circuit numbers are the same as in Table 1. We see that LHS provides little benefit over rMC, while QMC more than halves the required sample size on average. The average number of points needed are 813 for rMC, 645 for LHS and 341 for QMC. These results indicate that σ_d as a function of the RVs is not primarily a one-dimensional function – its multi-dimensional components, $f_{>1}$ from (14) are significant. As a result, QMC shows much better performance than LHS. We do not show estimated samples sizes for standard (i.e., unreduced) MC since it is not a real contender due to its large runtime. Also, these estimates will have high variance being computed from only 10 runs. Table 2 shows the actual runtimes for non-scrambled QMC using the required sample sizes for the 5% error criterion, along with the actual errors with respect to a 100,000 point MC (no KLE) run. It also provides the runtimes for this 100K-point MC run for reference. We can see that the runtimes and errors are acceptably small, even for our simple implementation with minimal code optimization.

Runtimes for much larger circuits, or for more sophisticated (and thus slower) core STA engines, will certainly increase. Nevertheless, our results show that it is easy to get accuracy within a few percent of a 100K-point MC run, using only a few hundred STA runs, with smart QMC sampling, coupled with KLE-based dimensionality reduction.

Circuit	n_{QMC}	QMC % error	QMC runtime (s)	100K-pt MC runtime (s)
s27	257	1.73	0.15	1.3
s1196	291	0.80	0.34	288
s5378	333	3.53	1.48	7123
s9234	292	5.17	2.65	27757
s13207	289	1.74	3.98	62883
s15850	359	0.84	5.84	90624
s35932	426	0.25	89.56	266627
s38584	596	1.74	25.25	384154
s38417	219	3.04	9.13	543684
Average	341	2.09	–	–

Table 2: n_{QMC} is the QMC sample size required for error $\leq 5\%$ with 95.45% confidence. Col. 2 shows the actual abs. percentage errors in the estimate of σ_d w.r.t. a 100,000 point MC run. Cols. 3 and 4 show runtime for QMC with n_{QMC} Sobol' points, and the reference 100K-point MC run, respectively.

7 Conclusions

A Monte Carlo type SSTA algorithm can be practical if it is chosen and implemented intelligently. We show how quasi-Monte Carlo sampling, combined with Karhunen Loéve expansion based dimensionality reduction, can result in such an algorithm. We study in detail the performance of standard Monte Carlo, Latin hypercube sampling and QMC in an SSTA setting and show reasons for the superior performance of QMC. Experimental results show that $\leq 5\%$ error with a confidence of 95.45% is achievable with a few hundred sample points (341 on average). Actual observed errors are 2.09% on average and the runtimes range from a few seconds up to 90 seconds for the ISCAS89 circuits. The QMC-based SSTA algorithm makes no assumptions about the gate models, wire models or any aspects of the core STA algorithm, or to any STA operations, unlike most model-based SSTA approaches. Further improvements in QMC [21], optimizations in the implementation and the advent of multi-core processors can make QMC-based SSTA practical for large industrial circuits.

Acknowledgments The authors acknowledge the support of the Semiconductor Research Corporation (SRC), and the support of C2S2, the Focus Center for Circuit & System Solutions, one of five research centers funded under the Focus Center Research Program, a SRC program.

References

[1] H. Chang and S. Sapatnekar, "Statistical timing analysis under spatial correlations," *DAC*, 2005.
[2] C. Visweswariah, et al, "First-order incremental block-based statistical timing analysis," *DAC*, 2004.
[3] J. Xiong et al, "Robust extraction of spatial correlation," *IEEE Trans. CAD*, 26(4), Apr, 2007.
[4] S. Bhardwaj et al, "A framework for statistical timing analysis using non-linear delay and slew models," *ICCAD*, 2006.
[5] A. Singhee and R. A. Rutenbar, "From finance to flip-flops: a study of fast quasi-Monte Carlo methods from computational finance applied to statistical circuit analysis," *ISQED*, 2007.
[6] A. Singhee, "Novel Algorithms for Fast Statistical Analysis of Scaled Circuits," Ph.D. thesis, Carnegie Mellon U., 2007.
[7] A. Singhee, S. Singhal and R. A. Rutenbar, "Exploiting correlation kernels for efficient handling of intra-die spatial correlation, with application to statistical timing," *DATE*, 2008.
[8] H. Chang et al, "Parameterized block-based statistical timing analysis with non-Gaussian parameters, nonlinear delay functions,' *DAC*, 2005.
[9] J. Singh et al, "Statistical timing analysis with correlated non-Gaussian parameters using independent component analysis," *DAC*, 2006.
[10] A. Ramalingam et al, "An accurate sparse matrix based framework for statistical static timing analysis," *ICCAD*, 2006.
[11] P. Glasserman, "Monte Carlo Methods in Financial Engineering", Springer, 2004.
[12] M. Loéve, "Probability Theory I & II", Springer, 4 ed., 1977.
[13] M. Hane et al, "Atomistic 3D process/device simulation considering gate line-edge roughness and poly-Si random crystal orientation effects," *IEDM*, 2003.
[14] P. Friedberg et al, "Modeling within-die spatial correlation effects for process-design co-optimization,", *ISQED*, 2005.
[15] A. Agarwal et al, "Statistical delay computation considering spatial correlations," *ASPDAC*, 2003.
[16] G. S. Fishman, "A First Course in Monte Carlo," Duxbury, 2006.
[17] M. D. McKay et al, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, 21(2), 1979.
[18] M. Stein, "Large sample properties of simulations using Latin hypercube sampling," *Technometrics*, 29(2), 1987.
[19] E. Hlawka, "Functionen von beschränkter variation in der theori der gleichverteilung (in german)," *Annali di Matematica Pura ed Applicata*, 54, 1961.
[20] J. Kiefer, "On large deviations of the empirical d. f. of vector chance variables and a law of the iterated logarithm," *Pacific J. Math.*, 11, 1961.
[21] H. Niederreiter et al, "The algebraic geometric approach to low-discrepancy sequences," *Monte Carlo and Quasi-Monte Carlo Methods 1996*, Springer, 1998.
[22] X. Wang et al, "Low discrepancy sequences in high dimensions: how well are their projections distributed?" *J. Comput. Appl. Math.*, doi: 10.1016/j.cam.2007.01.005, 2007.
[23] A. B. Owen, "Randomly permuted (t,m,s)-nets and (t,s)-sequences," *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, 299–317, Springer, 1995.
[24] H. S. Hong et al, "Algorithm 823: implementing scrambled digital sequences," *ACM Trans. Math. Soft.*, 29(2), 2003.
[25] J. Rubenstein et al, "Signal delay in RC tree networks," *IEEE Trans. CAD*, CAD-2, Jul, 1983.
[26] C. V. Kashyap et al, "Closed-form expressions for extending step delay and slew metrics to ramp inputs for RC trees," *IEEE Trans. CAD*, 23(4), Apr, 2004.
[27] H. B. Bakoglu, "Circuits, interconnections, and packaging for VLSI," Addison-Wesley, 1990.
[28] X. Li et al, "Projection-based performance modeling for inter/intra-die variations," *ICCAD*, 2005.
[29] A. E. Caldwell et al, "Can recursive bisection alone produce routable placements?," *DAC*, 2000.
[30] R. V. Hogg and A. T. Craig, " Introduction to Mathematical Statistics," 3rd ed., Macmillan, 1971.